A Hybrid Algorithm Based on Ant Colony Optimization and Differential Evolution for Vehicle Routing Problem

Hongbo Li, Xiaoxia Zhang, Shuai Fu and Yinyin Hu

***Abstract*—The vehicle problem (VRP) is a typical optimization problem in logistics and transportation. The objective function is to find the shortest route distances visited by all vehicles originating from a central deport to travel customers, and the sum of deliveries of each vehicle should meet the capacity constraint. This problem belongs to NP hard problems, so it is not easy to resolve it with common methods. Ant colony optimization (ACO) has shown prominent performance for many practical applications. However, it is inclined to premature convergence. The paper offers a hybrid ACO&DE algorithm, which hybridizes ant colony optimization (ACO) with differential evolution (DE) for the VRP. The main feature of the ACO&DE can make full use of advantages of the ACO and DE algorithm to make up for its own weakness, i.e., the ACO has fast construction mechanism, and the DE can extend the search scope of the ACO. Moreover, to make the DE suitable for solving the VRP, both strategies of mutation operator and crossover operator have been redesigned to implement the discrete DE directly. In addition, to increase the solution diversity by expanding the search space, we present a new selection strategy with probabilistic mechanism to determine new target vectors in the next iteration. Meanwhile, 2-opt heuristic and 2-exchange neighborhood is embedded in the ACO&DE to improve the local search performance. The results have shown that the proposed ACO&DE algorithm is competitive with existing optimal methods in solving the VRP, and thus can be further extended in variants of the VRP and other logistics transportation fields.**

***Index Terms*—Vehicle routing problem, 2-opt, Ant colony optimization, Selection operation, Differential evolution.**

I. Introduction

The effective transportation of goods has great influence on transportation cost and customer satisfaction by reducing the energy consumption and fast delivery. Most goods transportation problems can be defined as the vehicle routing problems. The VRP belongs to a classical optimization problem in logistics system. In the VRP, customers geographically located on the network are served by vehicles with capacity. Each vehicle with capacity constraints should start from the central depot, and finally return to it, and each customer should be visited by a vehicle only once. The objective function is to find the best vehicle paths that minimize total visited distance and the number of vehicles. The VRP is a NP-hard problem with wide applications, and it is complicated to solve with common optimization methods. The development of exact algorithms and approximate methods has made important progress. Pessoa et al. [1] presented an extended formulation and adopted a branch cut and price method to resolve the instances with 75 customers. Baldacci and Mingozzi [2] proposed a set partitioning based on Lagrangian relaxation to solve small instances. Nevertheless, this exact method can not be suitable for larger instances. Because of the high time complexity and more time consuming and calculation, exact algorithms is suitable for small cases. In practical application, the problems that need to be solved are large scale, scholars continue to seek approximate or heuristic algorithms to solve this problem.

At present, most researchers have focused particularly on designing the effective heuristics to improve the algorithm performance. Clarke and Wright [3] designed constructive heuristics, which build vehicle routes successively by adding unvisited customers according to saving criteria. These constructive heuristics can be conveniently improved by executing 2-opt procedure as local search. Recent research concentrates on the use of modern meta-heuristics. Taillard [4] and Rochat and Taillard[5] adopted tabu search to obtain the best known results for the VRPs while Osman[6] used simulated annealing to get similar results. Barrie et al. [7] incorporated several neighborhoods into the genetic algorithm (GA) to obtain more significant improvement and demonstrated that the GA is an effective method to solve the basic VRP. Asma et al. [8] presented a hybrid firefly algorithm integrated with genetic operators and two local searches to increase the solution's quality. The experiments demonstrated that hybrid firefly algorithm has high computational accuracy. Nevertheless, these algorithms usually need more computing times and some parameter settings to get better solutions [9], the performance of these algorithms still need to be further studied and improved

Manuscript received March 18, 2021; revised July 22, 2021. This work was supported by national college students' innovation and entrepreneurship training program project in 2021.

H. B. Li is a Student of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan, 114051, China (e_mail: 958789852@qq.com).

X. X. Zhang is a Professor of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan,11041,China (corresponding author, phone:86-0412-5929812; e_mail: aszhangxx@163.com).

S. Fu is a Master Student of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan, 114051, China (e_mail: 15904922295@qq.com).

Y. Y. Hu is a Master Student of School of Computer Science and Software Engineering, University of Science and Technology LiaoNing, Anshan, 114051, China (e_mail: 2799386740@qq.com).

The ACO is a meta-heuristic algorithm [10] and it imitates the process of ant colony behavior as they have ability to obtain the best path from the nest to the food source. Bullnheimer et al. [11] first proposed an ant system algorithm to solve the VRP, and then designed another improved algorithm incorporating ant colony optimization with 2-opt heuristic to improve the algorithm performance. Due to the complexity of the VRP, the basic ACO algorithms is difficult to deal with complicated information requirements. Many researchers have presented novel strategies to increase the high effectiveness of basic ACO and successfully applied these strategies to different variants of the VRP. It is an significant research topic to integrate the ACO algorithm with other methods for solving the large scale VRP. For each metaheuristc algorithm has own advantages and weakness, researchers have been exploring hybrid algorithms, which can take advantages of each algorithm to make up for its shortcomings.

More recently, differential evolution (DE) is a novel mataheuristic algorithm inspired from organism intelligent behaviors. The DE algorithm was originally presented for continuous optimization [12], and it could make the population evolve at each generation through simple operations. Because of its simple operations and fast convergence, the DE algorithms are successfully applied to solve continuous problems in different fields [13], including pattern recognition [14], and power systems [15], among others. However, the standard DE algorithm operates in continuous space problems, which is not used directly to solve discrete combinatorial optimization problems. Compared with continuous optimization, discrete optimization problems is more complex, therefore the literature on DE algorithm to solve discrete optimization problems is relatively limited. At present, some researchers have proposed improved strategies to extend application fields of the DE method. Pan et al. [16] presented a discrete DE method to solve flowshop scheduling problems. Zhang et al. [17] designed a novel DE for solving distributed blocking flowshop problem. Ali et al. [18] proposed a discrete DE method to solve traveling salesman problems. Experiment results have shown this DE algorithm can outperform other comparative methods. Since the DE algorithm has some advantages, i.e., simplicity, fast convergence and so on, it is successfully applied in resource-constrained project, flow show scheduling problems. The limitations of using discrete DE to solve the VRP and promising applications of discrete DE form the main motivation of our research.

Although meta-heuristics has made some progress in the vrp, a new methodology still needs further research and discussion. This paper presents a hybrid ACO&DE algorithm to solve the VRP. The main innovation of this paper is different from all the above literatures in the following aspects. First, the crucial idea is to combine the ACO with the DE to form a hybrid ACO&DE algorithm. The ACO&DE can make use of advantages of the ACO and DE algorithm to make up for its own weakness. Second, within the ACO&DE framework, an effective local search based on 2-opt heuristic and 2-exchange neighborhood is embedded in ACO&DE algorithm to expand the local search ability of algorithms. 2-opt heuristic should be applied to execute as an improved method within the same route, and 2-exchange can be used for the between two routes. To enhance the performance of the ACO for the VRP, the DE algorithm is also applied to extend the search scope of the ACO algorithm. Third, different from the basic DE, the hybrid ACO&DE algorithm adopted several vehicle sets to denote a solution, each of which represents the permutation of visiting customers. Moreover, to make the DE suitable for solving the VRP, both strategies of mutation operator and crossover operator have been redesigned to implement the discrete DE directly. In addition, a new selection strategy is adopted to promote the solution diversity by expanding the search space. In addition, the common selection strategy is survival of the fittest, and it is very easy to lead to local optimum. Hence, to increase the solution diversity by expanding the search space, we present a new selection strategy with probabilistic mechanism to determine new target vectors in the next iteration. That is to say, a certain quantity of inferior solutions can enter into the next generation according to the probability mechanism. Finally, numerical experiment results shows that the proposed ACO&DE algorithm is competitive with existing optimal methods in solving the VRP.

