Confidence-based Ant Colony Optimization for Capacitated Electric Vehicle Routing Problem with Comparison of Different Encoding Schemes

Ya-Hui Jia, *Member, IEEE,* Yi Mei, *Senior Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—The blossoming of electric vehicles gives rise to a new vehicle routing problem called capacitated electric vehicle routing problem. Since charging is not as convenient as refueling, both the service of customers and the recharging of vehicles should be considered. In this paper, we propose a confidence-based bi-level ant colony optimization algorithm to solve the problem. It divides the whole problem into the upper-level sub-problem capacitated vehicle routing problem and the lower-level sub-problem fixed routing vehicle charging problem. For the upper-level sub-problem, an ant colony optimization algorithm is used to generate customer service sequence. Both the direct encoding scheme and the order-first split-second encoding scheme are implemented to make a guideline of their applicable scenes. For the lower-level sub-problem, a new heuristic called simple enumeration is proposed to generate recharging schedules for vehicles. Between the two sub-problems, a confidence-based selection method is proposed to select promising customer service sequence to conduct local search and lower-level optimization. By setting adaptive confidence thresholds, the inferior service sequences that have little chance to become the iteration best are eliminated during the execution. Experiments show that the proposed algorithm has reached the state-of-the-art level and updated eight best known solutions of the benchmark.**

***Index Terms*—Capacitated Electric Vehicle Routing Problem, Ant Colony Optimization, Bi-level Optimization, Combinatorial Optimization.**

I. Introduction

VEHICLE routing problems (VRPs) have been widely studied since it was first proposed by Dantzig and Ramser in 1959 [1]. During the past few decades, many VRP variants were proposed to model different real-world applications [2]–[7]. Since fossil fuel is definitely the dominant energy source in the last century, most vehicles considered in VRPs are fossil-fueled vehicles. Benefited from the widespread gas stations and short refueling time, the refueling problem is usually not considered in VRPs [8]. New energy techniques have seen great development in the past few years, which has promoted the development of new energy vehicles, especially electric vehicles (EVs) [9]. Compared with fossil-fueled vehicles, EVs are more environmentally friendly [10]. To pursue the goal of carbon neutrality, logistics companies have already started to use EVs in their daily business [11].

EVs bring new opportunities as well as new challenges to the industry. Currently, the cruising range of EVs is still shorter than conventional vehicles and the number of charging stations is far less than the number of petrol stations as well [12]. Thus, when EVs are adopted in real-world logistics applications, not only the service orders of the customers but also the recharging schedules of the vehicles should be considered. This extra concern leads to a new category of VRP variants called electric VRP (EVRP). Corresponding to some traditional VRP variants, several EVRP variants were proposed in the past few years [13], such as capacitated EVRP (CEVRP) [8], EVRP with time window (EVRPTW) [14], and EVRP with pickup and delivery (EVRPPD) [15]. In this paper, we focus on CEVRP. The objective of CEVRP is to minimize the total travel distance of all EVs under several constraints. Classic constraints about customer serving and vehicle capacity that every customer needs to be served once and vehicles cannot be overloaded are still kept. A new constraint about energy is added that EVs cannot run out of electricity during the journey. If the initial capacity of the battery cannot support the EV to serve all the customers in its route, it can recharge the battery at charging stations on its way [8].

So far, the proposed methods to solve EVRPs can be roughly classified into three categories, exact algorithms, individual-based meta-heuristic algorithms, and population-based meta-heuristic algorithms. Exact algorithms transfer EVRPs into corresponding mixed integer linear programming (MILP) models to solve them [16]–[18]. They can generate very good solutions on small-scale problems, but the high computational complexity makes exact algorithms inefficient on larger problems with more than 50 customers [19]. To solve large-scale EVRPs, a number of individual-based meta-heuristic algorithms have been proposed by combining different local search methods and perturbation methods [14], [20], [21]. These algorithms can be very effective, but their effectiveness highly depends on the operators and the quality of the initial solutions, which makes them easy to be trapped into poor local optima [22]. Because of the good global search ability, population-based meta-heuristic algorithms were also applied [8], [23], [24]. Ant colony optimization (ACO) [8] is an outstanding one among them since it can utilize problem information well during the solution construction.

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1614, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, and the MBIE SSIF Fund under Contract VUW RTVU1914.

Ya-Hui Jia is with School of Future Technology, South China University of Technology, Guangzhou, China and also with School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. (email: jia.yahui@foxmail.com)

Yi Mei, and Mengjie Zhang are with School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. (email:yi.mei@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz)

ACO has been widely used to solve different VRPs [8], [25]–[27]. The encoding scheme is always a major concern [28]. Currently, there are two kinds of solution encoding schemes commonly used in meta-heuristic algorithms to solve VRPs: 1) direct and 2) indirect. The direct encoding scheme encodes a solution directly as a set of different routes, each route being a sequence of services by a vehicle [26]. The indirect encoding scheme encodes a solution as a giant tour of all customers, which is decoded into a set of feasible routes by an optimal split (i.e., dynamic programming) [29]. The indirect encoding is also known as the order-first split-second encoding [30]. The direct encoding scheme has been widely adopted in ACO [25]–[27], but since it is hard to decide the returning timing of vehicles (ACO typically opens a new route only when the current route is full), using direct encoding in ACO usually requires a good inter-route local search method. Indirect encoding does not need the aid of inter-route local search methods, but using indirect encoding in ACO would cause a mismatch between the pheromone matrix and the routes since the pheromone matrix is updated according to the giant tour rather than the routes. Both encoding schemes have their pros and cons. To the best of our knowledge, there is no guideline so far about which one is more suitable for ACO and when we should use one or another.

Meanwhile, when ACO is applied to CEVRP, there is an extra concern about vehicle recharging. Considering the serving order of customers and the recharging schedule of vehicles simultaneously leads to a huge and complex search space [19]. Treating CEVRP as a bi-level optimization problem is a promising way to decrease the complexity of the problem where the serving order of customers is the upper-level sub-problem and the recharging schedule of vehicles is the lower-level sub-problem [8]. In this regard, which sub-solution of the upper-level sub-problem should be fed into the lower-level sub-problem is a critical issue.

To address the aforementioned issues and challenges, we propose a confidence-based bi-level ACO (CBACO) algorithm in this paper to solve CEVRP effectively and efficiently. The contributions of CBACO are shown as follows:

- A confidence-based selection strategy is proposed to choose promising sub-solutions of the upper-level sub-problem to conduct local search and lower-level optimization. The confidence-based selection strategy can save nearly half of the execution time on large-scale instances, thus improving the efficiency of the algorithm.
- The effectiveness and efficiency of both the direct and indirect encoding schemes are investigated in the context of solving CEVRP with ACO. Accordingly, a guideline of under what circumstances an encoding scheme should be used is given. In addition, for the direct encoding scheme, an improved 2opt* operator is proposed as the inter-route local search method.
- A new heuristic called simple enumeration (SE) is proposed in CBACO to generate good recharging schedules for the lower-level sub-problem, fixed routing vehicle charging problem (FRVCP). The complexities of SE and FRVCP are analyzed both theoretically and empirically to gain an insight into CEVRP.

The rest of this paper is organized as follows. Section II shows the related work. Section III gives the formal description of CEVRP. The proposed algorithm CBACO is introduced in Section IV in detail. The experiments are set up in Section V and the results are analyzed in Section VI. Finally, the conclusions are drawn in Section VII.

II. Related Work

In this section, the methods proposed for different EVRPs are reviewed. These methods can be roughly classified into three categories: 1) exact methods, 2) individual-based meta-heuristic methods, and 3) population-based meta-heuristic methods.

The exact methods generate optimal solutions for EVRPs by transferring the problem into a corresponding mathematical programming model and solve them by mathematical programming software such as CPLEX and Gurobi. Lin [17] built a MILP model for an EVRP that considered the influence of load on the energy consumption of the EV and used CPLEX to solve the problem. The experiments showed that CPLEX found the optimal solution for a small-scale case with 13 customers. Chen *et al.* [15] modeled EVRPPD as a mixed-integer quadratically constrained programming problem and solved several instances with no more than 20 customers by CPLEX successfully. Xiao *et al.* [31] proposed a comprehensive energy consumption model for EVRPTW and applied CPLEX as well. The scale of the experiment was also small that no more than 30 customers were considered. Montoya *et al.* [19] studied another EVRP variant that considered the non-linear charging characteristic of EV and they have found the optimal solutions for several instances with less than 50 customers by Gurobi. These studies showed that the exact methods and the corresponding software are effective for small-scale problems, but they are inefficient when handling large-scale problems. This fact is not surprising since these methods originally cannot solve the traditional large-scale VRPs well [32] and adding the battery constraint makes EVRPs even harder than VRPs.

Individual-based meta-heuristic methods have been widely applied to solve many large-scale combinatorial optimization problems [22], and EVRPs are no exception. Many individual-based meta-heuristic algorithms have been applied to EVRPs, including tabu search (TS) [33], simulate annealing (SA) [34], iterative local search (ILS) [22], variable neighborhood search (VNS) [35], and adaptive large neighborhood search (ALNS) [5]. EVRPTW was firstly proposed in 2014 by Schneider *et al.* [14] and Afroditi *et al.* [36]. Along with the problem, Schneider also designed a individual-based meta-heuristic algorithm that combined TS with VNS to solve the problem. Following their studies, Bruglieri *et al.* [37], [38] changed the objective from distance to time and added a local branching method into the VNS algorithm. During the IEEE WCCI2020 competition on EC for EVRP, the VNS algorithm proposed by Woller and the SA algorithm proposed by Mak-Hau performed very well and ranked in the first and the second places [39]. Montoya *et al.* [19] proposed an ILS algorithm for the EVRP with non-linear charging and achieved good results. ALNS was also

used in several works and showed good performance [21], [40], [41]. Generally, the effectiveness of these algorithms is commendable but they highly rely on the applied local search methods, which are problem-dependent. Thus, they may frequently encounter the premature convergence problem if the perturbation or destruction methods do not work well [22].

Population-based meta-heuristic algorithms have been tried in some works including GA and ACO. Guo [23] and Shao [24] proposed two GA methods for EVRPTW. Due to inappropriate encoding schemes, these two algorithms are only effective for some small-scale problems. Previously, we proposed a bi-level ACO (BACO) algorithm for CEVRP that decomposed the problem into two levels, CVRP and FRVCP [8]. For the upper-level sub-problem CVRP, the sub-solutions of the serving orders of customers are generated only considering the capacity constraint, which may violate the battery constraint. Then, these sub-solutions are repaired at the lower level that the recharging schedules are generated to make them energy feasible. According to the difficulty of each level, we proposed OS-MMAS for CVRP and removal heuristic (RH) for FRVCP. Although BACO has outperformed the winners of the IEEE WCCI2020 competition of EVRP, we found that using indirect encoding scheme in ACO would cause a mismatching between the generated routes and the pheromone matrix. Also, as a bi-level optimization algorithm, which sub-solutions of the upper level sub-problem should be refined and sent to the lower level is still an open issue for BACO. These two problems motivate us to investigate both the direct and indirect encoding schemes and to propose CBACO.

