IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 18 August 2022, accepted 8 September 2022, date of publication 12 September 2022, date of current version 19 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3205756

RESEARCH ARTICLE

A Carnivorous Plant Algorithm With Heuristic Decoding Method for Traveling Salesman Problem

JIQUAN WANG, PANLI ZHANG, HONGYU ZHANG, HAOHAO SONG, JINLING BEI, WENFENG SUN, AND XIAOBO SUN
College of Engineering, Northeast Agricultural University, Harbin 150030, China

Corresponding authors: Wenfeng Sun (13304508203@163.com) and Xiaobo Sun (sunxiaobo@neau.edu.cn)

This work was supported by the National Social Science Found of China under Grant 21BGL174.

ABSTRACT The traveling salesman problem (TSP) is one of the most extensively studied problems in the combinatorial optimization area and still presents unsolved challenges due to its NP-hard attribute. Although many real-coded algorithms are available for TSP, they still have some performance challenges in the switch from continuous space to discrete space and perform at low convergence speed. This paper proposes a real-coded carnivorous plant algorithm with a heuristic decoding method (CPA-HDM) to solve the traveling salesman problem (TSP), which exhibits good convergence speed and solution accuracy. In this improved method, a new heuristic decoding method (HDM) is designed, which helps to map continuous variables to discrete ones without losing information, maintain population diversity, and enhance the solution quality after decoding. To balance the algorithm's search capability and enhance the probability of preferable individuals generated, an adaptive attraction probability (AAP), an improved growth model of carnivorous plants (IGMOCP), and a position update method of prey (IPUMOP) are developed. Aiming to reduce the probability of premature and prevent search stagnation, an improved reproduction strategy (IRS) and an adaptive combination perturbation are reconstructed. Finally, a local search algorithm is employed to improve the exploitation capability. To verify its validation, CPA-HDM is compared with six algorithms, for solving 28 TSP instances. The simulation results and statistical analyses demonstrate the superior performance of the proposed algorithm.

INDEX TERMS Real-coded carnivorous plant algorithm, traveling salesman problem, heuristic decoding method, adaptive combination perturbation.

I. INTRODUCTION

Optimization problems can be divided into continuous and combinatorial optimization problems, and mostly belong to combinatorial ones in the real world. As one of the classical discrete combinatorial optimization problems, the TSP, aims to find the shortest route by traveling m cities with minimum distance, each city travels exactly once and can be visited in any order, finally returning to the start city. The practical applications involve logistics distribution [1], AGV path planning [2], UAV route optimization [3], [4], production scheduling [5], [6], and drilling holes in printed circuit boards [7] can be ascribed to TSP. Generally, a minor enhancement of solutions or a reduction in execution time can economize millions of dollars or a significant increase in productivity for enterprises. Besides, TSP has been verified as an NP-hard problem [8], which means that with the city number growing, it is difficult to find an optimal or even suboptimal solution within a polynomial time. Hence, it is of considerable practical and scientific significance to study TSP, and it is still an active research direction in artificial intelligence.

Methods for solving TSP can be summarized into three categories: exact algorithms, heuristic algorithms, and

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

meta-heuristic algorithms. The exact algorithms such as dynamic programming [9], branch and bound [10], and integer linear programming [11] are not practical due to their exponential time cost. Heuristic algorithms such as nearest neighbor [12] and local search algorithm [13], [14] can acquire optimal or near-optimal value in an acceptable time when solving TSP with relative medium-scale, whereas it tends to get trapped into a local optimum.

The meta-heuristic algorithm is easy to implement, and only the information about the fitness function is needed in the optimization process. Exploration and exploitation ability play a significant role in the meta-heuristic algorithm, the former enables the algorithm to explore areas with better solutions in the search space and escape from local optimum, and the latter improves the possibility of achieving preferable solutions. The algorithm which combines these two abilities nicely can refrain itself from converging prematurely in the early stages and quickly converges to the global optimal at the end of optimization. In consequence, meta-heuristic algorithms perform better in TSP with less computation time compared with exact algorithm and heuristic algorithm, it can be summarized into three categories: (1) evolution-based which consists of the genetic algorithm [15], [16]; differential evolution [17]; (2) physics-based which consists of the water cycle algorithm [18], randomized gravitational emulation search algorithm [19]; (3) swarm intelligence-based which consists of ant colony optimization algorithm [20], particle swarm optimization [21], bat algorithm [22], grey wolf optimizer [23], etc.