The rest of this paper is arranged as follows. In Section 2, we present a description of the VRP, while Section 3 briefly gives basic ACO and DE algorithm. In Section 4, the proposed ACO&DE algorithm is introduced. Section 5 discusses experimental results. At last some conclusions are drawn in Section 6.

II. Problem Description

The VRP has always been a hot issue for scholars in the logistics and transportation. The VRP is often described as an undirected graph $G=(V, A)$, in which $V=\{0,\ldots, n\}$ denotes a set of nodes, node 0 represents a depot, namely a central depot, and a node corresponds to a customer position with a demand d_i, $i=\{1,\ldots, n\}$, to be transported. Let $A=\{(i, j)|i, j\in V\}$ be an edge set. The distance c_{ij} is related to each edge $(i,j)\in A$ and denotes the distance between customers (i, j). M denotes a set, i.e., $M=\{1,\ldots, m\}$, in which m indicates the number of serving vehicles. Customers geographically located are served by vehicles and assigned at the depot, while each vehicle leaves from the depot and must return to it. Based on the above description, the VRP model can be defined mathematically as follows:

$$\text{Minimize}\sum_{k=1}^{m}\sum_{i=0}^{n}\sum_{j=0}^{n}c_{ij}x_{ijk} \tag{1}$$

$$\sum_{k=1}^{m}y_{ik}=1,\quad i\in N \tag{2}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{n}x_{0ik}=\sum_{k=1}^{m}\sum_{i=1}^{n}x_{i0k} \tag{3}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}d_i x_{ijk}\le Q\ ,k\in M; \tag{4}$$

$$\sum_{j=1, j\neq i}^{n} x_{ijk} = \sum_{j=1, j\neq i}^{n} x_{jik} = y_{ik}, i \in N; k \in M \quad (5)$$

$$\sum_{i, j \in \Omega} x_{ijk} \leq |\Omega| - 1, 2 \leq |\Omega| \leq n; k \in M \quad (6)$$

$$x_{ijk} \in \{0,1\}, \quad i, j \in N, k \in M \quad (7)$$

$$y_{ik} \in \{0,1\}, \quad i \in N, k \in M \quad (8)$$

Formula (1) is an objective function that minimizes total visited distance by all vehicles originating from a depot to visit a set of customers. Constraints (2) indicate that a customer should be served by one vehicle at most. Constraints (3) ensure that a vehicle should leave and finish a route at this depot. The Constraints (4) assure that each vehicle should not exceed a predetermined amount. Constraints (5) should guarantee that when a customer is served in a certain vehicle, there must be exactly one customer before and one customer after it in the vehicle respectively. Constraints (6) avoid producing subtours. Constraints (7) and (8) specify decision variables as binary. Consider decision variables x_{ijk} such that x_{ijk} =1 if the vehicle k visited from node i to node j, otherwise, x_{ijk}=0. Consider variables y_{ik}=1 which represent if node i is visited by vehicle k , otherwise, y_{ik}=0.

III. Ant colony optimization and Differential Evolution

The ACO algorithm was first presented by Dorigo et al. [10]. Their inspiration came from direct observation of ant colonies. Through communicating pheromone information, the ACO algorithm simulates the process of ant colony behavior as ants can seek the shortest routes between their nest and food source position. When ants find food sources, the pheromone trails will attract the other ants to move. The paths with more pheromone trails can increase the probability of other ants choosing these ones. Over a long time, as more and more the ants have finished their shorter paths, more pheromone trails deposit on the shorter paths, but are less intensified on the longer paths. Ants can not only seek the shortest route between a food position and the nest, but adapt to surrounding environmental change. The basic principle of ACO algorithms is based on the natural behaviors of seeking the shortest route from a nest to the food position. The ACO algorithm mainly includes solution construction rule and pheromone updating rule. Pseudo of the ACO algorithm is listed in Fig.1.

The DE method is also regarded as a meta-heuristic algorithm based on a population stochastic search. Just like other meta-heuristic algorithms, the DE consists of four steps, i.e., population initialization phase, mutation, crossover and selection operations. The main principle of this algorithm is to make the solutions evolve continually at each generation through above these steps. Starting with the population of target vectors, the DE algorithm can produce mutant vectors by adopting a mutation operation. Then, to expand the diversity of the population, these mutant vectors are recombined with target vectors to obtain trail vectors by implementing the crossover process. At the same time, trail

```
Begin
  Initialize pheromone values
  While (the termination condition is not met) do
    For each ant i, i = 1, 2, ..., NP
      Construct a new solution Xi, i = 1, 2, ..., NP, using
       probabilistic rule
      Update the local pheromone trails
    End For
    Update the global pheromone trails
  End While
 Output the best solution XB in the population
End
```

Fig. 1. Pseudo of ant colony optimization.

vectors can inherit some features from mutant vectors. After that, competition between a target vector and a trial vector should make the better one participate in the next iteration according to the survival of the fittest. These operations are performed repeatedly until a predefined termination condition is met. Generally, the performance of this algorithm is affected by the initial population size, the mutation strategy, crossover factor, selection strategy, and other factors. Therefore, many variants of the DE algorithms have been proposed in research literatures to focus on these factors to enhance performance of the DE. DE algorithm is a random search process that can record and share the optimal information within the population. The complexity of simple evolutionary process is reduced and the global convergence ability is enhanced. As being used in the continuous space, DE algorithm easily operates the floating points, so it can decrease the computational time in the process of encoding and decoding. Pseudo of the DE algorithm is listed in Fig.2

```
Begin
  Initialize a population set P1, P1={X1, X2, …, XNP}.
  While the termination condition is not met
    For each solution Xi, i = 1, 2, ..., NP
      Generate a mutant vector Vi according to Mutation operation
      Generate a trail vector Ui according to Crossover operation
    End For
    For each solution Xi, i = 1, 2, ..., NP
      Update Xi with Ui according to Selection operation
    End For
  End While
  Output the best solution XB in the population
End
```

Fig. 2. Pseudo of differential evolution.

In the paper, we present a novel hybrid ACO&DE, which adopts a distinctive probability selection operation to produce new individuals in the next generation. The ACO&DE possesses the excellent features of the DE. Moreover, the probability selection operation can expand the diversity of the population. Therefore, the hybrid ACO&DE offers a novel effective method for vehicle routing problems.

IV. Hybrid ACO&DE Algorithm

In the process of solving the VRP, on the one hand, determine which customers a vehicle will visit, on the other hand, each vehicle should also consider the order of visiting customers. Therefore, the VRP is an extremely complex problem, which makes solution methods more difficult. We presented a novel hybrid ACO&DE algorithm integrating the ACO and differential evolution (DE) to solve the VRP. Meanwhile, The ACO&DE can take advantages of ACO, the ability to construct a solution quickly, and that of DE, the ability to expand search areas to seek the global optimal solution. Furthermore, within the ACO&DE framework, an effective local search based on 2-opt heuristic and 2-exchange neighborhood is embedded into the ACO&DE to enhance the local search ability. To make the DE suitable for solving the VRP, both strategies of mutation operator and crossover operator have been redesigned to implement the discrete DE directly. The proposed ACO&DE algorithm consists of solution representation, solution construction, some operations and pheromone trail updating.