III. Problem Definition

CEVRP is proposed in [42] by extending the traditional CVRP. The "CEVRP" studied in this paper means exactly the "EVRP" in [42]. To follow a standard nomenclature, we added "capacitated" to the name of the problem. A CEVRP is usually defined on a fully connected weighted graph $G = (V, E)$. There are three kinds of vertices $V = \{0\} \cup V_c \cup V_f$: depot $\{0\}$, customers V_c, and charging stations V_f. Depot is the place where every EV departs from and returns to. In CEVRP, only one depot is considered and it is indexed as 0. Customers V_c represent the tasks that need to be executed. Each customer $i \in V_c$ has a fixed number of cargo demand, denoted as c_i. Charging stations V_f provide charging service to EVs. Each edge $(i, j) \in E$ has a fixed weighted value d_{ij} representing the distance between i and j. Besides the load limit Q_c like conventional vehicles, EVs also have a battery limit Q_b. The consumption rate of battery is defined as r_b that $d_{ij} \cdot r_b$ amount of battery will be consumed if (i, j) is traversed.

For mathematical modeling, an extended set of charging stations $\hat{V_f}$ is introduced where each charging station $i \in V_f$ is copied $2|V_c|$ times[1]. Two more variables u_i and y_i are added to represent the remaining load capacity and the remaining battery of an EV when it arrives at i. Defining the variable x_{ij} that if (i, j) is traversed by an EV, $x_{ij} = 1$, otherwise $x_{ij} = 0$, the mathematical model of CEVRP is given as follows [42]:

[1]In the worse case, each EV serves only one customer and all EVs need to recharge twice at the same recharging station on the way to serve customers and on the way back, respectively.

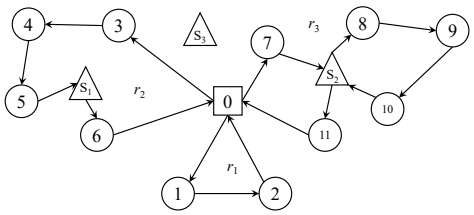


Fig. 1. An example solution of CEVRP with three routes.

$$\min\ F(\mathbf{x}) = \sum_{i \in V, j \in V, i \neq j} d_{ij} x_{ij}, \tag{1}$$

s.t.

$$\sum_{j \in V, i \neq j} x_{ij} = 1, \forall i \in V_c, \tag{2}$$

$$\sum_{j \in V, i \neq j} x_{ij} \leq 1, \forall i \in \hat{V_f}, \tag{3}$$

$$\sum_{j \in V, i \neq j} x_{ij} - \sum_{j \in V, i \neq j} x_{ji} = 0, \forall i \in V, \tag{4}$$

$$u_j \leq u_i - c_i x_{ij} + Q_c(1 - x_{ij}), \forall i \in V, \forall j \in V, i \neq j, \tag{5}$$

$$0 \leq u_i \leq Q_c, \forall i \in V, \tag{6}$$

$$y_j \leq y_i - r_b d_{ij} x_{ij} + Q_b(1 - x_{ij}), \forall i \in V_c, \forall j \in V, i \neq j, \tag{7}$$

$$y_j \leq Q_b - r_b d_{ij} x_{ij}, \forall i \in \hat{V_f} \cup \{0\}, \forall j \in V, i \neq j, \tag{8}$$

$$0 \leq y_i \leq Q_b, \forall i \in V, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \forall i \in V, \forall j \in V, i \neq j. \tag{10}$$

(1) shows the objective of CEVRP that is to minimize the total traveling distance of all vehicles. (2) indicates that each customer should be served exactly once. (3) literally means that each copy of a charging station can be visited no more than once. Since each charging station has $2|V_c|$ copies, (3) actually means that each charging station can be visited at most $2|V_c|$ times. To explain (3) clearly, an example solution is shown in Fig. 1. Since there are 11 customers, each station has $2 \times 11 = 22$ copies. The first route r_1 does not visit any charging station. The second route r_2 visits the charging station S_1 once, meaning that one S_1 copy is visited and the other copies are not visited. The third route r_3 visits the charging station S_2 twice, meaning that two S_2 copies are visited and the other copies are not visited. All copies of S_3 are not visited. (4) represents the flow conservation that the indegree (vehicle entering) of a node should be equal to its outdegree (vehicle leaving). (5) and (6) represent the capacity constraint that an EV cannot be overloaded during the trip and each customer's demand must be fully served. (7), (8), and (9) represent the battery constraint that an EV cannot run out of battery during the trip. Besides these constraints, there is another assumption indicated by (7), (8), and (9) that for a feasible route, an EV will always recharge its battery enough at a station to reach the next charging station or the depot. (10) defines the value range of the variable.

IV. Confidence-based Bi-level Ant Colony Optimization

In this section, CBACO is introduced in detail. First, the overall process with the confidence-based selection method is explained. Then, the sub-solution construction methods of customer serving schedules with different encoding schemes and the local search methods are illustrated. Afterward, the newly proposed SE heuristic is shown.

A. Overall Process with Confidence-based Selection

1) Process: The overall process of CBACO is shown in Algorithm 1. At the beginning, the algorithm components and parameters are initialized, and the global best solution $\mathbf{x}_{gb}$ is initialized by the nearest neighbor and SE (lines 1-2). In each generation, the applied ACO algorithm first builds capacity-feasible sub-solutions of the upper-level sub-problem CVRP without considering the battery constraint (line 4). Then, through confidence-based selection, promising sub-solutions are chosen to be refined (lines 5-22). The refined sub-solutions are further screened to get the sub-solutions that have a chance to generate the iteration best solution to conduct the lower-level optimization (lines 23-29). Recharging schedules are generated by SE to repair the selected sub-solutions to be electricity-feasible (lines 30-34). Afterward, these feasible solutions are evaluated and the pheromone information is updated according to the iteration best solution (lines 35-36). Whenever the stopping criterion is met, the recharging schedule of the global best solution is finally improved by the restricted enumeration method that we proposed in our previous work [8] and returned.

2) Confidence-based Selection: When local search methods are adopted to enhance the effectiveness of a population-based meta-heuristic algorithm, a common problem is how to select the promising solutions to conduct local search. A similar problem, which sub-solution of the upper-level sub-problem should be considered for the lower-level optimization, is also frequently encountered when a population-based meta-heuristic algorithm is used to solve a bi-level optimization problem. Referring to memetic algorithms [43], there are mainly three selection methods: 1) only selecting the iteration best or overall best solutions [44]; 2) selecting all solutions [45]; 3) selecting some solutions according to some strategies [46]. The first method is efficient but not effective sometimes since it may miss many promising solutions. The second method is effective but not efficient since many poor solutions are included. The third one can be both effective and efficient, although it is usually problem-dependent that requires expertise to design. For VRPs, usually the second strategy that selects all is adopted [29], [45].

In CBACO, we propose a confidence-based selection method in order to increase the efficiency of BACO without losing effectiveness. The confidence-based selection method belongs to the third category that selects some sub-solutions. The basic idea of the confidence-based selection is to knock the sub-solutions that have small probability to be the iteration best out. Unlike some other bi-level optimization problems that the sub-solution of the upper-level sub-problem is hard to be evaluated individually [47], the customer serving schedules in CEVRP can be evaluated individually just like normal CVRP. We denote the total length of a sub-solution $\mathbf{x}_i^u$ as $f'(\mathbf{x}_i^u)$. It is worth noting that both local search and lower-level optimization may change the rank of the sub-solutions, meaning that a best sub-solution $\mathbf{x}^u$ in terms of f' is not necessarily the best final solution $\mathbf{x}$.

Based on f', the confidence-based selection method is conducted by screening the sub-solutions according to a selected exemplar and an estimated confidence degree. If within the confidence degree a sub-solution is possible to surpass the exemplar, we include the sub-solution to do following executions. Specifically, to select promising sub-solutions for local

Algorithm 1: CBACO

Input: confidence ratio of local search γ_l, confidence ratio of recharging γ_r, confidence interval δ, fitness function F, fitness function f' without considering recharging.
Output: final solution $\mathbf{x}_{gb}$

1 initialize pheromone information and $\mathbf{x}_{gb}$;
2 P = empty queue; $r = 0$; $g = 0$;
3 **while** *the stopping criterion is not met* **do**
4 build capacity-feasible sub-solutions $S = \{\mathbf{x}_1^u, \ldots, \mathbf{x}_m^u\}$ by ACO;
5 $S_1 = S$;
6 **if** $g > \delta$ **then**
7 $\mathbf{x}_a^u = \arg\min_{\mathbf{x}_i^u \in S} f'(\mathbf{x}_i^u)$;
8 local search on $\mathbf{x}_a^u$ and get $\hat{\mathbf{x}_a^u}$;
9 $v_1 = f'(\mathbf{x}_a^u) - f'(\hat{\mathbf{x}_a^u})$;
10 $v_2 = \max(P)$;
11 **if** $v_2 < v_1$ **then**
12 $v_2 = v_1 \cdot \gamma_l$;
13 **end**
14 $S_1 = \{\mathbf{x}_i^u | \mathbf{x}_i^u \in S \wedge f'(\mathbf{x}_i^u) - v_2 \leq f'(\hat{\mathbf{x}_a^u})\}$;
15 **end**
16 local search on the sub-solutions in S_1 and get $\hat{S_1}$;
17 $v_2 = \max(\{f'(\mathbf{x}_i^u) - f'(\hat{\mathbf{x}_i^u}) | \mathbf{x}_i^u \in S_1, \hat{\mathbf{x}_i^u} \in \hat{S_1}\})$;
18 $v_2 = \max(v_1, v_2)$;
19 push v_2 into the back of P;
20 **if** $|P| > \delta$ **then**
21 pop the first value in P;
22 **end**
23 $\mathbf{x}_b^u = \arg\min_{\hat{\mathbf{x}_i^u} \in \hat{S_1}} f'(\hat{\mathbf{x}_i^u})$;
24 generate $\mathbf{x}_b^l$ according to $\mathbf{x}_b^u$ by SE, denote the corresponding solution as $\mathbf{x}_b = (\mathbf{x}_b^u, \mathbf{x}_b^l)$;
25 $v_3 = F(\mathbf{x}_b) - f'(\mathbf{x}_b^u)$;
26 **if** $r > v_3$ **then**
27 $r = v_3 \cdot \gamma_r$;
28 **end**
29 $S_2 = \{\hat{\mathbf{x}_i^u} | \hat{\mathbf{x}_i^u} \in \hat{S_1} \wedge f'(\hat{\mathbf{x}_i^u}) + r \leq F(\mathbf{x}_b)\}$;
30 generate $\mathbf{x}_i^l$ for $\hat{\mathbf{x}_i^u} \in S_2$, get $S_3 = \{\mathbf{x}_i | \hat{\mathbf{x}_i^u} \in S_2\}$;
31 $v_3 = \min(v_3, \{F(\mathbf{x}_i) - f'(\hat{\mathbf{x}_i^u}) | \mathbf{x}_i \in S_3, \hat{\mathbf{x}_i^u} \in S_2\})$;
32 **if** $r == 0$ *or* $r > v_3$ **then**
33 $r = v_3$;
34 **end**
35 find the iteration best solution $\mathbf{x}_{ib}$ in $S_3 \cup \{\mathbf{x}_b\}$;
36 update $\mathbf{x}_{gb}$ and pheromone information by $\mathbf{x}_{ib}$;
37 $g = g + 1$;
38 **end**
39 adjust the recharging schedules of $\mathbf{x}_{gb}$ by the restricted enumeration method;
40 **return** $\mathbf{x}_{gb}$;

search, after ACO builds the capacity-feasible sub-solutions (line 4), we find the sub-solution with the best f' as the exemplar $\mathbf{x}_a^u$ and conduct local search on it (lines 7-8). The improvement of $\mathbf{x}_a^u$ in terms of f', denoted as v_1, is compared with the largest improvement v_2 that we recorded in the past δ generations. If v_1 is larger, we use it multiplying a ratio γ_l as the confidence degree where $\gamma_l \geq 1$, otherwise v_2 is set as the confidence degree (lines 9-13). According to the exemplar and the confidence degree, the promising sub-solutions are selected (line 14). These sub-solutions will be improved by the local search methods, and the largest improvement is recorded (lines 16-22).