CPA is a swarm intelligence-based optimization algorithm inspired by the survival skills of carnivorous plants, first proposed in 2021 [24], and has been proven to be effective and robust in addressing continuous problems and engineering design problems. Then Mukherjee and Roy [25] presented an improved binary CPA, which exhibits strong exploration and exploitation ability and address the optimal micro-PMU placement problem in the distribution network successfully. To the best of our knowledge, CPA in the existing literature applied to solve TSP has not been detected. The method for algorithms suitable for continuous problems to solve TSP can be classified into two categories: (1) the algorithm employs permutation-coded, and the new solutions, in this type, are generated without changing their discrete properties; (2) the algorithm introduces a decoding method to generate the legal solution. In the first class, Akhand *et al.* [26] and Khan *et al.* [27], [28] employed swap sequence and swap operator to keep the discrete format by swapping the positions of two genes, although the above algorithms do not need to be decoded, they play a minor role in the offspring quality improvement of its heuristic insufficiency and exhibit low convergence speed. The literature [18], [23], [29] employed hamming distance to redesign the individual generation operator according to the characteristics of TSP. Although these methods above can improve the solution quality and have strong search ability, they are prone to stick to local optimum and the algorithm design idea is changed. In the second class, the order-based arrangement [30], [31] and rounding method [32], [33] are commonly employed for decoding, which is easy to implement but plays a negative role in the solution quality of its randomicity. The limitations of the decoding method for TSPs, the lack of CPA in addressing combinatorial optimization problems, and the promising results achieved by CPA in continuous problems have severed as the main motivation of this paper.

Considering the above problems, a new decoding method HDM, which both considers the distance between cities and the continuous variables of individuals, is designed at first. For one thing, it can extract the outstanding features of parent individuals, for another, it can diversify the population. Thus, HDM can play a positive role in the convergence rate of the algorithm.

The main effort of the proposed algorithm is to improve the convergence rate and the search precision on TSP instances of different sizes. To achieve it, CPA has been further optimized. Firstly, in the existing literature, the growth of carnivorous plants or the update of prey depends on attraction probability, and attraction probability is a constant, which cannot well balance the exploration and exploitation ability. Thus, an AAP based on distance and city size is proposed, the distance between the carnivorous plant and prey determines whether the prey can be successfully attracted. Secondly, the attraction may approach 0 in the growth phase, resulting in a slow convergence speed. Then the CPA-HDM adds the guidance of the best individual in the growth model of carnivorous plants and the subgroup's best individual in the position update rule of prey, which can improve the quality of offspring. Thirdly, the reproduction phase only allows the optimal individual to reproduce, which may enhance the probability of the algorithm falling into the local optimum. Therefore, the IRS is proposed, and all carnivorous plants are allowed to reproduce, which is helpful to increase the information interaction among subgroups. Fourthly, the 2-opt exchange is generally used to detect a better solution, however, the edges involved in the standard 2-opt exchange are chosen at random, and the low probability of excellent individuals is generated, which causes unnecessary search times in the iteration. To address this problem, the neighborhood 2-opt and double-bridge exchange are presented in this paper. Finally, to improve the search accuracy, the 2-Opt algorithm is adopted. Thus, the proposed algorithm can both possess exploration and exploitation capabilities, which obtain high-quality solutions and high convergence speed.

The key contributions of this paper can be summarized as follows:

- A new decoding method HDM is designed to improve the solution quality and population diversity;
- The AAP is presented to balance the exploration and exploitation ability.
- The IGMOCP & IPUMOP are developed to extend the search space and increase the probability of producing better offspring.

- The IRS is proposed to reduce the probability of premature.
- An adaptive combination perturbation is added to prevent search stagnation.
- 28 instances are adopted to verify the validation of CPA-HDM, the experiment results and statistical analyses indicate that the proposed algorithm is performance superior to its competitors.

The remainder of this paper is done as follows: the knowledge of standard carnivorous plant algorithm and TSP are introduced in Section II; the details of CPA-HDM for solving TSP are presented in Section III; the experimental analyses are conducted in Section IV; the conclusion and the future work are proposed in Section V.

II. RELATED WORKS

This section introduces the meta-heuristics algorithms for solving TSP, the standard carnivorous plant algorithm, and the TSP. The survey of recently meta-heuristics for the TSP is briefly introduced in Section *A*. The details of the standard carnivorous plant algorithm are introduced in Section *B*. The TSP and its goal are described in Section *C*.

A. META-HEURISTIC ALGORITHMS FOR TSP

Many meta-heuristics algorithms have been proposed for solving TSP, which can be broadly divided into the meta-heuristic algorithms with decoding method and without decoding method for TSP.