A. Solution Representation

For a VRP with n customers and m vehicles, a feasible solution is defined as a set of m vehicle permutations, i.e., $\boldsymbol{X}=\{X_k \mid (k=1, 2, \ldots, m)\}$, in which $X_k=\{x_{i,1}, x_{i,2}, \ldots, x_{i,nk}\}$ is a sequence of customers and $\sum_{k=1}^{m} n_k = n$. Such a solution form is decoded easily to a VRP schedule. In this schedule, X_k denotes the partial customer schedule at vehicle k. In others words, it indicates the assigned customers and their visiting order at vehicle k. For an example as shown in Fig. 3, consider a VRP with n=12 and m=3. A feasible solution is X={$\boldsymbol{X_1}$, $\boldsymbol{X_2}$, $\boldsymbol{X_3}$}, in which $\boldsymbol{X_1}$ = (v_2, v_8, v_{11}, v_{10}, v_1), $\boldsymbol{X_2}$ = (v_9, v_{12}, v_6) and $\boldsymbol{X_3}$ = (v_4, v_7, v_5, v_3). Using the decoding strategy, customers v_2, v_8, v_{11} , v_{10} , and v_1 are assigned at the first vehicle, and visited in the order of customers $v_2 \rightarrow v_8 \rightarrow v_{11} \rightarrow v_{10} \rightarrow v_1$, customers v_9, v_{12}, and v_6 at the second vehicle following customers $v_9 \rightarrow v_{12} \rightarrow v_6$, and customers v_4, v_7, v_5 and v_3 at the third vehicle following customers $v_4 \rightarrow v_7 \rightarrow v_5 \rightarrow v_3$.

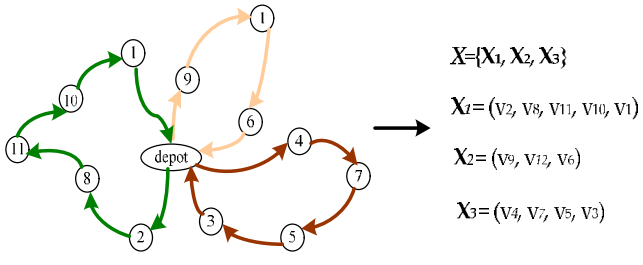

Fig. 3. Illustration of a solution representation.

B. Solution Construction

In our ACO&DE algorithm, the initial population is generated by two different construction methods so that it should ensure the diversity of population. One method is a greedy algorithm, the other is ACO algorithm. The greedy algorithm adopts the principle of nearest neighbor method, which select the nearest customer from the current point. Randomly choose a customer as the current point, and add it into the current vehicle route. And then, determine the nearest customer from the current node at each step. The vehicle keeps visiting customer until it reaches its capacity. When all customers are visited by vehicles, terminate the algorithm process, and output a feasible solution. Fig.4 presents the pseudo of greedy algorithm.

```
Begin
  Initialize customer set V={0,1, 2, ..., n}, vehicle set M={1, 2, ..., m},
          vehicle capacity Q, current loading capacity Qk=0,
          customer demand di, i∈V/{0}, each vehicle set Vk=, k∈M
  For each vehicle k, k = 1, 2, ..., m
    Choose randomly a customer i∈V
    Assign the customer i to vehicle k, Vk=Vk∪{i}
    Repeat
    Choose the closest customer j of V to customer i
    Compute vehicle capacity for customer j
    If di+Qk <=Q then
      Assign customer j to the current vehicle, compute current vehicle Qk
        Qk =di+ Qk
      Discard customer j from V, V=V/{j}
        Update current customer i, i=j;
      End if
    Until no customer j to the current vehicle k
  End for
End
```

Fig. 4. Pseudo of greedy algorithm.

We mainly focus on the construction mechanism of ACO algorithm. In the VRP, a feasible solution is directly defined as a set of m vehicle permutations. Thus, an artificial ant represents a vehicle in the construction solution process of ACO algorithm. Initially the ants begin from a depot and successively visit customers, until every customer is served. The ant k at the node i moves to node j according to the following probabilistic rule:

$$j=\begin{cases}\underset{u\notin M_k}{\operatorname{argmax}}\{t_{il}\cdot[h_{il}]^b\}, & \text{if } q\le q_0\\ S, & \text{otherwise}\end{cases} \tag{9}$$

$$P_{ij}^k=\begin{cases}\dfrac{[\tau_{ij}]^\alpha[\eta_{ij}]^\beta}{\sum_{l\notin M_k}[\tau_{il}]^\alpha[\eta_{il}]^\beta}, & j\notin M_k\\ 0, & \text{otherwise}\end{cases} \tag{10}$$

In which t_{il} denotes the pheromone trail of edge (i,l), and b is the visibility. The variable q represents a random variable ($0\le q \le 1$), and parameter q_0 ($0\le q_0 \le 1$) can determine the relative significance of exploitation. If $q\le q_0$ then the short edge is chosen according to Equation (9); otherwise, determine an edge according to S. S denotes a variable depending on the probability rule as in Formula (10). The visibility h_{il} is the inverse of the edge length, and M_k record nodes already traveled by an ant.

C. Mutation Operation

After initializing the population, each vector X_i, i = 1, 2,⋯, NP, defined as the target vector, can produce a mutant vector V_i. This mutation vector V_i = {$v_{i,1}$, $v_{i,2}$, ..., $v_{i,j}$, ..., $v_{i,D}$, i = 1, 2, ..., NP, j = 1, 2, ..., D}, is obtained by using three randomly chosen target vectors. With regard to each target individual X_i, the mutation operation is executed to produce a mutation vector V_i by calculating the difference between two randomly

chosen solutions from the population. That is to say, three vectors X_{p1}, X_{p2}, X_{p3} shall be randomly opted from the current population such that, p_1, p_2, $p_3 \in \{1, 2, \ldots, NP\}$, and $p_1 \neq p_2 \neq p_3 \neq i$. A mutant vector V_i is generated as follows:

$$V_i = X_{p_1} + F \cdot (X_{p_2} - X_{p_3}) \quad p_1 \neq p_2 \neq p_3 \neq i \tag{11}$$

in which F is a mutation scale parameter, $F \in [0,2]$, which can control the search direction by amplifying the differential values $\boldsymbol{X_{p2}} - \boldsymbol{X_{p3}}$.