Then, among these improved sub-solutions, again we select the sub-solution with the best f' as the exemplar $\mathbf{x}_b^u$ (line 23). Its corresponding recharging schedule $\mathbf{x}_b^l$ is generated by SE (line 24). Afterward, we check how much extra length v_3 is brought by $\mathbf{x}_b^l$. This value is compared with r, which is the minimum extra length brought by recharging schedules that we have ever checked during evolution. If r is larger than v_3, r decreases to $v_3 \cdot \gamma_r$ where $\gamma_r \leq 1$ (lines 25-28). Using r as the confidence degree, we select promising sub-solutions to generate corresponding schedules for them (lines 29-30). Then, r is updated (lines 31-34).

The basic idea of these two steps of selection is similar, but the ways to set the confidence degree are different. For the confidence degree of the local search step, a larger value would recruit more sub-solutions. What we want is a large enough value to cover more sub-solutions meanwhile small enough to block the poor sub-solutions. Generally, the improvement brought by the local search methods decreases along with the execution of the algorithm since the solutions found by the algorithm will be closer and closer to some optima. Thus, we use the largest improvement of the previous δ generations as the confidence degree. Contrast to the local search step, for the confidence degree of the lower-level optimization step, a smaller value would recruit more sub-solutions. Thus, we use the smallest extra distance brought the recharging schedule that we have ever found as the confidence degree. γ_l and γ_r are used when the chosen exemplars give the confidence degree.

B. ACO for Route Construction

Pheromone Setting: ACO uses a constructive way to build routes that each time an ant chooses the next customer to visit based on the pheromone value and the heuristic information like the distance between the next customer and its current location. In CBACO, we use the max-min ant system (MMAS) algorithm for route construction because of its better capability than the other ACO variants [48]. Since the charging stations are not considered in this step, the size of the pheromone matrix Φ is $n \times n$ where $n = |V_c| + 1$, each element φ_{ij} representing the pheromone value of going to j from i. Besides, there are two boundaries of the pheromone values that are calculated as:

$$\varphi_{\max} = \frac{1}{(1-\rho) \cdot f(\mathbf{x}_{gb})}, \tag{11}$$

$$\varphi_{\min} = \frac{\varphi_{\max}(1 - \sqrt[n]{pr})}{(n/2 - 1)\sqrt[n]{pr}}, \tag{12}$$

Algorithm 2: Route Construction (**Direct** Encoding)

Input: pheromone matrix Φ, edge set E, customer set V_c, capacity constraint Q_c, the number of vehicles K.
Output: a set of capacity-feasible routes $\mathbf{\Gamma}$. //$\mathbf{\Gamma} == \mathbf{x}^u$

1 $k = 0$, append 0 to Γ_k, $Q_k = 0$;
2 **while** V_c *is not empty* **do**
3 $V_c' = \{i | i \in V_c \wedge c_i \leq Q_c - Q_k\}$;
4 **if** $\sum_{i \in V_c} c_i \leq Q_c \cdot (K - k - 1)$ *or* $V_c' == \varnothing$ **then**
5 add depot 0 into V_c';
6 **end**
7 take the last node i in Γ_k;
8 $j = roulette\_wheel\_selection(V_c', i)$;
9 append j to Γ_k;
10 **if** $j == 0$ **then**
11 append Γ_k to $\mathbf{\Gamma}$;
12 $k = k + 1$, append 0 to Γ_k, $Q_k = 0$;
13 **else**
14 $Q_k = Q_k + c_j$;
15 remove j from V_c;
16 **end**
17 **end**
18 append 0 to Γ_k, append Γ_k to $\mathbf{\Gamma}$;
19 **return** $\mathbf{\Gamma}$;

Algorithm 3: Route Construction (**Indirect** Encoding)

Input: pheromone matrix Φ, edge set E, customer set V_c, capacity constraint Q_c.
Output: a set of capacity-feasible routes $\mathbf{\Gamma}$. //$\mathbf{\Gamma} == \mathbf{x}^u$

1 $T = [0]$, $V_c' = V_c$;
2 **while** V_c' *is not empty* **do**
3 take the last node i in T;
4 $j = roulette\_wheel\_selection(V_c', i)$;
5 append j to T;
6 remove j from V_c';
7 **end**
8 $\mathbf{\Gamma} = split(T, Q_c)$;
9 **return** $\mathbf{\Gamma}$;

where ρ is the evaporation rate. pr is usually set to 0.05.

Route Construction: There are mainly two encoding schemes that were widely used in meta-heuristic algorithms for route construction of CVRP, direct encoding and indirect encoding [30]. In the direct encoding scheme, a solution is directly encoded as different routes, each route representing a vehicle. In the indirect encoding scheme, a solution is encoded as a giant tour that contains all customers. The routes of different vehicles are obtained through a splitting process.

The route construction process of ACO with the two encoding schemes are shown in Algorithm 2 and Algorithm 3, corresponding to line 4 of Algorithm 1. For the direct encoding scheme (Algorithm 2), we initialize a route starting from 0 in the first place (line 1). Then, for every step of route construction, a candidate list V_c' is organized. All the customers that have not been served and have less cargo demand than the remaining capacity of the EV are put into V_c' (line 3). We also need to judge whether the depot is allowed to be added into V_c' (lines 4-6). An ant selects the next node from V_c' through a roulette wheel selection strategy where the

probability of each candidate node j is calculated by:

$$p_{ij} = \frac{\varphi_{ij}^{\alpha}/d_{ij}^{\beta}}{\sum_{l \in V_c'} \varphi_{il}^{\alpha}/d_{il}^{\beta}}, \text{ if } j \in V_c' \tag{13}$$

where α and β are two parameters to adjust the weights of pheromone and heuristic information (lines 7-8). The selected node is appended to the route (line 9). If the node is a customer, the route construction process continues (lines 13-16). If it is the depot, we starts a new route (lines 10-12).

For direct encoding, when the vehicle is allowed to return is a problem, i.e. when the candidate node set V_c' should include the depot 0. A simple strategy is to allow a vehicle to return only when it cannot serve any more customer. However, this returning strategy often lead to bad results. We use an example shown in Fig. 2(a) to explain. Suppose we have five customers. Three of them {1,2,3} require one unit cargo each and the other two {4,5} require two units cargo each. The capacity of a vehicle is five. Assume a vehicle departs from depot 0 and already chooses customer 5 and 4 successively. Since it can still serve one customer with one unit cargo demand, it will choose customer 3 and then return to depot. Then, the second vehicle will serve customer 1 and 2. However, the total length of these two routes {[1,2], [5,4,3]} is 16 that is longer than the optimal routes {[1,2,3], [4,5]} with length 15. In Algorithm 2, we propose a new strategy that when the total demand of all remaining customers is less than the total capacity of all remaining vehicles or when there is no feasible customer, the depot is added into the candidate set V_c' for the vehicle to choose (lines 4-6). If the number of vehicles K is not provided, it can be estimated as $K = \lceil \sum_{i \in V} c_i / Q_c \rceil$. This strategy can alleviate the problem shown in Fig. 2(a), although it cannot radically solve it. There are some other strategies, such as setting a fixed number depot copies and always including depot in V_c' if the vehicle is not in the depot [27]. However, these strategies may lead to capacity-infeasible routes or excess number of routes. No matter which strategy is used, the direct encoding scheme usually requires a good inter-route local search method to amend the generated routes.

For the indirect encoding (Algorithm 3), starting from the depot (line 1), it builds a giant tour containing all customers first (lines 2-7). In each step of the tour construction, the next customer is also selected through a roulette wheel selection with the same probability calculation method (13). After all customers are inserted into the giant tour, a splitting algorithm [29] is applied to partition the giant tour into capacity-feasible routes (line 8). The splitting algorithm is a dynamic programming method that enumerates all feasible partitions and choose the one with shortest total length. Thus, we do not need to consider the returning problem in the indirection encoding scheme. However, from Fig. 2(b), we can find that when building the giant tour, the move from customer 2 to 3 is actually different from the move from customer 3 to 4 since the edge (3, 4) is finally cut by the splitting algorithm. Thus, there is mismatching between the final routes and the pheromone matrix. This mismatching problem may affect the performance of the algorithm if the length of the edges linked to depot occupies a large proportion of the total length.

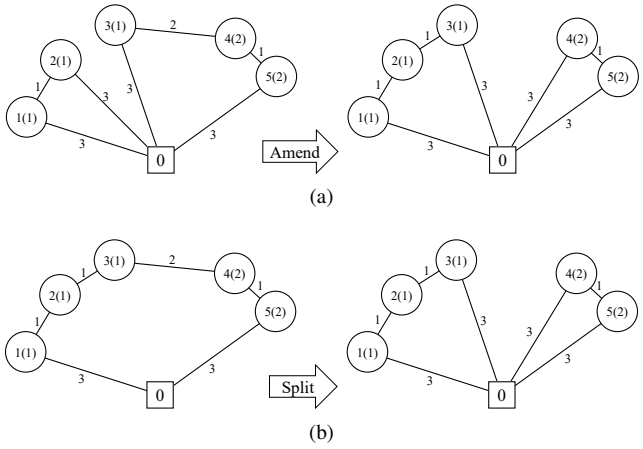


Fig. 2. Different encoding schemes. (a) direct encoding, (b) indirect encoding.

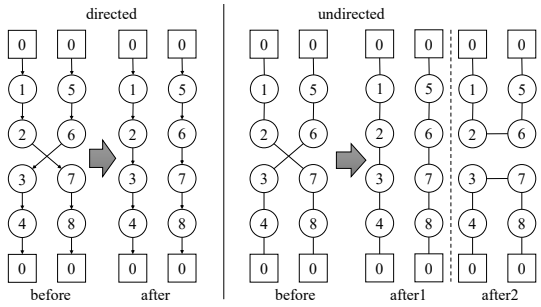


Fig. 3. 2opt* operation for directed and undirected routes.