1) ALGORITHMS WITHOUT DECODING METHOD FOR TSP

Various meta-heuristic algorithms adopt permutation-coded, which need to alter updating methods to keep the discrete properties of TSP. Osaba *et al.* [29] presented a discrete bat algorithm (DBA) with hamming distance, two well-known operators 2-Opt and 3-Opt are employed to improve the solution quality. Khan and Maiti [28] proposed a swap sequence based artificial bee colony algorithm (ABC) to update the solution without changing its discrete properties, then the 3-Opt operator is introduced to improve the stagnant solution in the scout bee phase, and at the end of the search process. Wang *et al.* [34] proposed a discrete symbiotic organism search with excellent coefficient and self-escape (ECSDSOS) by the new calculation method of position update rules. The excellent coefficient strategy helps to enhance the exploitation capability and the self-escape strategy helps to keep population diversity. Akhand *et al.* [26] adopted a discrete spider monkey optimization (DSMO). In DSMO, all the spider monkey was represented as TSP solution. To find a better individual, the swap sequence and swap operator were employed to make interaction among monkeys. A discrete water cycle algorithm (DWCA) was proposed by Eneko *et al.* [18]. Three strategies are employed to improve the performance of the algorithm, which are: (1) the Hamming distance is introduced to measure the difference between two individuals; (2) the insert mutation operator is adopted to emulate the evaporation and raining process in the discrete solution space; (3) an adaptive modification parameter is proposed to choose movement operators. Kóczy *et al.* [35] presented a discrete bacterial memetic evolutionary algorithm (DBMEA) with the local search algorithm, which employed the nearest neighbor, secondary nearest neighbor, alternating nearest neighbor heuristic, and random creation method to generate the initial population, combined gene transfer operation to improve the solution quality. Zhang *et al.* [36] presented a whale optimization algorithm with several effective components. The Gaussian disturbance helps to maintain population diversity, and the variable neighborhood search strategy helps to improve the solution quality. Panwar and Deep [23] presented a discrete grey wolf optimizer (DGWO) by introducing the 2-Opt operator and hamming distance in the grey wolf optimizer. Wu *et al.* [37] designed a new sparrow search algorithm with a greedy algorithm. In this method, several components are employed to enhance the performance of the algorithm. First, the greedy algorithm helps to keep population diversity. Second, a sine and cosine search strategy is employed to update the solution. Finally, the genetic operators are employed to balance the search capability. Saji and Barkatou [38] presented a discrete bat algorithm with Lévy Flight (DBAL) by improving the velocity updating formula of the bat algorithm. To avoid trapping into a local optimal and enhance the population diversity, the improved uniform crossover operator and neighborhood search are employed to solve TSP. Gunduz and Aslan [39] reconstructed a discrete Jaya algorithm (DJAYA) with different swap, shift, and symmetry exchange operator combinations, and finally, the 2-Opt algorithm is employed to enhance the quality of the optimal individual in the population. Huang *et al.* [40] proposed a discrete shuffled frog-leaping algorithm (DSFLA). In DSFLA, four strategies are incorporated to enhance its performance. First, an improved roulette selection is proposed to maintain the population diversity; Second, the independent set is proposed to increase the exploration ability; Third, the local optimum mutation operator is presented to reduce the probability of stagnating; Finally, the local search algorithm is employed to improve the solution accuracy.

Some algorithms employ mathematical formulas suitable for the continuous problem and need to adopt the decoding method to generate legal TSP paths. Ezugwu and Adewumi [33] adopted the rounding method and restructured symbiotic organisms search by incorporating swap, insert, and inverse operators to form a discrete symbiotic organisms search (DSOS). Three mutation operators are employed to improve its initial population. Zhang and Han [30] applied the order-based arrangement to map continuous variables as discrete ones in the discrete sparrow search algorithm (DSSA), the roulette wheel selection, Gaussian mutation, and swap operator are introduced in DSSA to increase the probability of jumping out of the local optimum, the 2-Opt algorithm is adopted to enhance the solution quality.

2) ALGORITHMS WITH DECODING METHOD FOR TSP

Several heuristic decoding methods [41] [42] have been proposed to tackle job shop problems, however, the decoding methods commonly used in TSP lack heuristic. Some algorithms employ mathematical formulas suitable for the continuous problem and need to adopt the decoding method to generate legal TSP paths. Ezugwu *et al.* [32] adopted the rounding method and restructured symbiotic organisms search by incorporating swap, insert, and inverse operators to form a discrete symbiotic organisms search (DSOS). Three mutation operators are employed to improve its initial population. Osaba *et al.* [29] applied the order-based arrangement to map continuous variables as discrete ones in the discrete sparrow search algorithm (DSSA), the roulette wheel selection, Gaussian mutation, and swap operator are introduced in DSSA to increase the probability of jumping out of the local optimum, the 2-Opt algorithm is adopted to enhance the solution quality.