To make the DE suitable for discrete problems, a new mutation strategy has been redesigned to solve the VRP directly. Let $\boldsymbol{d}$ be the differential set, i.e., $\boldsymbol{d} = \boldsymbol{X_{p2}} - \boldsymbol{X_{p3}}$, and the subtraction operator between $\boldsymbol{X_{p2}}$ and $\boldsymbol{X_{p3}}$ is performed as follows:

$$x_{p2,j} - x_{p3,j} = \begin{cases} x_{p2,j}, & \text{if } x_{p2,j} \neq x_{p3,j} \\ 0 & \text{otherwise} \end{cases}, j = 1, 2, \ldots, D \tag{12}$$

If $x_{p2,j}$ and $x_{p3,j}$ have the different elements, the resulting set is $x_{p2,j}$; otherwise it is 0. Let J be a scale differential set, i.e., $J = F \times \boldsymbol{d}$. The scale operator between the scale parameter F and $\boldsymbol{d}$ is stated as

$$F \times d_j = \begin{cases} d_j, & \text{if } r \leq F \\ 0, & \text{otherwise} \end{cases}, j = 1, 2, \ldots, D \tag{13}$$

In which $r \in [0,1]$ denotes a random value, and $\mathbf{d}_j$ is the jth element of $\boldsymbol{d}$. If $r \leq F$, the scale differential set J learns from the corresponding $\mathbf{d}_j$; otherwise it is 0. And then, adding a nonzero differential vector J to the base set $\boldsymbol{X_{p1}}$ can obtain a required mutant solution $\boldsymbol{V_i}$. The sum calculation between $\boldsymbol{X_{p1}}$ and J is defined as follows:

$$x_{p1,j} + J_j = \begin{cases} x_{p1,j}, & \text{if } J_j = 0 \\ J_j, & \text{otherwise} \end{cases}, j = 1, 2, \ldots, D \tag{14}$$

For a target vector $\boldsymbol{X_i}$, a mutant solution $\boldsymbol{V_i}$ obtains elements from J if the condition $J_j = 0$ is met, otherwise, the remaining elements of $\boldsymbol{V_i}$ are copied from the corresponding the base set $\boldsymbol{X_{p1}}$. A mutant solution $\boldsymbol{V_i}$ should be generated by redefining on the above operators. An example is explained in Fig. 5. Note that $\boldsymbol{V_i}$ may not be feasible, but subsequent operators can make it feasible.

In order to explain the redefined mutant process, consider $\boldsymbol{X_{p1}} = \{[v_1, v_2, v_3], [v_4, v_5, v_6]\}$ $\boldsymbol{X_{p2}} = \{v_4, v_2, v_3, v_5, v_6, v_1\}$, $X_{p3} = \{v_1, v_2, v_5, v_4, v_6, v_3\}$, and $F = 0.5$. A mutant solution $\boldsymbol{V_i}$ is described as follows:

Step 1: Arrange a permutation $\boldsymbol{X_{p1}}$, $\boldsymbol{X_{p2}}$ and X_{p3} in a row and get customer sequence, i.e., $\boldsymbol{X_{p1}} = \{[v_1, v_2, v_3], [v_4, v_5, v_6]\}$, $\boldsymbol{X_{p2}} = \{[v_4, v_2, v_3], [v_5, v_6, v_1]\}$, $\boldsymbol{X_{p3}} = \{[v_1, v_2, v_5], [v_4, v_6, v_3]\}$.

Step 2: Applying formulas (12) and (13) for $\boldsymbol{X_{p2}}$ and $\boldsymbol{X_{p3}}$ to obtain the differential set $\boldsymbol{d} = \{v_4, 0, v_3, v_5, 0, v_1\}$.

Step 3: Produce a random number set, $\boldsymbol{r} = \{r_1, r_2, \ldots, r_n\}$. The set $\boldsymbol{r}$ is orderly as 0.3, 0.4, 0.6, 0.2, 0.7 and 0.8, i.e., $\boldsymbol{r} = \{0.3, 0.4, 0.6, 0.2, 0.7, 0.8\}$.

Step 4: Generate the scale differential set J according to formula (14). For random number r_1, r_2, and r_4 are smaller than scale parameter F, set $J_1 = v_4$, $J_2 = 0$ and $J_4 = v_5$, and the remaining elements of J are 0, i.e., $J = \{v_4, 0, 0, v_5, 0, 0\}$.

Step 5: Applying formula (15) for $\boldsymbol{X_{p1}} = \{[v_5, v_3, v_6], [v_4, v_1, v_2]\}$ and $J = \{v_4, 0, 0, v_5, 0, 0\}$ to produce a mutant solution $\boldsymbol{V_i}$. For $v_2 \neq 0$, $v_5 \neq 0$, set $v_{i,1} = v_4$, and $v_{i,4} = v_5$. And then, the remaining elements of $\boldsymbol{V_i}$ are copied from the remaining customer of $\boldsymbol{X_{p1}}$ in their original order. Set $v_{i,2} = v_3$, $v_{i,3} = v_6$, $v_{i,5} = v_1$ and $v_{i,6} = v_2$.

Step 6: Output the mutant solution $\boldsymbol{V_i} = \{[v_4, v_3, v_6], [v_5, v_1, v_2]\}$.

D. Crossover Operation

A crossover operation can be used to improve the global searching ability. A trail vector, $\boldsymbol{U_i}$, $i = 1, 2, \ldots, NP$, can be obtained by recombining $\boldsymbol{X_i}$ with $\boldsymbol{V_i}$ in which this operation process makes $\boldsymbol{U_i}$ inherit some features from the mutant vector. Let p_{cr} be a crossover parameter in [0,1]. If the value of parameter p_{cr} is larger, it means that the trail vector can inherit more features from the mutant vector. A trail vector $\boldsymbol{U_i} = (u_{i,1}, u_{i,2}, \ldots, u_{i,D})$ can be obtained as follows:

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } r_j \leq p_{cr} \\ x_{i,j}, & \text{otherwise} \end{cases} \tag{15}$$

where $u_{i,j}$ denotes the jth element of vector $\boldsymbol{U_i}$, $j = 1, 2, \ldots, D$, $i = 1, 2, \ldots, NP$, and $r_j \in (0, 1)$ is a random number. Trail vector $\boldsymbol{U_i}$ obtains elements from $\boldsymbol{V_i}$ if the condition $r_j \leq p_{cr}$ is satisfied, otherwise, the remaining elements of the trail vector $\boldsymbol{U_i}$ are obtained from the corresponding $\boldsymbol{X_i}$.

To solve the VRP, a new cross strategy has been redesigned. The difference between this cross operation and basic cross operation is that if the condition $rand(j) > p_{cr}$ is satisfied, it is not simply copying from the corresponding target vector $\boldsymbol{X_i}$, but filling in the remaining vacant positions of $\boldsymbol{U_i}$ with the rest customer of $\boldsymbol{X_i}$. An example of the mutation and crossover operators is explained in Fig. 5. Note that $\boldsymbol{U_i}$ cannot always represent a feasible schedule because some vehicle paths may not meet the capacity constraints. A trial individual $\boldsymbol{U_i}$ can be obtained by the crossover operator.

In order to illustrate the cross operation, consider $\boldsymbol{V_i} = \{[v_4, v_3, v_6], [v_5, v_1, v_2]\}$, $\boldsymbol{X_i} = \{[v_1, v_3, v_6], [v_4, v_2, v_5]\}$, and $p_{cr} = 0.5$. The crossover operator is explained as follows:

Step 1: Arrange the permutation $\boldsymbol{V_i}$ and $\boldsymbol{X_i}$ in a row and produce customer sequence, i.e., $\boldsymbol{V_i} = \{[v_4, v_3, v_6], [v_5, v_1, v_2]\}$, $\boldsymbol{X_i} = \{[v_1, v_3, v_6], [v_4, v_2, v_5]\}$.

Step 2: Produce a set $\boldsymbol{r} = \{r_1, r_2, \ldots, r_n\}$. The random number set $\boldsymbol{r}$ is orderly as 0.3, 0.6, 0.4, 0.2, 0.7 and 0.8, i.e., $\boldsymbol{r} = \{0.3, 0.6, 0.4, 0.2, 0.7, 0.8\}$.