So far, there is not a comprehensive comparison that shows which encoding scheme suits ACO more. In this paper, we use both encoding schemes in CBACO to see which one is better. For the direct encoding scheme, we propose an improved 2-opt* method as the inter-route local search method. 2-opt* is a famous inter-route operator that was originally proposed for the directed routes of VRPTW [49]. For each pair of links from two different routes, 2-opt* first breaks them and then swaps the tails of the two routes. If the swap leads to a better solution, the new solution is accepted. Otherwise, a new pair of links is tried. Since the routes are undirected in CEVRP, we consider one more situation where the head of a route is swapped with the tail of another route. Fig. 3 gives the demonstration of the improved 2-opt* operator. Following [49], for the intra-route local search methods, we choose 2-opt and node-shift. 2-opt replaces two non-adjacent edges $(i, i+)$ and $(j, j+)$ by (i, j) and $(i+, j+)$, where $i+$ and $j+$ represent the nodes after i and j in the route. Node-shift shifts a node to another position in the route. Because of the complexity of each operator, the first improvement operation is applied to 2-opt* and the best improvement operation is applied to 2-opt and node-shift [50]. The sequence of using these three operators also follows [49] that 2-opt, 2-opt*, and node-shift are used successively. For the indirect encoding, we keep our original settings of BACO that only 2-opt is applied.

Pheromone Updating: After local search and lower-level optimization, we can get some feasible solutions. At the end of each iteration, we find the iteration best solution $\mathbf{x}_{ib}$ among the feasible solutions and use it to update the

pheromone values (lines 35-36, Algorithm 1). If the iteration best solution $\mathbf{x}_{ib}$ is better than the global best solution $\mathbf{x}_{gb}$, $\mathbf{x}_{gb}$ is updated $\mathbf{x}_{gb} = \mathbf{x}_{ib}$. Then, the two pheromone boundaries $\varphi_{\max}$ and $\varphi_{\min}$ are updated according to (11) and (12). Finally, each pheromone value in the pheromone matrix is updated according to:

$$\varphi_{ij}(t+1) = \min(\max(\rho \cdot \varphi_{ij}(t) + \Delta\varphi_{ij}^{\text{best}}, \varphi_{\min}), \varphi_{\max}) \tag{14}$$

$$\Delta\varphi_{ij}^{\text{best}} = \begin{cases} 1/f(\mathbf{x}_{ib}) & \text{if } (i,j) \in \mathbf{\Gamma}_{ib} \text{ (direct)} \\ 1/f(\mathbf{x}_{ib}) & \text{if } (i,j) \in T_{ib} \text{ (indirect)} \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

It should be noted that when the direct encoding scheme is used, the pheromone matrix is updated according to the capacity-feasible routes $\mathbf{\Gamma}$ corresponding to $\mathbf{x}_{ib}^{u}$. When the indirect encoding scheme is used, the pheromone matrix is updated according to the giant tour T rather than the capacity-feasible routes generated by splitting.

C. Simple Enumeration for Recharging Schedule Generation

In CBACO, we use a new heuristic method called simple enumeration to generate the recharging schedules for the lower-level optimization FRVCP. Algorithm 4 shows the process of SE. First, the best station between each pair of nodes is found (lines 1-3). This process can be done in advance as a pre-processing so that we do not need to repeat this process every time when we use SE. Then, we estimate how many stations are needed by calculating the total length of the route (lines 4-5). lb represents the minimum number of stations that are needed. Afterward, we consider to insert $n = lb$ and $n = lb + 1$ charging stations into the routes, respectively. Larger values than $lb+1$ are not considered since under normal circumstances, inserting more charging stations means more detours which may bring unnecessary increase to the length of the route. The function 'addStations' will enumerate all the feasible combinations of the positions in the routes to insert n stations, and the best insertion is recorded into Γ^* (lines 9-26).

SE is proposed based on the posteriori knowledge that we obtained from our previous work. The analyses of the final solution obtained by BACO show that a vehicle at most will charge four times during its trip. In most cases, it will only charge once or twice. This is consistent with real life that a battery-electric truck usually can run three hours with a fully charged battery [51]. If a vehicle charges twice, it can already run nine hours. Since a vehicle does not need to charge many times during its trip, enumerating the possible recharging schedules is not super computationally expensive. In addition, in SE, we only consider one charging station between each pair of nodes, which also decreases the algorithm complexity. This setting may miss the best recharging schedule for some routes, but in most cases, it can generate very good recharging schedules. The theoretical and empirical analyses of the complexities of FRVCP and SE are given in the experimental studies to verify the rationality of SE.

Algorithm 4: Simple Enumeration

Input: an electricity-infeasible route $\Gamma \in \mathbf{\Gamma}$ starting and ending at 0, edge set E, customer set V_c, charging station set V_f, maximum battery capacity Q_b, battery consumption rate r_b.
Output: an electricity-feasible route Γ^*.

1 **foreach** $(i,j), i,j \in V_c \cup \{0\} \wedge i \neq j$ **do**
2 find $\theta_{ij} = \theta_{ji} = \arg\min_{\theta \in V_f} d_{i\theta} + d_{\theta j}$;
3 **end**
4 calculate the total length of Γ as d_Γ;
5 $lb = \lceil d_\Gamma \cdot r_b / Q_b \rceil$;
6 $\Gamma' = \Gamma$, $\Gamma^* = \Gamma$, $f_{\Gamma^*} = +\infty$;
7 **for** $sn = lb \rightarrow lb + 1$ **do**
8 call the following recursion function;
9 **Function** `addStations` $(m = 0, n = sn, \Gamma')$ **:**
10 **for** $i = m \rightarrow |\Gamma| - 1 - n$ **do**
11 add $\theta_{\Gamma_i,\Gamma_{i+1}}$ into Γ' between Γ'_j and Γ'_{j+1} where $\Gamma'_j == \Gamma_i \wedge \Gamma'_{j+1} == \Gamma_{i+1}$;
12 denote the distance between this station and the previous station or depot in Γ' as d_p;
13 **if** $d_p \cdot r_b > Q_b$ **then**
14 remove $\theta_{\Gamma_i,\Gamma_{i+1}}$ from Γ';
15 **break**;
16 **end**
17 **if** $n > 1$ **then**
18 `addStations` $(m+1,\ n-1,\ \Gamma')$;
19 **else**
20 **if** Γ' *is electricity-feasible and* $F(\Gamma') < f_{\Gamma^*}$ **then**
21 $\Gamma^* = \Gamma'$, $f_{\Gamma^*} = F(\Gamma')$;
22 **end**
23 **end**
24 remove $\theta_{\Gamma_i,\Gamma_{i+1}}$ from Γ';
25 **end**
26 **End Function**;
27 **end**
28 **return** Γ^*;

V. Experimental Setup

A. Benchmark Test Cases

The benchmark proposed for the IEEE WCCI2020 competition on EC for EVRP [42] is taken to check the performance of CBACO. The test cases in this benchmark are derived from two famous CVRP test sets [2], [52] by evenly distributing charging stations on the map and switching traditional vehicles to EVs. The test cases can be divided into two groups. The first group contains seven relatively small instances in which the number of customers is $21 \leq |V_c| \leq 100$. The second group contains ten relatively large instances in which the number of customers is $100 < |V_c| \leq 1000$. Details of the instances are shown in Table I. Also, four representative cases with different customer distributions are shown in Fig. 4. The maps of the other test cases can be found in [42].

B. Algorithm Settings

The performance of CBACO is compared with BACO, GA, SA, VNS, and ILS. BACO is proposed in our previous work [8]. GA, SA, and VNS are the three winners of the WCCI2020 EVRP competition [39]. ILS is proposed in [19] for EVRP with non-linear charging function that can be easily used to solve CEVRP. Since both codes and papers of GA, SA, and

TABLE I
TEST CASES OF CEVRP BENCHMARK SET.

Name	Customer	Station	Route
E22	21	8	4
E23	22	9	3
E30	29	6	4
E33	32	6	4
E51	50	5	5
E76	75	7	7
E101	100	9	8
X143	142	4	7
X214	213	9	11
X352	351	35	40
X459	458	20	26
X573	572	6	30
X685	684	25	75
X749	748	30	98
X819	818	25	171
X916	915	9	207
X1001	1000	9	43

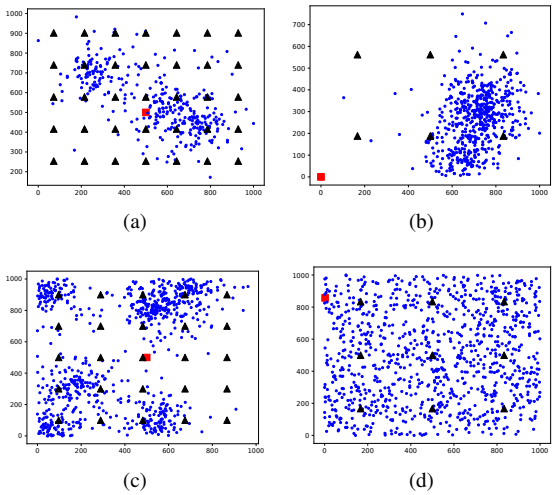


Fig. 4. Representative test cases. The blue dots represent customers. The red square represents the depot. The black triangles represent charging stations. (a) X351, (b) X573, (c) X819, (d) X1001.

VNS are not published yet, the results of these three algorithms are taken from the competition report [39]. CBCAO, BACO, and ILS are implemented by ourselves in C++ and executed on Intel i7-6700 3.40Hz CPU with Arch Linux system. The stopping criterion is set to a fixed execution time following [8] that allows the algorithms to fully converge:

$$ExeTime = \vartheta \cdot \frac{|V_c| + |V_f|}{100}(\text{hr}), \tag{16}$$

where $|V_c|+|V_f|$ is the total number of nodes of the problem. ϑ is set to 1, 2, 3 for E22-E101, X143-X916, and X1001, respectively.

The parameters of CBACO related to the MMAS algorithm follow the conventional setting [48]. The pheromone evaporation speed ρ is set to 0.98. The population size is set to $|V_c| + 1$ which is the number of customers and depot. The weights in 13, α and β, are set to 1 and 2 according to the experimental studies of several ACO algorithms including BACO [8], [48], [53]. Another setting that is not shown in Algorithm 2 and Algorithm 3 is the utilization of neighborhood candidate list. For the test cases with more than 500 customers, a neighborhood candidate list is calculated in advance for each customer. Each time, the next customer is chosen from the candidate list of the last one rather than all remaining customers, unless there is no feasible customers in the list. This is a common strategy that is widely adopted by ACO to accelerate the convergence speed [48], [54]. The size of the candidate list is set to 20. As to the three parameters related to the confidence-based selection method, they are tuned in the following experiment. CBACO with the direct encoding scheme is denoted as CBACO-D. Similarly, CBACO with the indirect encoding scheme is denoted as CBACO-I.

For the following experiments, if there are no special settings, we run the algorithms 30 times on each instance to get the statistical information including the best objective value, the mean objective value, the standard deviation, and the significance test results. Friedman test is adopted to compare multiple algorithms with Holm post-hoc analysis. The significance level is set to 0.05 with correction.

C. Parameter Tuning

There are three parameters related to the confidence-based selection, δ, γ_l, and γ_r. δ and γ_l take charge of selecting sub-solutions to conduct the local search process. γ_r takes charge of selecting sub-solutions to conduct the lower-level optimization.