Meta-heuristic algorithms with the real-coded need to introduce a decoding method to map continuous variables of solutions to discrete legal solutions when solving TSP. Samanlioglu *et al.* [43] proposed a random-key genetic algorithm (RKGA) with the ranked-order value decoding to solve multi-objective TSP. RKGA combined the 2-Opt algorithm to improve the solution quality. Ezugwu *et al.* [32] proposed a simulated annealing-based symbiotic organisms search optimization algorithm (SOS-SA) with the rounding method. The SA in SOS-SA can help to reduce the probability of getting stuck into the local minimum and increase the population diversity. Ali *et al.* [44] presented a novel discrete differential evolution (NDDE) with the ranked-order value and best-matched value decoding method, and several strategies are incorporated to enhance the performance of the NDDE. Among them, the k-means clustering is used to improve the quality of the initial population, and a combined mutation is introduced to heighten the exploration ability. Finally, the 3-Opt and double-bridge are employed to enhance the exploitation ability. Kanna *et al.* [45] constructed a new hybrid algorithm named earthworm-based deer hunting optimization algorithm (EW-DHOA) to address large-scale TSP.

B. STANDARD CARNIVOROUS PLANT ALGORITHM

Unlike most plants that absorb nutrients through photosynthesis, carnivorous plants are autotrophic plants that capture and digest animals to obtain nutrients. They usually grow in harsh environments lacking nutrients, trapping insects, frogs, small lizards, birds, and other small animals through color or secretions to supplement additional nutrients such as nitrogen and phosphorus needed for growth and reproduction. The CPA is a meta-heuristic algorithm proposed by imitating the whole predation process of carnivorous plants attracting, preying, and digesting. Four stages are comprised in the algorithm, which are classification and grouping, growth, reproduction, and recombination. The details are described as follows.

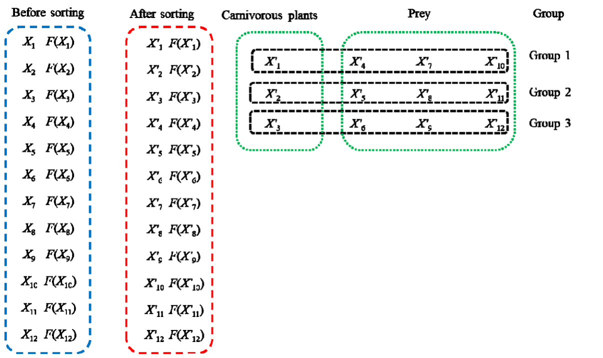


FIGURE 1. The grouping process at size 12 in CPA.

1) CLASSIFICATION AND GROUPING

The population size of CPA is nn, individuals in the population are sorted from the smallest to largest according to their fitness values for the minimization problem, the best n individuals are regarded as carnivorous plants, and the remaining n_1 individuals are regarded as prey ($n_1 > n$, n_1 is divisible by n). The group number is n, individuals in each group are comprised of one carnivorous plant and n_1/n prey. The best prey is attracted by the best carnivorous plant, the second-best prey is attracted by the second-best carnivorous plant, the process is repeated, and the n^{th} best prey is attracted by the n^{th} best carnivorous plant. It is noted that the $(n+1)^{\text{th}}$ best prey is attracted by the best carnivorous plant, and the $(n+2)^{\text{th}}$ prey is attracted by the second-best carnivorous plant, and the process is repeated until the n_1^{th} prey is attracted by the n^{th} carnivorous plant.

The grouping process is depicted by an example in Fig. 1, where population size $nn = 12$, the number of carnivorous plants $n = 3$, the number of prey $n_1 = 9$, $X = (X_1, X_2, \ldots, X_{12})$ before sorting and it becomes $X' = (X'_1, X'_2, \ldots, X'_{12})$ after sorting, the objective function values satisfied $F(X'_1) \leq F(X'_2) \leq \cdots \leq F(X'_{12})$.