Step 3: Apply formula (15) for $\boldsymbol{X_i} = \{[v_1, v_3, v_6], [v_4, v_2, v_5]\}$ and $\boldsymbol{r} = \{0.3, 0.6, 0.4, 0.2, 0.7, 0.8\}$ to produce elements of a mutant solution $\boldsymbol{U_i}$. Set $u_{i,j} = v_{i,j}$, if $r_j \leq p_{cr}$, $j = 1, 2, \ldots, n$ i.e., $u_{i,1} = v_4$, $u_{i,3} = v_6$ and $u_{i,4} = v_5$.

Step 4: Fill out the remaining empty elements of $\boldsymbol{U_i}$ with the rest elements of $\boldsymbol{X_i}$ in their original order. Set $u_{i,2} = v_1$, $u_{i,5} = v_3$ and $u_{i,6} = v_2$, and then obtain a trail vector $\boldsymbol{U_i} = \{[v_4, v_1, v_6], [v_5, v_3, v_2]\}$.

Step 5: Output the trail vector $\boldsymbol{U_i}$.

E. Selection Operation

Following crossover operation, compute the function values of vector $\boldsymbol{U_i}$ and target vector $\boldsymbol{X_i}$. If $f(\boldsymbol{U_i}) \leq f(\boldsymbol{X_i})$

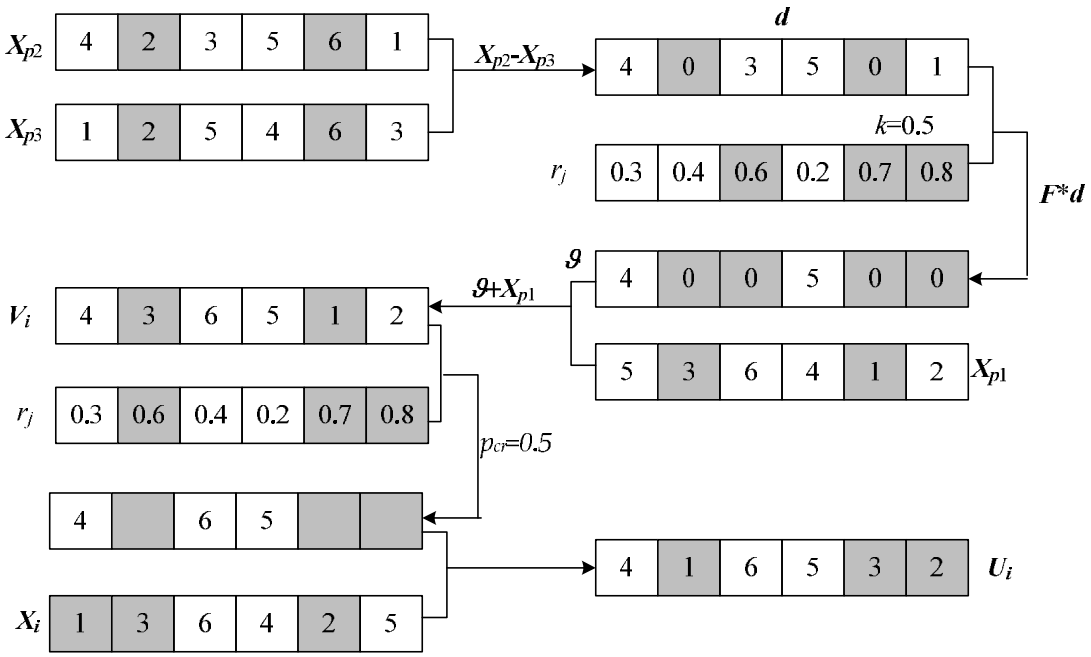

Fig. 5. An example of mutation and crossover.

condition is met , then $\boldsymbol{U_i}$ will replace the target vector $\boldsymbol{X_i}$, i.e. $\boldsymbol{X_i}$=$\boldsymbol{U_i}$; otherwise, the vector $\boldsymbol{X_i}$ will be unchanged. The trial vector with the lower function value will survive to the next iteration. The selection scheme based on a greedy strategy is defined as follows:

$$X_i = \begin{cases} U_i, & if\ f(U_i) \le f(X_i) \\ X_i, & \text{otherwise} \end{cases} \tag{16}$$

In which $f(\boldsymbol{U_i})$ and $f(\boldsymbol{X_i})$ denote the objective values of vector $\boldsymbol{U_i}$ and vector $\boldsymbol{X_i}$, respectively. With regard to the minimization problem, the vectors with the lower function values have more choices to participate in the next iteration during the search process. The selection strategy has a shortcoming, that is, it will be easy to lead to local optimum. Hence, to overcome the shortcoming, we design a novel selection strategy with probabilistic escaping mechanism to choose whether trial vectors will be members of the next iteration. The selection strategy is determined as follows,

$$X_i = \begin{cases} U_i & \text{if } f(U_i) \le f(X_i) \\ U_i & \text{if } r < \exp(-(f(U_i) - f(X_i))/T) \\ X_i & \text{otherwise} \end{cases} \tag{17}$$

Where T is temperature. Evaluate the function values of two vectors $\boldsymbol{U_i}$, $\boldsymbol{X_i}$, and determine whether to update $\boldsymbol{X_i}$ with $\boldsymbol{U_i}$. If $f(\boldsymbol{U_i})$≤$f(\boldsymbol{X_i})$, then $\boldsymbol{U_i}$ is accepted and update $\boldsymbol{X_i}$ with it. On the other hand, if $f(\boldsymbol{U_i})$>$f(\boldsymbol{X_i})$ can be accepted with probability according to formula (14). By introducing this new selection strategy into DE algorithm, some solutions with poor quality may become members of the next iteration. Thus, the target vectors with large dispersion can be obtained by exploring the different regions, and this selection strategy can avoid the algorithm fall into local minimum.

F. Pheromone Trail Updating

In ACO, the pheromone trail updating is divided into local updating and global updating. Local trail is updated in the process of constructing solutions, and global trail is performed after solutions are built. The local updating is to avoid generating more pheromone on edges being chosen by other ants. The local pheromone trail is performed as follows:

$$t_{ij} = (1 - r)t_{ij} + g/(n \cdot L_0) \tag{18}$$

in which $r \in [0,1]$ denotes a evaporation parameter, L_0 represents initial function value of a solution, and n means the customer number.

Global pheromone updating is to enhance neighborhood search of the optimum solution. The global updating can not only converge faster by expanding differences search spaces, but also overcome the rapid enhancement of pheromones on optimal edges. In this ACO, the pheromone trails on the edges of the best solution should be updated as the following formula,

$$t_{ij} = (1 - f) \cdot t_{ij} + f/L_{best} \tag{19}$$

In which $f \in [0,1]$ denotes a decay parameter, L_{best} represents the distance of the best solution.

G. Procedure of the ACO&DE Algorithm

In this paper, we present a hybrid algorithm which hybridizes the ACO with DE algorithm to enhance the performance of the algorithm. The hybrid algorithm mainly consists of four stages. For the first stage, generate initial solutions with different methods. In the second stage, DE algorithm is adopted to enhance solution quality to extend the search scope of ant colony algorithm. Finally, 2-opt heuristic and 2-exchange neighborhood is embedded in the ACO&DE to expand the local search ability. The steps of hybrid ACO&DE algorithm are as follows:

Step 1: Initialization. Input pheromone trails on all edges. Set population number as *PopSize*, and $\boldsymbol{P}=\{f\}$ as initial population set. Set *iteration*= 0 where *iteration* will be compared to terminal condition, i.e., the maximum number of iterations *MaxIteration*. Set initial temperature $T=T_0$.