Generally, larger δ and γ_l will recruit more sub-solutions for local search which may result in a waste of computing resources. On the contrary, smaller δ and γ_l will recruit less sub-solutions for local search, but some promising sub-solutions may be missed. To get an appropriate setting, we test three groups of values, $(\delta = 10, \gamma_l = 1)$, $(\delta = 30, \gamma_l = 1.2)$, and $(\delta = 50, \gamma_l = 1.4)$ on X143 and X214. CBACO without confidence-based selection is taken as the baseline. By setting a fixed iteration number 10^4, we evaluate the settings by two indicators, 1) how many sub-solutions are recruit to conduct local search and 2) the final objective value. The confidence-based selection for lower-level optimization is not used in order to avoid the influence of other factors. The experimental results are shown in Fig. 5 and Table II.

From the results, we can see that the confidence-based selection of local search works well on both X143 and X214 for CBACO-I. Considering both the number of solutions selected and the final objective values, we choose (30, 1.2) for CBACO-I since (10,1) generated significantly worse results on X143. However, for CBACO-D, the confidence-based selection of local search seems useless on X214. We have further tested the selection method for CBACO-D on the other instances and found that when the scale is smaller than 150, it is effective. Otherwise, it has little influence to the algorithm. Thus, we apply confidence-based selection of local search to CBACO-D only when the scale of the test instance is smaller than 150. The reason of the ineffectiveness of the selection method when the scale is large is because the inter-route local search method, i.e. 2opt*, would change the quality of a solution greatly for a large-scale instance, which makes it hard to predict how much improvement it will bring.

TABLE II
PERFORMANCE OF CBACO UNDER DIFFERENT δ AND γ_l VALUES.

		CBACO-D			
		BL	(10,1)	(30,1.2)	(50,1.4)
X143	mean	15953.29	16033.61↑	16003.96↕	15995.04↕
	std	37.17	79.43	81.82	84.07
X214	mean	11731.87	11725.30↕	11732.22↕	11717.49↕
	std	38.68	26.68	40.11	38.81
		CBACO-I			
		BL	(10,1)	(30,1.2)	(50,1.4)
X143	mean	16017.15	16074.09↑	16061.56↕	16042.69↕
	std	37.99	66.67	63.04	33.35
X214	mean	11277.03	11318.40↕	11324.46↕	11273.51↕
	std	69.27	74.76	65.62	69.98

'BL' means baseline that does not apply confidence-based selection. ↕, ↑, and ↓ mean that the performance is equal to, significantly worse than, and significantly better than the baseline according to Wilcoxon rank-sum test with significance level of 0.05, respectively.

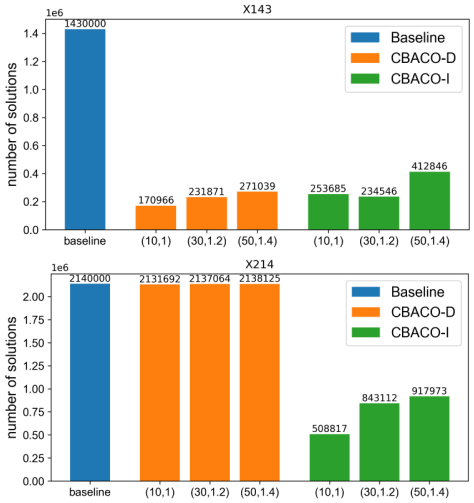


Fig. 5. The number of solutions that were selected to conduct local search.

Meanwhile, smaller γ_r will recruit more sub-solutions to be considered in the lower-level optimization and larger γ_l will recruit less. We also set three values for γ_r, 1.0, 0.8, and 0.6. The experimental results are shown in Table III and Fig. 6.

From the results, we can see that the confidence-based selection method has successfully selected a small number of sub-solutions to conduct the lower-level optimization for both CBACO-D and CBACO-I. Different γ_r values have little

TABLE III
PERFORMANCE OF CBACO UNDER DIFFERENT γ_r VALUES.

		CBACO-D			
		BL	1	0.8	0.6
X143	mean	15953.29	15949.11↕	15949.11↕	15949.11↕
	std	37.17	38.01	38.01	38.01
X214	mean	11731.87	11706.73↕	11706.73↕	11714.71↕
	std	38.68	57.67	57.67	52.49
		CBACO-I			
		BL	1	0.8	0.6
X143	mean	16017.15	16016.00↕	16016.00↕	16012.51↕
	std	37.99	35.22	35.22	34.92
X214	mean	11277.03	11246.33↕	11260.80↕	11260.80↕
	std	69.27	52.16	65.82	65.82

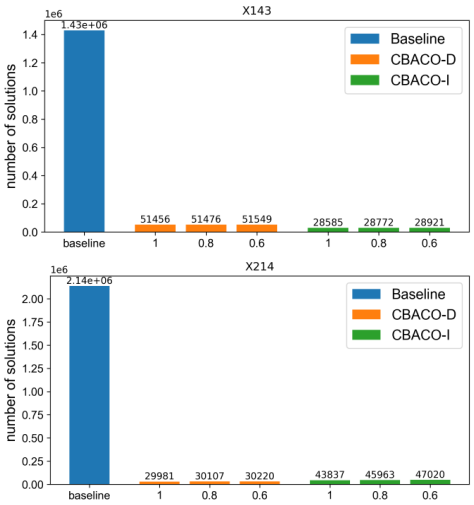


Fig. 6. The number of solutions that were selected to conduct lower-level optimization.

influence to the selection, which means the algorithm is not sensitive to γ_r. For the sake of generality, we use 0.8 in the following experiment in case that some other instances may require less strict threshold.

VI. COMPARISONS AND ANALYSES

A. Comparison of the Objective Values

We first compare these algorithms on 17 test instances in terms of effectiveness. The experimental results are shown in Table IV. The best min and mean objective values are highlighted. Behind the mean objective values of ILS, GA, SA, VNS, and BACO, there are two symbols. ↑, ↓, and ↕ means the algorithm is significantly worse than, significantly better than, and equal to CBACO-D or CBACO-I, respectively. Behind the mean objective values of CBACO-D, there is one symbol representing the Friedman test result between CBACO-D and CBACO-I. 'w/t/l' shows on how many instances the compared algorithm wins, ties, or loses to CBACO-D and CBACO-I. 'BKS' represents the best known solution. 'rank' shows the rank of each algorithm according to the Friedman test.

From the overall rank of the compared algorithms, we can get CBACO-I<BACO<VNS<CBACO-D<SA<GA<ILS (A<B means A is better than B). First, we compare CBACO-D and CBACO-I with other algorithms. Then, they two are compared with each other to show the advantages and disadvantages of each encoding scheme.

- Setting 150 as a demarcation line, we can find that CBACO-D is extremely effective and stable on E22 to X143. Only on E33 it was beaten by VNS. On the other instances with less than 150 customers, CBACO-D either has competitive performance or significantly outperformed the others. On E101 and X143, it has even updated the best known solutions. Also, its performance is very stable that the standard deviations of the results are usually very small. On the instances with more than 150 customers, the performance of CBACO-D is barely satisfactory. Its performance is similar to GA and SA on

TABLE IV
COMPARISON BETWEEN CBACO AND THE STATE-OF-THE-ART ALGORITHMS ON 17 BENCHMARK INSTANCES IN TERMS OF OBJECTIVE VALUE.

case	index	BKS	ILS	GA	SA	VNS	BACO	CBACO-D	CBACO-I
E22	best	384.67	**384.67**	**384.67**	**384.67**	**384.67**	**384.67**	**384.67**	**384.67**
	mean		385.69↕↕	**384.67**↕↕	**384.67**↕↕	**384.67**↕↕	**384.67**↕↕	**384.67**↕	**384.67**
	std.		2.11	0.00	0.00	0.00	0.00	0.00	0.00
E23	best	571.94	590.35	**571.94**	**571.94**	**571.94**	**571.94**	**571.94**	**571.94**
	mean		592.05↑↑	**571.94**↕↕	**571.94**↕↕	**571.94**↕↕	**571.94**↕↕	**571.94**↕	**571.94**
	std.		1.04	0.00	0.00	0.00	0.00	0.00	0.00
E30	best	509.47	**509.47**	**509.47**	**509.47**	**509.47**	**509.47**	**509.47**	**509.47**
	mean		**509.47**↕↕	**509.47**↕↕	**509.47**↕↕	**509.47**↕↕	**509.47**↕↕	**509.47**↕	**509.47**
	std.		0.00	0.00	0.00	0.00	0.00	0.00	0.00
E33	best	840.14	840.57	844.25	840.57	**840.14**	840.57	840.57	840.57
	mean		844.07↑↕	845.62↑↑	854.07↑↑	**840.43**↓↓	842.30↑↕	840.57↓	843.59
	std.		7.78	0.92	12.80	1.18	1.42	0.00	1.29
E51	best	529.90	**529.90**	**529.90**	533.66	**529.90**	**529.90**	**529.90**	**529.90**
	mean		539.03↑↑	542.08↑↑	533.66↑↑	543.26↑↑	**529.90**↕↕	**529.90**↕	**529.90**
	std.		6.92	8.57	0.00	3.52	0.00	0.00	0.00
E76	best	692.64	694.64	697.27	701.03	**692.64**	**692.64**	**692.64**	**692.64**
	mean		704.24↑↑	717.30↑↑	712.17↑↑	697.89↑↑	**692.85**↕↕	692.88↕	694.58
	std.		7.15	9.58	5.78	3.09	0.81	0.92	3.33
E101	best	839.29	841.02	852.69	845.84	839.29	840.25	**838.84**	840.69
	mean		851.62↑↑	872.69↑↑	852.48↑↑	853.34↑↑	845.95↑↕	**840.56**↓	845.95
	std.		6.93	9.58	3.44	4.73	4.58	0.42	4.03
X143	best	15901.23	16058.29	16488.60	16610.37	16028.05	15901.23	**15884.58**	15895.52
	mean		16318.57↑↑	16911.50↑↑	17188.90↑↑	16459.31↑↑	16031.46↑↕	**15930.76**↓	15993.34
	std.		160.07	282.30	170.44	242.59	262.47	31.15	47.27
X214	best	11133.14	11323.56	11762.07	11404.44	11323.56	11133.14	11579.18	**11091.37**
	mean		11537.58↓↑	12007.06↕↑	11680.35↕↑	11482.20↓↑	**11219.70**↓↕	11706.55↑	11245.18
	std.		72.55	156.69	116.47	76.14	46.25	38.73	66.09
X352	best	26478.34	27947.89	28008.09	27222.96	27064.88	26478.34	27800.71	**26456.95**
	mean		28364.41↑↑	28336.07↕↑	27498.03↕↑	27217.77↓↑	**26593.18**↓↕	27953.59↑	26637.80
	std.		142.04	205.29	155.62	86.20	72.86	56.33	91.24
X459	best	24763.93	26511.28	26048.21	NA	25370.80	**24763.93**	25648.48	24776.44
	mean		26726.69↑↑	26345.12↕↑	25809.47↕↑	25582.27↓↑	24916.60↓↕	25884.82↑	**24911.98**
	std.		126.93	185.14	157.97	106.89	94.08	78.66	72.95
X573	best	51929.24	53102.46	54189.62	**51929.24**	52181.51	53822.87	52706.59	53818.40
	mean		53507.46↑↓	55327.62↑↑	52793.66↕↓	**52548.09**↓↓	54567.15↑↕	52853.78↓	54264.16
	std.		275.76	548.05	577.24	278.85	231.05	53.12	197.71
X685	best	70834.88	74409.65	73925.56	72549.90	71345.40	**70834.88**	74816.89	70943.23
	mean		75087.58↕↑	74508.03↓↑	73124.98↓↑	71770.57↓↑	71440.57↓↕	75144.84↑	**71339.17**
	std.		259.60	409.43	320.07	197.08	281.78	191.72	297.11
X749	best	80299.76	84298.43	84034.73	81392.78	81002.01	80299.76	82721.82	**80042.38**
	mean		84860.28↑↑	84759.79↑↑	81848.13↓↑	81327.39↓↑	80694.54↓↕	83035.49↑	**80571.38**
	std.		287.26	376.10	275.26	176.19	223.91	147.63	222.72
X819	best	164289.95	168651.19	170965.68	165069.77	164289.95	164720.80	165219.00	**163751.23**
	mean		169837.06↑↑	172410.12↑↑	165895.78↕↑	164926.41↓↕	165565.79↕↑	165650.45↑	**164826.42**
	std.		483.35	568.58	403.70	318.62	401.02	208.70	599.22
X916	best	341649.91	348733.86	357391.57	342796.88	341649.91	342993.01	342436.63	**341369.36**
	mean		350822.41↑↑	360269.94↑↑	343533.85↕↕	**342460.70**↓↕	344999.95↑↑	343334.11↕	343127.47
	std.		1177.08	229.19	556.98	510.66	905.72	357.84	877.97
X1001	best	76297.09	79493.37	78832.90	78053.86	77476.36	76297.09	79046.58	**75666.26**
	mean		79928.29↕↑	79163.34↕↑	NA↓↑	77920.52↓↑	77434.33↓↑	79293.37↑	**76405.53**
	std.		265.91	NA	306.27	234.73	719.8671	123.11	337.78
w/t/l	vs D		1/4/12	1/7/9	3/9/5	10/3/4	6/6/5		
	vs I		1/3/13	0/3/14	1/4/12	2/5/10	0/14/3	4/6/7	
rank			5.39	5.84	4.43	3.24	2.84	3.69	2.56