Step 2: Generate *NP* solutions with a greedy heuristic, each solution $\boldsymbol{X_i}$, i = 1, 2, ..., *NP*, and put these solutions in the solution set $\boldsymbol{P}_1$, $\boldsymbol{P}_1=\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_i, \ldots, \boldsymbol{X}_{NP}\}$. Then, add these solutions in the population set $\boldsymbol{P}$, $\boldsymbol{P}=\boldsymbol{P}\cup\boldsymbol{P}_1$.

Step 3: Generate $|\boldsymbol{P}|$-$\boldsymbol{NP}$ solutions by executing construction solution procedure of ACO algorithm, and put these solutions in the solution set $\boldsymbol{P}_2$, $\boldsymbol{P}_2=\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_i, \ldots, \boldsymbol{X}_{|\boldsymbol{P}|-\boldsymbol{NP}}\}$. Then, add these solutions in the population set $\boldsymbol{P}$, $\boldsymbol{P}=\boldsymbol{P}\cup\boldsymbol{P}_2$.

Step 4: Select randomly *NP* solutions from the population set $\boldsymbol{P}$ as target individuals.

Step 5: Mutation operation. Choose randomly three target vectors X_{p1}, X_{p2}, X_{p3} from the population, p_1, p_2, $p_3\in\{1, 2,\ldots, NP\}$. A mutation vector $\boldsymbol{V_i}$, i=1, 2, ..., *NP*, is produced according to formulas (11)- (14).

Step 6: Crossover operation. Produce a new trial vector, $\boldsymbol{U_i}$, i=1, 2, ..., *NP*, by combining a target vector $\boldsymbol{X_i}$ with a relevant mutated vector $\boldsymbol{V_i}$ based on the formula (15).

Step 7: Selection operation. Compute the fitness function values of $\boldsymbol{U_i}$ and $\boldsymbol{X_i}$, i=1, 2,…, *NP*. If $f(\boldsymbol{U_i})<f(\boldsymbol{X_i})$ condition is met, then $\boldsymbol{U_i}$ will replace $\boldsymbol{X_i}$, i.e. $\boldsymbol{X_i}=\boldsymbol{U_i}$; otherwise, a certain inferior solutions can be accepted with a probability according to Equation (17).

Step 8: Improvement solutions. After the new solutions are obtained, the 2-opt method should be used as an improved method within the route, and 2-exchange can be executed for the between two routes.

Step 9: Compute the function values of the solutions and update the best solution $\boldsymbol{X_B}$ in population set. Record the best solution $\boldsymbol{X_B}$ with the minimum cost.

Step 10: Update global pheromone trail on the edges of $\boldsymbol{X_B}$ according to formula (19).

Step 11: Update the population. Some individuals in the population are replaced with new offspring.

Step 12: Increase the number of iterations and decrease the temperature, i.e., *iteration* = *iteration* +1,$T=T*k$.

Step 13: Check whether the terminal condition is satisfied, if *iteration*<*MaxIteration*, repeat Step2-12 until the terminal criteria condition is satisfied, terminate the algorithm evolution process, and output the optimal solution.

V. Computational results

In the section, we evaluated the effectiveness of the ACO&DE algorithm. We first present data set and parameter setting. Next, we compared ACO&DE with basic ACO and DE algorithms. Finally, ACO&DE and other methods were used for comparison

A. Data set and Parameter Setting

In this study, the ACO&DE algorithm as described in Section 4 was coded in the visual C++. To test validity of ACO&DE algorithm, it has been implemented on benchmark instances selected randomly and obtained from the OR Library. The website is at available at http://neo.lcc.uma.es/vrp/. These instances introduced by Taillard et al. [4] include the A, B and C series. We select 12 different instances to test the performance of our proposed method in the experimental data. The first 3 instances and the second 2 instances are respectively from A and B series presented by Augerat et al. and the remaining 7 instances are from C series proposed by Christofides. The instances are named as A1-A3, B1-B2, C1-C5, and C11-C12. In addition to the depot, the size of customers per instance is between 30 and 199, and these instances have no service time and maximum length constraints. Table I lists a description of instances. In Table I, *Instance* denotes the test instance, *n* denotes the number of nodes, *m* is the size of vehicles, *Q* is vehicle capacity, and *Best_known* refers to the shortest distance reported in some literature.

There are several parameters in the ACO&DE algorithm, and their values have a certain impact on the final results. For small-scale cases, the algorithm can find optimal solutions. Therefore, some parameter values should been obtained on small-scale cases. Table II lists the parameter values used by ACO&DE algorithm.

TABLE I
THE DESCRIPTION OF INSTANCES.

No.	*Instance*	*n*	*m*	*Q*	*Best_known*
1	A1	31	5	100	784
2	A2	32	5	100	661
3	A3	32	6	100	742
4	B1	30	5	100	672
5	B2	33	5	100	788
6	C1	50	5	160	524.61
7	C2	75	10	140	835.26
8	C3	100	8	200	826.14
9	C4	150	12	200	1028.42
10	C5	199	17	200	1291.45
11	C11	120	7	200	1042.11
12	C12	100	10	200	819.56

TABLE II
PARAMETER VALUES IN EXPERIMENTS.

No.	Terminology	Symbol	value
1	Importance of distance	β	{3, 4, 5}
2	Relative importance of exploitation	q_0	0.60–0.85
3	Pheromone evaporation	ρ	0.35–0.60
4	Population number	*PopSize*	20
5	Target vector number	*NP*	10
6	Crossover parameter	p_{cr}	0–1.0
7	Max iteration	*MaxIteration*	1000

B. Comparison Results and Analysis

To justify the performance of ACO&DE algorithm, the ACO&DE is compared with basic ACO and DE algorithms. The difference between basic ACO and DE algorithms lies in the different construction solution mechanism of population. The ACO uses construction solution mechanism of ACO algorithm, and the DE adopts a greedy construction solution mechanism. Each method run 10 times on one instance and then tables III listed the corresponding best and average distances and computational time obtained using three methods. In Table III, ***Best*** refers to the best solutions, and ***Mean*** to the average values of solutions for 10 runs. For each method, ***Best a***nd ***Mean*** are adopted as the evaluation criterion to measure the performance of methods. For providing a fair comparison, the results of three algorithms are show in Table III. First, the ACO&DE algorithm is compared with basic ACO. With regard to ***Best*** and ***mean*** values, the ***Best*** value of the ACO&DE algorithm is smaller than that of basic ACO, and the ***Mean*** value of the ACO&DE algorithm is smaller than that of basic ACO. Thus, the ACO&DE is obviously superior to basic ACO method for the VRP. Second, the ACO&DE is compared with basic DE. The ACO&DE algorithm can obtain significantly better performance than the DE in terms of ***Best*** and ***mean*** values. Moreover, the comparison is between the ACO and basic DE can be made. Two methods of ACO and DE can achieve similar results in terms of ***Best*** and ***mean*** values. More precisely, the proposed ACO&DE algorithm is effective to reduce the computational time gave much better results regarding the computational time.

Obviously, the proposed ACO&DE algorithm has much lower ***Best a***nd ***Mean*** distances compared with basic ACO and ED method, and the results demonstrates its effectiveness for solving VRP. From the above analysis, we can draw such a conclusion that proposed ACO&DE algorithm can perform better than ACO and ED in terms of ***best*** and ***mean*** values of the solutions for almost all instances. The average best value is 841.64 and the average mean value is 862.89. With regard to obtaining the optimal solution, ACO and DE algorithms can achieve similar results to ***Best_known*** in only two instances. The ACO algorithm can obtain the optimal solutions in A1 and A2 instances, whereas the DE algorithm can obtain in A2 and B1 instances. Moreover, the ACO&DE method provides more optimal solutions compared to the ACO and DE methods for all instances, and it can obtain the optimal solutions on small scale instances (A1-A3, B1-B2, C1) with 30-50 nodes. Despite, for six instances (C2-C5, C11-C12), ACO&DE algorithm can not obtain optimal solutions, the solutions obtained for these instances are close to the ***best_known*** values. With regard to obtaining the optimal solution, three algorithms can obtain the optimal solutions for A2 instance with 32 nodes. Fig. 6 shows the optimal solution using these algorithms and Table IV lists the optimal routes visited by the respective vehicles for A2 instance. The difference among the ACO, DE and ACO&DE algorithms is that the convergence speed is different. The convergence results are listed in Fig. 7. We can see that convergence speed of the ACO&DE is faster than the other two algorithms. ACO&DE algorithm is convergent within

TABLE III
THE COMPARISON OF ACO&DE WITH ACO AND DE ALGORITHMS.

No.	*Instance*	*Best_known*	ACO			DE			ACO&DE		
			Best	*Mean*	Time(s)	*Best*	*Mean*	Time(s)	*Best*	*Mean*	Time(s)
1	A1	784	784.00	786.27	7.18	792.15	794.24	6.31	784.00	784.00	6.22
2	A2	661	661.00	684.87	8.16	661.00	671.32	7.24	661.00	664.26	6.84
3	A3	742	756.23	763.26	8.52	750.54	758.43	8.04	742.00	746.65	7.35
4	B1	672	682.15	686.73	7.80	672.00	686.18	6.80	672.00	680.65	6.50
5	B2	788	795.24	818.36	8.75	792.37	816.58	8.12	788.00	796.47	7.66
6	C1	524.61	533.15	550.26	13.20	536.24	546.13	10.42	524.61	542.62	9.78
7	C2	835.26	860.26	883.13	17.37	854.63	878.25	15.28	841.38	876.34	13.24
8	C3	826.14	840.51	860.16	26.56	837.43	862.56	22.34	832.62	854.61	20.22
9	C4	1028.42	1073.00	1102.23	32.70	1078.51	1097.72	28.32	1048.33	1089.42	25.64
10	C5	1291.45	1350.24	1376.28	90.16	1336.62	1378.45	86.43	1314.24	1372.67	80.35
11	C11	1042.11	1073.00	1102.24	30.50	1078.53	1096.54	27.34	1056.26	1088.78	21.26
12	C12	819.56	846.00	864.65	28.73	843.32	867.63	25.52	835.25	858.25	20.54

TABLE IV
THE OPTIMAL ROUTES VISITED BY VEHICLES FOR A2 INSTRANCE.

Instance	Vehicle	Number of customers	Customers sequence of each route
A2	Route 1	6	15→17→9→3→16→29
	Route 2	8	12→5→26→7→8→13→32→2
	Route 3	6	20→4→27→25→30→10
	Route 4	4	23→28→18→22
	Route 5	8	24→6→19→14→21→1→31→11

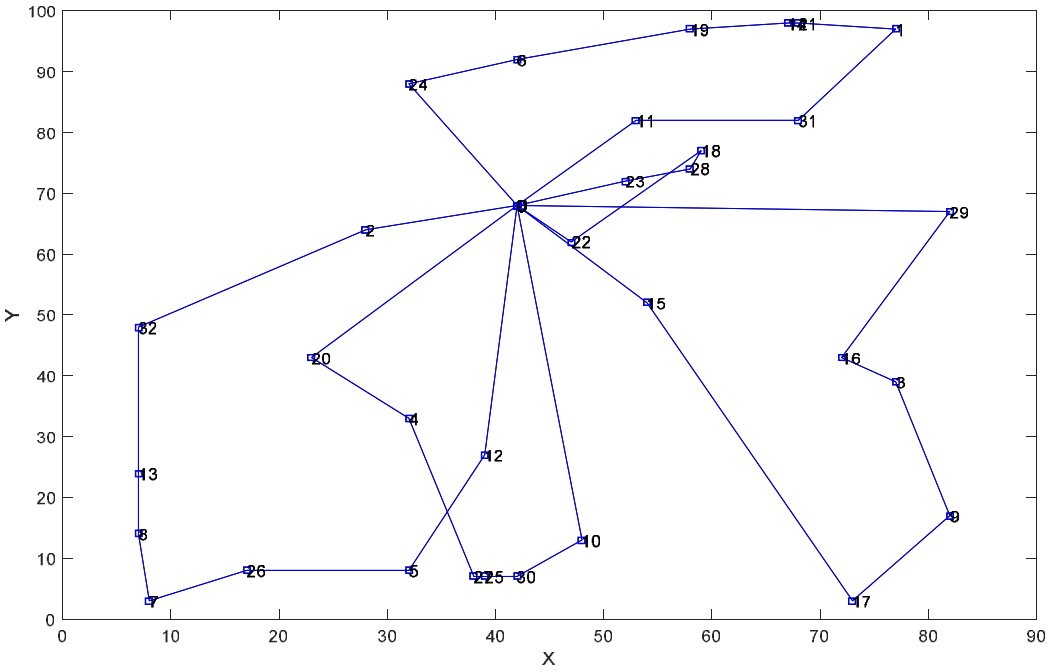

Fig. 6. The optimal routes for A2 instrance.

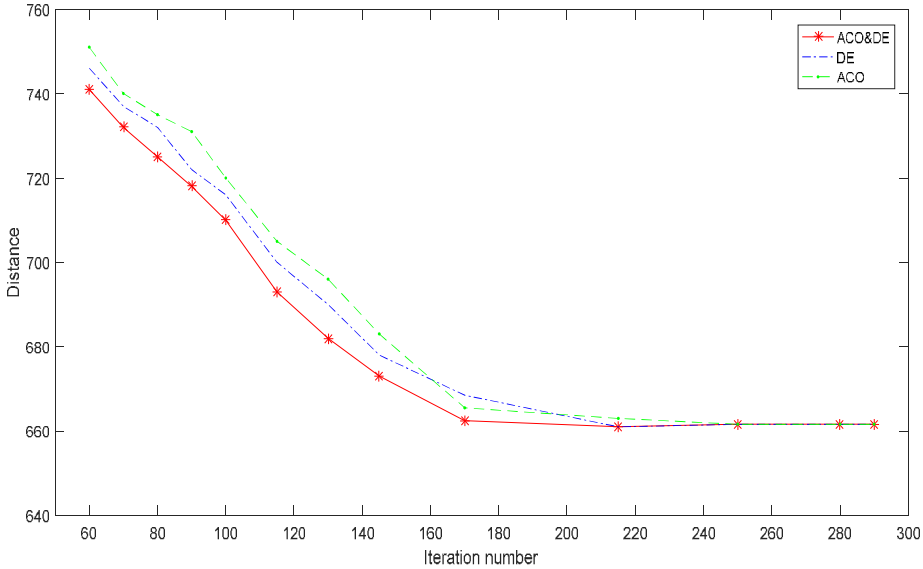

Fig. 7. The convergence results of three algorithms for A2 instance.