most of the large-scale instances, which is overall worse than VNS and BACO. The only exception is X916 where it has similar performance with CBACO-I.

- CBACO-I inherits from BACO the indirect encoding scheme. Because of the confidence-based selection method, it is more efficient than BACO. That is why CBACO-I has given better mean objective values on the last five large-scale instances than BACO. Because of the improvement of the efficiency, CBACO-I has achieved the same level of performance as VNS on X819 and X916 but did not surpass VNS. Meanwhile, CBACO-I did not perform well on X573 either just like BACO. The reason that CBACO-I did not surpass VNS on these three instances is the pheromone mismatching problem that we have mentioned before, but there are two different factors aggravating the influence of the mismatching. For X573, most customers are far away from the depot, as shown in Fig. 4(c), which makes the edges between customers and the depot occupy a big proportion of the total traveling distance of the routes. For X819 and X916, the pheromone mismatching comes from the distribution of customer demand. Fig. 7 shows the normalized distributions $D(c_i/Q_c)$ of customer demand of the large test cases. As we can see from Fig. 7, in X819 and X916, the

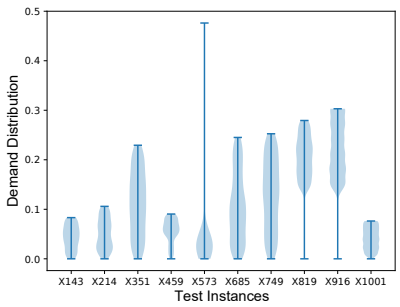


Fig. 7. The normalized distributions $D(c_i/Q_c)$ of customer demand of the ten large test cases.

average demand of a customer is very high so that each EV can only serve a small number of customers. There will be a large number of routes in the solutions of X819 and X916, which brings a lot of edges linked between the depot and the customers. Therefore, the pheromone mismatching problem is worsened. For the other test cases, since the average demand is low, an EV can serve many customers before returning to the depot. There are only a few of the edges linked between the depot and the customers. Thus, the pheromone mismatching problem does not have a major impact to the final performance.

The comparison between CBACO-D and CBACO-I gives us a guideline of when to use each of them. In general, CBACO-D shows dominant performance on instances with less than 150 customers, and CBACO-I performs better on the other large-scale instances.

The reason that CBACO-D performs better on small-scale instances but worse on large-scale instances is the dependence on local search methods, especially the inter-route local search method 2-opt*. For small-scale instances, the number of local optima is small and the quality of the local optimal solutions is usually not bad. The applied local search methods are capable enough to lead the initial solutions to the global optimum or good local optima. However, when the scale of the instance increases, the number of local optima increases dramatically and most of the local optimal solutions are poor. Under such circumstances, the local search methods do not have the ability to lead the solutions to good local optima. Meanwhile, since many solutions generated by ACO are guided to poor local optima by the local search methods, the algorithm encounters the premature convergence problem that the pheromone matrix updated by the poor local optimal solutions makes the algorithm stuck there. Compared with CBACO-D, CBACO-I is less dependent on inter-route local search methods since the splitting algorithm can guarantee the best partition of the giant tour. The solutions generated by CBACO-I during the evolution process are more diverse. The high diversity can protect the algorithm from premature convergence. Thus, CBACO-I performs well on large-scale instances. However, the high diversity sacrifices the exploitation ability a little which makes CBACO-I less effective and less stable than CBACO-D on small-scale instances.

B. Comparison of Convergence

Then, the convergence curves of ILS, BACO, CBACO-D, and CBACO-I on X143-X1001 are drawn in Fig. 8 to check the algorithms' converging speeds. The convergence curves of GA, SA, and VNS are not available because both their codes and papers have not been published yet.

From the figures, we can get the following observations:

1) There is not a universal rank of the converging speed that can fit all instances. Generally, the converging speed of ILS is slower than the others except for X573 on which ILS surpasses BACO and CBACO-I and X685 on which ILS surpasses CBACO-D. ILS usually has a fast converging speed in the early stage. Then, in the middle and later stages, it maintains a steady but very slow converging speed. This is because of the adopted perturbation method, i.e. double-bridge, that only changes four links each time. For problems with more than one hundred customers, changing four links each time is inefficient for exploration.
2) Among the three ACO-based algorithms, CBACO-D has the quickest converging speed that it usually takes only one fifth of the total execution time to converge. The reason of the quick converging speed of CBACO-D is its dependence on local search algorithms, especially the inter-route local search method. Since all the good solutions generated in the early stage are pushed into local optima by local search and the pheromone matrix is updated according to these solutions, the exploration ability of CBACO-D degenerates rapidly. On most of the instances, the quick converging speed of CBACO-D leads to premature convergence, but sometimes, if the local search methods fits the problem well, CBACO-D can get very good results in a short time such as on X143, X819, and X916. The superiority of CBACO-D on small-sale instances like X143 has been explained previously. On X819 and X916, Fig. 8(h) and Fig. 8(i) show that CBACO-D establishes its advantage in the very beginning because of the local search methods. From Table I, we can see that X819 and X916 share a characteristic that there are a lot of routes in the problem and each route does not have many customers. For X819, the average number of customers per route is 4.78 and for X916, it is 4.42. Obviously, under the circumstances, the inter-route local search will be very effective.
3) Comparing BACO and CBACO-I that both use indirect encoding scheme, we can see that the confidence-based selection method has successfully accelerated the converging speed of CBACO-I without interfering the effectiveness of the algorithm. On X143-X459, there is no significant difference between BACO and CBACO-I in terms of both converging speed and final result. On X573-X1001, CBACO-I showed obviously faster converging speed than BACO, and due to the fast converging speed, CBACO-I also got obviously better final solution on the last three instances. On X143-X459, since the scale of the problem is not very large, the local search and the lower-level optimization processes do not occupy a

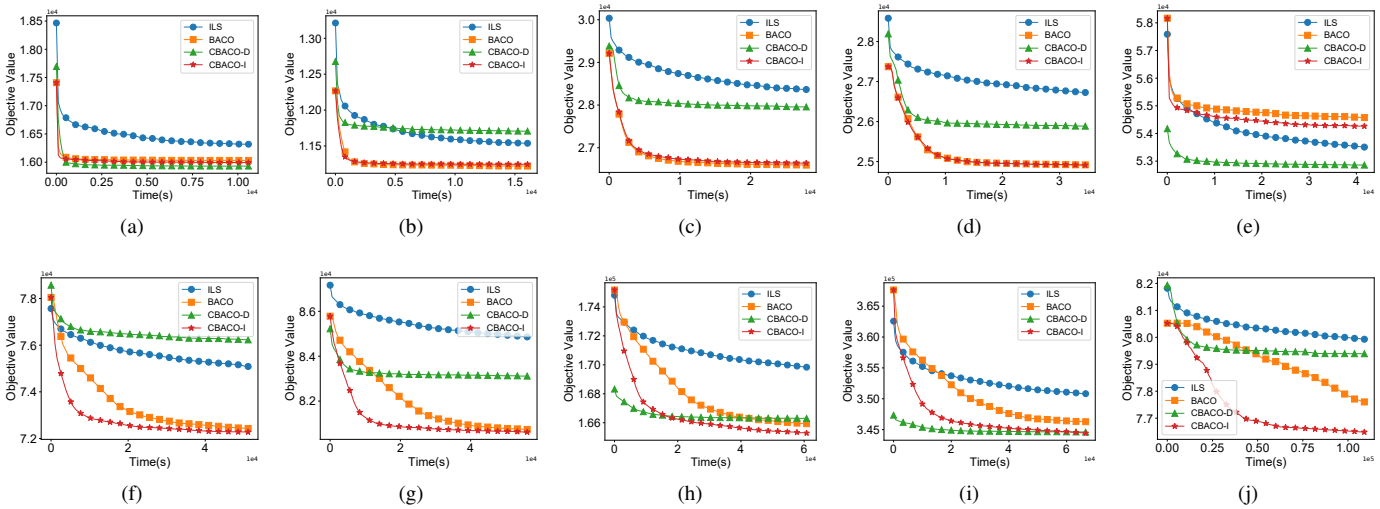


Fig. 8. Convergence curves of ILS, BACO, CBACO-D, and CBACO-I. (a)-(j) represent X143-X1001.

big proportion of the total execution time. Meanwhile, although we have set a relatively loose threshold for the confidence-based selection, it may still miss a small number of iteration best solutions. Under the influence of these two factors, on X143-X459, the confidence-based selection method does not improve the converging speed significantly. By contrast, the local search and the lower-level optimization processes on larger instances X573-X1001 consume a large amount of the execution time, getting rid of the bad sub-solutions can save near half of the execution time to reach the same level of performance. Thus, the larger the problem is, the better the confidence-based selection performs.

Overall, we have verified the efficiency of the confidence-based selection method. Meanwhile, although CBACO-I showed comprehensively better capability than CBACO-D, CBACO-D showed good capability to handle the small-scale instances with less than 150 customers and the instances that have many routes with small number of customers per route.

C. Comparison between Different Selection Strategies

The comparison between BACO and CBACO-I has verified that the confidence-based selection strategy is more efficient than selecting all sub-solutions to do local search and lower-level optimization. As we have stated in Section IV.A, there is another selection strategy that only selects the iteration best solutions to conduct local search and lower-level optimization. Here, we compare our confidence-base selection method with this simple strategy to verify that only selecting one is not enough for ACO to solve CEVRP. BACO is re-implemented with this strategy that only selects the iteration best sub-solution in terms of f', i.e. the total length of the sub-solution before local search and lower-level optimization, and it is denoted as BACO-ONE. It is compared with CBACO-I and BACO since they all use indirect encoding scheme. We choose X459 and X685 to make the comparison. On X459, although the confidence-based selection did not improve the efficiency, it did not affect the final performance as Fig. 8(d) showing.

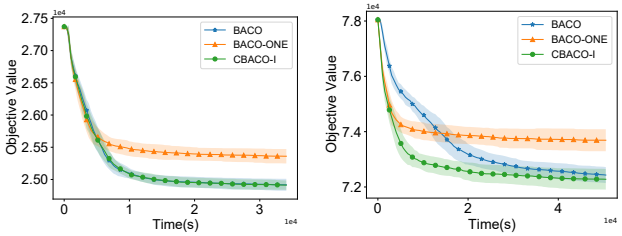


Fig. 9. Convergence curves with standard deviation of the three selection strategies: selecting all, selecting one, and confidence-based selection. (a) X459, (b) X685.

On X685, the confidence-based selection method successfully accelerated the converging speed of CBACO-I as Fig. 8(f) showing. The comparison results are shown in Fig. 9.

From the comparison, we can clearly see that the strategy of selecting one is not good. On X459, BACO-ONE did not accelerate the converging speed of BACO and it incurred the premature convergence problem that the final performance was much worse than BACO and CBACO-I. On X685, although BACO-ONE indeed had a faster converging speed than BACO in the early stage, it converged too early that the final performance was still much worse than BACO and CBACO-I. Also, it is worth noting that even the converging speed BACO-ONE is faster than BACO in the early stage on X685, its converging speed is still slightly slower than CBACO-I, which perfectly demonstrated the efficiency and effectiveness of the proposed confidence-based selection method.

D. Investigation of SE

The effectiveness and efficiency of SE is investigated through comparison with the other heuristics including, forward heuristic (FH), greedy heuristic (GH), removal heuristic (RH), and the restricted enumeration heuristic (RE). FH finds a nearest charging station for an EV whenever the EV cannot reach its next customer. GH inserts charging stations into a route to minimize the electricity deficit of the route [19]. RH

TABLE V
COMPARISON OF THE OBJECTIVE VALUE AND THE EXECUTION TIME (US) AMONG SE, FH, GH, RH, AND RE

case	index	SE	FH	GH	RH	RE
E22	obj.	384.68	394.79	384.68	384.68	384.68
	time	44	20	1390	100	131
E23	obj.	571.95	589.58	599.23	571.95	571.95
	time	48	18	3791	128	237
E30	obj.	509.47	531.26	509.47	509.47	509.47
	time	37	15	629	155	84
E33	obj.	840.57	876.81	840.57	840.57	840.57
	time	71	18	1484	170	176
E51	obj.	529.90	567.23	529.90	529.90	529.90
	time	59	10	1900	107	160
E76	obj.	692.64	726.48	692.64	692.64	692.64
	time	46	9	1524	108	142
E101	obj.	838.84	893.84	842.73	838.84	838.84
	time	65	9	2015	128	194
X143	obj.	15884.58	16981.99	15957.38	15884.58	15884.58
	time	114	6	1566	234	298
X214	obj.	11091.37	11500.43	11091.37	11091.37	11091.37
	time	108	11	2582	327	220
X352	obj.	26465.91	26743.32	26610.69	26465.91	26456.95
	time	125	35	16811	410	739
X459	obj.	24763.93	24928.12	24923.01	24763.93	24763.93
	time	180	28	12358	821	515
X573	obj.	52662.37	53127.51	52808.50	52662.37	52662.37
	time	398	27	6058	991	1003
X685	obj.	71817.18	71772.45	70952.02	71817.18	70834.89
	time	339	76	23837	853	890
X749	obj.	80059.61	81277.84	80443.05	80063.24	80042.38
	time	269	87	26595	798	1044
X819	obj.	163999.08	164447.65	163919.11	163999.08	163751.23
	time	253	111	17547	708	977
X916	obj.	342475.75	342408.37	341799.17	342475.75	341369.36
	time	311	156	9088	665	1012
X1001	obj.	75667.82	76752.63	76144.27	75667.82	75666.26
	time	761	44	16739	1774	1974

inserts charging stations into every pair of successive nodes of a route and then removes redundant ones [8]. RE is a complete enumeration method that considers more charging stations than SE [8]. To compare these heuristic algorithms, we remove the charging stations in the best solutions that we have ever found and apply these different heuristics to them respectively. The objective values and execution time are shown in Table V.

From the perspective of the effectiveness (objective value), the general rank of these heuristics is RE<SE<RH<GH<FH (the lower the better), although there are some special situations such as X819 and X916 on which FH or GH has shown better results than SE and RH. From the perspective of the efficiency (execution time), the rank of these heuristics is FH<SE<RH<RE<GH (the lower the better).

Regarding both effectiveness and efficiency, SE is in the second place which means that it is a very good choice. On small-scale problems with less than 300 customers, it can guarantee to find the optimal recharging schedule as same as RE can do. On large-scale problems, it can generate a good recharging schedule very quickly.

E. Complexity of FRVCP and SE

After getting the empirical analysis of SE, we study its theoretical complexity to gain an insight into FRVCP. Assume we have a route that contains n_r nodes including customers and a depot, and m charging stations are available. Setting a maximum number of recharging in this route to q, we know that there will be totally $\mathrm{C}_{n_r}^{q} \cdot m^q$ different recharging schedules. If $q \approx n_r$ meaning that the vehicle needs to recharge between every pair of successive nodes in the route, the number of candidate recharging schedules is m^{n_r} which grows exponentially with n_r. Thus, the complexity of the lower-level optimization of CEVRP, i.e. FRVCP, is very high theoretically. As to the SE heuristic, since only the best charging station that brings minimum extra cost is considered between each pair of nodes, $\mathrm{C}_{n_r}^{q}$ recharging schedules will be considered.

However, is the practical situation on the test instances as horrific as the theoretical analysis? In this section, we make an analysis to check the real values of the two variables n_r (the number of customers and depot in one route) and q (the number of recharging needed in one route). m is already shown in Table I. The best solutions that we have gotten from the experiment are taken to analyze. The results are shown in Table VI.

From Table VI, we can see that range of the maximum number of customers in one route is [7, 41], which is in line with real-world supply chains of supermarkets or grocery stores [55]. It means that the maximum n_r of the test instances is 41. Then, we can see that the maximum number of recharging of one route is three on X352. Thus, q is actually at most three for the test instances.

Since q is far less than n_r that can be considered as a constant value, $\mathrm{C}_{n_r}^{q} \cdot m^q \approx (n_r \cdot m)^q$, which becomes polynomial. Thus, although FRVCP is theoretically non-deterministic polynomial hard (NP-hard) [56], its real complexity in CEVRP is quite close to polynomial rather than NP. Accordingly, the time complexity of the SE heuristic is approximately equal to $\mathrm{O}({n_r}^q)$ that is also polynomial.

The antagonism between the complexity of FRVCP and the quality of the CEVRP solution is the main reason that the FRVCP in CEVRP is not as horrific as theoretical analysis. Considering a solution of CEVRP that requires the vehicles to recharge between each pair of successive customers, we intuitively know that this solution is poor since with a large probability, it has scheduled some customers that are far away from each other to the same vehicle. Thus, although the complexity of FRVCP is high under this specific solution, it is unnecessary to consider this solution. For a good solution that does not need to recharge the vehicles many times, we need to optimize the FRVCP under it but the complexity of FRVCP becomes lower.

These analyses verify the rationality of using SE to solve FRVCP in CEVRP. However, it is worth noting that FRVCP is just a lower-level sub-problem in CEVRP. For a bi-level optimization problem like CEVRP, even a polynomial complex sub-problem can enlarge the search space dramatically since the original problem itself is NP-hard. Also, we have arbitrarily eliminated many possible recharging schedules in SE to make it efficient. Thus, on the large-scale instances, it cannot guarantee to give the optimal recharging schedule.

TABLE VI
THE NUMBER OF CUSTOMERS n AND THE NUMBER OF RECHARGING q IN ONE ROUTE.

	n_r		q	
inst.	mean	max	mean	max
E22	6.25	7	0.75	1
E23	8.33	15	1	2
E30	8.25	14	0.25	1
E33	9	15	0.5	1
E51	11	13	1	1
E76	11.71	17	0.71	1
E101	13.5	17	1	2
X143	21.29	26	0.86	2
X214	18.75	23	0.42	1
X352	9.54	15	0.59	3
X459	17.96	41	0.52	2
X573	20.07	38	0.57	2
X685	9.77	21	0.55	2
X749	8.48	13	0.62	2
X819	5.57	7	0.55	2
X916	5.28	7	0.61	2
X1001	23.73	29	0.73	2

VII. CONCLUSIONS AND FUTURE WORKS

The goal of this paper is to pursue better performance of ACO in handling CEVRP. This goal has been successfully achieved by proposing a new algorithm, i.e. CBACO. Through applying the confidence-based selection strategy and different encoding schemes, CBACO has shown competitive results compared with the state-of-the-art algorithms and updated eight of best known solutions out of the seventeen benchmark instances. The proposed confidence-based selection method has greatly improved the efficiency of the algorithm without compromising the effectiveness. Meanwhile, according to the experiments, we had achieved a guideline. 1) When the scale of the instance is smaller than 150 or the number of customer per route is smaller than seven, CBACO-D is an efficient and effective choice. 2) When the scale of the instance is larger than 150, we recommend CBACO-I because of its high exploration ability. 3) The only exception is the problem with large scale and the customers gather around a place far away from the depot where CBACO is not a good choice.

Regarding the future research on EVRP, there are mainly two directions to follow. 1) From the problem perspective, besides CEVRP, there are still many other EVRP variants, such as EVRPTW, EVRP with nonlinear charging function, and EVRPPD. Exploring the application of the bi-level population-based meta-heuristic algorithms on these problems is a promising direction. 2) From the algorithmic perspective, the confidence-based selection method proposed in this paper may only suit a few algorithms that rely on the best solutions, either iteration best or overall best, to evolve. For some algorithms like GA and particle swarm optimization, it may not be suitable. Meanwhile, adaptive parameter tuning methods should be designed to control the confidence-based selection process automatically as part of the future work. Also, the proposed lower-level optimization technique, i.e. SE, is hard to be directly applied to other EVRP variants. Thus, more selection strategies and corresponding heuristics should be designed to meet the requirement of different problems and algorithms in the future.

REFERENCES

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

[2] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *Eur. J. Oper. Res.*, vol. 257, no. 3, pp. 845–858, 2017.

[3] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.

[4] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems," *J. für Betriebswirtschaft*, vol. 58, no. 1, pp. 21–51, 2008.