TABLE V
COMPARISON RESULTS OF DIFFERENT METHODS FOR THE VEHICLE ROUTING PROBLEM.

No.	*Best_known*	SA		GA		TS		ACO&DE	
		Best	*Gap*(%)	*Best*	*Gap*(%)	*Best*	*Gap*(%)	*Best*	*Gap*(%)
1	784	813.78	3.80	805.60	2.76	805.50	0.23	784.00	0
2	661	706.42	6.87	680.50	2.95	690.40	4.45	661.00	0
3	742	761.85	2.68	747.80	0.78	787.00	6.06	742.00	0
4	672	690.24	2.71	684.6	1.88	719.00	6.99	672.00	0
5	788	796.60	1.09	788.4	0.05	788.40	0.05	788.00	0
6	524.61	528	0.65	524.81	0.04	524	0.00	524.61	0
7	835.26	838	0.33	849.77	1.74	844	1.05	841.38	0.73
8	826.14	829	0.35	840.72	1.76	835	1.07	832.62	0.78
9	1028.42	1058	2.88	1055.85	2.67	1052	2.29	1048.33	1.93
10	1291.45	1376	6.55	1378.73	6.76	1354	4.84	1314.24	1.76
11	1042.11	1176	12.85	1060.24	1.74	1042.41	0.03	1056.26	1.35
12	819.56	826	0.79	877.8	7.11	819.76	0.02	835.25	1.91
	Average		3.46		2.52		2.26		0.71

180 iterations, while ACO and DE algorithms do not converge until after 220 iterations.

In order to further verify the performance of proposed ACO&DE algorithm, it is compared with other methods, including simulated annealing (SA) [6], genetic algorithm (GA) [7] and Tabu Search (TS) [6] for the VRP. The results are listed in Table V, in which the first column provides the description of instances, and the second column gives *Best_known* values that represent the shortest known solution distance reported in some literature. The next three columns give the best solutions and relative deviation using SA, GA, and TS. In the last column, ACO&DE is our proposed algorithm as described above. In Table V, *Best* refers to the best solutions, and *Gap* is the relative deviation between the best value of each algorithm and *Best_known*, i.e., $Gap=100*(f - f^*)/f^*$, in which f indicates the distance of the best solution, and f^* denotes *Best_known* value for each instance. The ACO&DE algorithm performs better than other methods. And in further comparisons to *Best_known* values, ACO&DE can perform well for the nine instances, and the result is very close to *Best_known* values during experiments. With regard to *Gap* values, the small the *Gap* value is, the higher the quality of the solution is. A zero gap means that the optimal solution can be found. From Table V, it's easy to show that the GA, SA and TS methods can get large *Gap* values, and the *Gap* value of a few instances is more than 5%. From Table V, among the 12 instance, the ACO&DE algorithm can obtain small *Gap* values, and the *Gap* value of 6 samples is equal to 0. The *Gap* value of 2 samples is less than 1, and the *Gap* value of only 4 instances is between 1 and 2. Generally, the average Gap of ACO&DE is found to be only 0.71% for all the instances tested, while the average Gap of TS is 3.46%, the average Gap of SA is 2.52%, and the average Gap of TS is 2.26%. Therefore, we can see that the ACO&DE algorithm has strong robustness and achieves excellent generalization performance.

VI. CONCLUSION

This article proposes ACO&DE algorithm based on the ACO and DE algorithm for the VRP in logistics transportation management system. The experiment results have shown that: firstly, the ACO&DE can make full use of advantages of the ACO and DE algorithms to make up for its own weakness, and it has achieved much improvement compared with basic ACO and DE algorithms. Secondly, to make the DE algorithm suitable for solving the VRP, three strategies of mutation operator, crossover operator and selection strategy have been redesigned to implement the discrete DE directly. In addition, the ACO&DE algorithm can obtain higher solutions than the other heuristics. Finally, because the DE algorithm can expand the search scope of the algorithm, it would effectively enhance the optimization performance of ACO algorithm. Thus, this methodology can be further extended in variants of the VRP and other logistics transportation fields. The limitation of this research is that the scale of instances is not large enough. Therefore, we will focus on a parallel version of the ACO&DE algorithm in the future, and thus this can cut down the calculation time greatly and enhance the performance of algorithms.

REFERENCES

[1] A. Pessoa, E. Uchoa and M. P. de Aragpo, "A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem," *Networks*, vol. 54, pp. 167-177, 2009.

[2] R.Baldacci, N.Christofides and A. Mingozzi, " An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts," *Mathematical Programming*, vol. 115, pp. 351-385, 2008.

[3] G. Clarke and J.W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, pp. 568-581, 1964.

[4] R. E. Taillard, " Parallel iterative search methods for vehicle routing problems, " *Networks*, vol. 23, pp. 661-673,1993.

[5] Y. Rochat and E. D. Taillard, "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of Heuristics*, vol.1, pp. 147-167, 1995.

[6] I. H. Osman, "Metastrategy simulated annealing and Tabu search algorithms for the vehicle routing problem," *Operations Research*, vol.41, pp. 421-451, 1993.

[7] B. M. Baker and M. A. Ayechew , "A genetic algorithm for the vehicle routing problem, " *Computers & Operations Research*, vol. 30, no. 5, pp. 787-800, 2003.

[8] A. M. Altabeeb, A. M.Mohsen and A. Ghallab, "An improved hybrid firefly algorithm for capacitated vehicle routing problem," *Applied Soft Computing* , vol. 84, pp. 1-9, 2019.

[9] C. A. Rego, "Subpath ejection method for the vehicle routing problem, " *Management science*, vol. 44, no.10, pp. 1447-1458, 1998.

[10] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 53-66, 1997.

[11] B. Bullnheimer, R. F. Hartl and C. Strauss, "An improved Ant System algorithm for the Vehicle Routing Problem," *Annals of Operations Research*, vol. 89, pp. 319-328, 1999

[12] R.Storn and K.Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of. Global Optimization*, vol. 11, pp. 341-359, 1997.

[13] Z. Q. Yu, X. X. Zhang and X. Shen, "An Improved Differential Evolutionary Algorithm Based on Simulated Annealing and Levy Flights Mechanism, " *Engineering Letters*, vol. 29, no.2, pp. 697-703, 2021.

[14] U. Maulik and I. Saha, "Modified differential evolution based fuzzy clustering for pixel classification in remote sensing imagery," *Pattern Recognition*, vol. 42, ppt. 2135-2149, 2009.

[15] M. Varadarajan and K. S. Swarup, "Differential evolution approach for optimal reactive power dispatch, " *Applied Soft Computing*, vol. 8, pp. 1549-1561, 2008.

[16] Q. K. Pan, M. F. Tasgetiren and Y.C. Liang, "A discrete differential evolution algorithm for the permutation flowshop scheduling problem, " *Computers and Industrial Engineering*, vol. 55, pp. 795-816, 2008.

[17] G. H. Zhang, K.Y. Xing and F. Cao, "Discrete differential evolution algorithm for distributed boocking flowshop scheduling with makespan criterion," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 96-107, 2018.

[18] I. M. Ali, D. Essam and K. Kasmarik, "A novel design of differential evolution for solving discrete traveling salesman problems," *Swarm and Evolutionary Computation*, vol. 52, pp. 1-17, 2020.