[5] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transport. Sci.*, vol. 40, no. 4, pp. 455–472, 2006.

[6] C. Lin, K. L. Choy, G. T. Ho, S. H. Chung, and H. Lam, "Survey of green vehicle routing problem: past and future trends," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1118–1138, 2014.

[7] S. N. Kumar and R. Panneerselvam, "A survey on the vehicle routing problem and its variants," 2012.

[8] Y.-H. Jia, Y. Mei, and M. Zhang, "A bilevel ant colony optimization algorithm for capacitated electric vehicle routing problem," *IEEE Trans. Cybern.*, 2021.

[9] Z. P. Cano, D. Banham, S. Ye, A. Hintennach, J. Lu, M. Fowler, and Z. Chen, "Batteries and fuel cells for emerging electric vehicle markets," *Nat. Energy*, vol. 3, no. 4, pp. 279–289, 2018.

[10] F. Knobloch, S. V. Hanssen, A. Lam, H. Pollitt, P. Salas, U. Chewpreecha, M. A. Huijbregts, and J.-F. Mercure, "Net emission reductions from electric cars and heat pumps in 59 world regions over time," *Nat. Sustain.*, pp. 1–11, 2020.

[11] D. Shicong, "Jd.com will introduce 1,000 logistics nevs in over 10 chinese cities," [Online], 2017, https://www.yicaiglobal.com/news/jdcom-will-introduce-1000-logistics-nevs-in-over-10-chinese-cities.

[12] J. Bi, Y. Wang, Q. Sai, and C. Ding, "Estimating remaining driving range of battery electric vehicles based on real-world data: A case study of beijing, china," *Energy*, vol. 169, pp. 833–843, 2019.

[13] T. Erdelić and T. Carić, "A survey on the electric vehicle routing problem: variants and solution approaches," *J. Adv. Transport.*, vol. 2019, 2019.

[14] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle-routing problem with time windows and recharging stations," *Transport. Sci.*, vol. 48, no. 4, pp. 500–520, 2014.

[15] T. Chen, B. Zhang, H. Pourbabak, A. Kavousi-Fard, and W. Su, "Optimal routing and charging of an electric vehicle fleet for high-efficiency dynamic transit systems," *IEEE Trans. on Smart Grid*, vol. 9, no. 4, pp. 3563–3572, 2016.

[16] I. I. Cplex, "V12. 1: User's manual for cplex," *IBM*, vol. 46, no. 53, p. 157, 2009.

[17] J. Lin, W. Zhou, and O. Wolfson, "Electric vehicle routing problem," *Transport. Res. Procedia*, vol. 12, no. Supplement C, pp. 508–521, 2016.

[18] J. P. Vielma, "Mixed integer linear programming formulation techniques," *Siam Review*, vol. 57, no. 1, pp. 3–57, 2015.

[19] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas, "The electric vehicle routing problem with nonlinear charging function," *Transport. Res. B-Meth*, vol. 103, pp. 87–110, 2017.

[20] S. Zhang, Y. Gajpal, S. Appadoo, and M. Abdulkader, "Electric vehicle routing problem with recharging stations for minimizing energy consumption," *Int. J. Prod. Econ.*, vol. 203, pp. 404–413, 2018.

[21] M. Keskin and B. Çatay, "A matheuristic method for the electric vehicle routing problem with time windows and fast chargers," *Comput. Oper. Res.*, vol. 100, pp. 172–188, 2018.

[22] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.

[23] Z. Guo, Y. Li, X. Jiang, and S. Gao, "The electric vehicle routing problem with time windows using genetic algorithm," in *Proc. IAEAC*. IEEE, 2017, pp. 635–639.

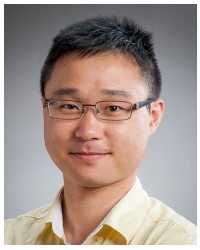
[24] S. Shao, W. Guan, B. Ran, Z. He, and J. Bi, "Electric vehicle routing problem with charging time and variable travel time," *Math. Probl. Eng.*, vol. 2017, 2017.

[25] B. Yu, Z.-Z. Yang, and B. Yao, "An improved ant colony optimization for vehicle routing problem," *Eur. J. Oper. Res.*, vol. 196, no. 1, pp. 171–176, 2009.

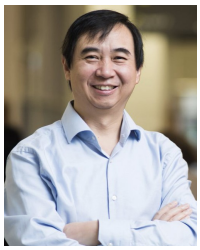
[26] J. E. Bell and P. R. McMullen, "Ant colony optimization techniques for the vehicle routing problem," *Adv. Eng. Inform.*, vol. 18, no. 1, pp. 41–48, 2004.

[27] R. Montemanni, L. M. Gambardella, A. E. Rizzoli, and A. V. Donati, "Ant colony system for a dynamic vehicle routing problem," *J. Comb. Optim.*, vol. 10, no. 4, pp. 327–343, 2005.
[28] Y.-H. Jia, W.-N. Chen, H. Yuan, T. Gu, H. Zhang, Y. Gao, and J. Zhang, "An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 1, pp. 634–649, 2021.
[29] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Comput. Oper. Res.*, vol. 31, no. 12, pp. 1985–2002, 2004.
[30] C. Prins, P. Lacomme, and C. Prodhon, "Order-first split-second methods for vehicle routing problems: A review," *Transport. Res. C-Emer.*, vol. 40, pp. 179–200, 2014.
[31] Y. Xiao, X. Zuo, I. Kaku, S. Zhou, and X. Pan, "Development of energy consumption optimization model for the electric vehicle routing problem with time windows," *J. Clean. Prod.*, vol. 225, pp. 647–663, 2019.
[32] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM, 2002.
[33] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*. Springer, 1998, pp. 2093–2229.
[34] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[35] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
[36] A. Afroditi, M. Boile, S. Theofanis, E. Sdoukopoulos, and D. Margaritis, "Electric vehicle routing problem with industry constraints: trends and insights for future research," *Transport. Res. Procedia*, vol. 3, pp. 452–459, 2014.
[37] M. Bruglieri, F. Pezzella, O. Pisacane, S. Suraci *et al.*, "A variable neighborhood search branching for the electric vehicle routing problem with time windows." *Electron. Notes in Discrete Math.*, vol. 47, no. 15, pp. 221–228, 2015.
[38] M. Bruglieri, S. Mancini, F. Pezzella, O. Pisacane, and S. Suraci, "A three-phase matheuristic for the time-effective electric vehicle routing problem with partial recharges," *Electron. Notes in Discrete Math.*, vol. 58, pp. 95–102, 2017.
[39] M. Mavrovouniotis, "Cec-12 competition on electric vehicle routing problem," [Online], 2020, https://mavrovouniotis.github.io/EVRPcompetition2020/.
[40] M. Keskin and B. Çatay, "Partial recharge strategies for the electric vehicle routing problem with time windows," *Transpor. Res. C-Emer*, vol. 65, pp. 111–127, 2016.
[41] Ç. Koç, O. Jabali, J. E. Mendoza, and G. Laporte, "The electric vehicle routing problem with shared charging stations," *Int. Trans. Oper. Res.*, vol. 26, no. 4, pp. 1211–1243, 2019.
[42] M. Mavrovouniotis, C. Menelaou, S. Timotheou, C. Panayiotou, G. Ellinas, and M. Polycarpou, "Benchmark set for the ieee wcci-2020 competition on evolutionary computation for the electric vehicle routing problem," 2020.
[43] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm Evol. Comput.*, vol. 2, pp. 1–14, 2012.
[44] W.-N. Chen and J. Zhang, "Ant colony optimization for software project scheduling and staffing with an event-based scheduler," *IEEE Trans. Software Eng.*, vol. 39, no. 1, pp. 1–17, 2012.
[45] G. Gutin and D. Karapetyan, "A memetic algorithm for the generalized traveling salesman problem," *Nat. Comput.*, vol. 9, no. 1, pp. 47–60, 2010.
[46] D. Molina, F. Herrera, and M. Lozano, "Adaptive local search parameters for real-coded memetic algorithms," in *Proc. CEC*, vol. 1. IEEE, 2005, pp. 888–895.
[47] E. Kieffer, G. Danoy, M. R. Brust, P. Bouvry, and A. Nagih, "Tackling large-scale and combinatorial bi-level problems with a genetic programming hyper-heuristic," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 44–56, 2019.
[48] T. Stützle and H. H. Hoos, "Max–min ant system," *Future Gener. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.
[49] J.-Y. Potvin, T. Kervahut, B.-L. Garcia, and J.-M. Rousseau, "The vehicle routing problem with time windows part i: tabu search," *INFORMS J. Comput.*, vol. 8, no. 2, pp. 158–164, 1996.
[50] P. Hansen and N. Mladenović, "First vs. best improvement: An empirical study," *Discrete Appl. Math.*, vol. 154, no. 5, pp. 802–817, 2006.
[51] I. B. Hovi, D. R. Pinchasik, E. Figenbaum, and R. J. Thorne, "Experiences from battery-electric truck users in norway," *World Electr. Vehicle J.*, vol. 11, no. 1, p. 5, 2020.
[52] N. Christofides and S. Eilon, "An algorithm for the vehicle-dispatching problem," *J. Oper. Res. Soc.*, vol. 20, no. 3, pp. 309–318, 1969.
[53] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man Cybern. B*, vol. 26, no. 1, pp. 29–41, 1996.
[54] 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, 1997.
[55] J. Fernie and A. McKinnon, "The grocery supply chain in the uk: improving efficiency in the logistics network," *Int. Rev. Retail Distrib. Consumer Res.*, vol. 13, no. 2, pp. 161–174, 2003.
[56] N. D. Kullman, A. Froger, J. E. Mendoza, and J. C. Goodson, "frvcpy: An open-source solver for the fixed route vehicle charging problem," *INFORMS J. Comput.*, 2021.

Ya-Hui Jia (M'20) received the B.Eng. and Eng.D. degrees from Sun Yat-sen University, China, in 2013 and 2019, respectively. He is currently an Assistant Professor at the School of Future Technology, South China University of Technology, Guangzhou, China. His research interests include evolutionary computation, combinatorial optimization, software engineering, cloud computing, and intelligent transportation.



Yi Mei (M'09-SM'18) received the BSc and PhD degrees from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively. He is currently a Senior Lecturer at the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His research interests include evolutionary scheduling and combinatorial optimisation, machine learning, genetic programming, and hyper-heuristics. He has over 100 fully referred publications, including the top journals in EC and Operations Research such as IEEE TEVC, IEEE TCYB, and European Journal of Operational Research. He serves as a Vice-Chair of the IEEE CIS Emergent Technologies Technical Committee, and a member of Intelligent Systems Applications Technical Committee. He is an Editorial Board Member/Associate Editor of three International Journals, and a guest editor of a special issue of the Genetic Programming Evolvable Machine journal. He serves as a reviewer of over 30 international journals.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively. He is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. Prof. Zhang is a Fellow of Royal Society of New Zealand, a Fellow of IEEE, and an IEEE Distinguished Lecturer. He was the chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, the chair for the IEEE CIS Emergent Technologies Technical Committee, the chair of Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a vice-chair of the Task Force on Evolutionary Computer Vision and Image Processing, and the founding chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a committee member of the IEEE NZ Central Section.