2) GROWTH PHASE

The carnivorous plant lured prey by its scent, but prey may successfully escape from the plants or not be attracted. Hence, an attraction probability γ is introduced in CPA, if γ ($\gamma = 0.8$) is greater than a random number λ (λ is generated in the range [0,1]), the carnivorous plant successfully lures the prey to growth, and the model can be formulated as:

$$newxp_i = p_{iv} + \alpha \otimes (xp_i - p_{iv}) \tag{1}$$

$$\alpha = gr * rand \tag{2}$$

where $\otimes$ represents multiplying the variables at the same position in two vectors, xp_i is the carnivorous plant in group i, p_{iv} is the v^{th} prey in group i, $rand$ is the random vector in the range [0,1], gr is the growth rate, usually equals to 2.

If γ is less than λ, which stands for the prey escapes from the trap or not being attracted by the plant and the growth model of prey can be expressed as:

$$newp_{ij} = p_{iv} + \alpha \otimes (p_{iu} - p_{iv}) \tag{3}$$

$$\alpha = \begin{cases} gr * rand & f(p_{iu}) < f(p_{iv}) \\ 1 - gr * rand & f(p_{iv}) < f(p_{iu}) \end{cases} \tag{4}$$

where $\otimes$ represents multiplying the variables at the same position in two vectors, p_{iu} and p_{iv} are the u^{th} and v^{th} prey in group i, respectively. $rand$ is the random vector in the range [0, 1], $f(p_{iv})$ and $f(p_{iu})$ are the fitness value of the v^{th} and u^{th} prey in group i, respectively.

3) REPRODUCTION PHASE

The best carnivorous plant is allowed to perform the reproduction operation, and the mathematical model is summarized as:

$$newxp_i = \begin{cases} xp_1 + \beta \otimes (xp_j - xp_i) & f(xp_j) < f(xp_i) \\ xp_1 + \beta \otimes (xp_i - xp_j) & f(xp_i) < f(xp_j) \end{cases} \tag{5}$$

$$\beta = \mu * rand \tag{6}$$

where $\otimes$ represents multiplying the variables at the same position in two vectors, xp_1 is the best individual in the population, xp_i, xp_j are the carnivorous plant in group i and group j, respectively, $rand$ is the random vector in the range [0, 1], μ is the reproductive rate, usually equals to 1.8, $f(xp_i)$ and $f(xp_j)$ are the fitness value of the carnivorous plant in group i and group j, respectively.

4) RECOMBINATION PHASE

First, the newly generated individuals and the previous population are combined into a new population. Second, individuals in the new population were ranked in order of fitness value from small to large. Finally, select nn best individuals to maintain the same population size as the previous population. This process is called recombination, which ensures that fitter individuals can be selected for the later generation.

The pseudo-code of standard CPA is presented in **Algorithm 1**.

Algorithm 1 CPA

Input: the population size nn; the population size of carnivorous plants xp: n, the population size of prey p: n_1, growth_rate, reproduction_rate, Maximum iteration: *Maxgen*;

Output: Best solution and the optimal value;

Generate nn initial individuals in the population;
Calculate the fitness value and sort based on the fitness value;
While $gen < Maxgen$
  Set n best individuals as carnivorous plants, the remaining n_1 individuals as prey, and sort a group as depicted in Fig.1;
  Newxp, *Newp* is updated with (1) and (3);
  Newxp is updated with (5);
  Newxp, *Newp*, *xp*, and *p* combined a new population;
  Calculate the fitness of the population;
  Sort according to the fitness value and select nn best individuals;
End While

C. THE TRAVELING SALESMAN PROBLEM

TSP is usually described as a merchant who traverses m cities to sell goods. In this process, one must pass through all the cities, each city can only pass through once and finally return to the original city. TSP can be represented as a weighted graph $G = (V, E)$, which goal is to find a Hamilton loop with the smallest weights. V is the set of vertices, and E is the set of edges. The vertices of the graph represent cities, the edges denote the path between cities, and the weight of an edge indicates the Euclidean distance between two cities. Although the definition of TSP is simple, as the number of cities increases, the number of possible tours increases dramatically. The challenge is to solve this problem in an acceptable time with the lowest travel costs. Until now, there is still no effective way to solve this problem, and it can be divided into symmetric and asymmetric TSP. If the distance from city i to city j equals from j to i, it is considered a symmetric problem, otherwise, an asymmetric problem. Mathematically, the problem with m cities can be expressed as:

$$\text{Min} f(D) = \sum_{i=1}^{m-1} d(t_i, t_{i+1}) + d(t_m, t_1) \tag{7}$$

where $f(D)$ is the distance traveled by the merchant in TSP, t_i denotes i^{th} city, $d(t_i, t_{i+1})$ represents the distance between the i^{th} city and $(i+1)^{\text{th}}$ city, which is calculated as

$$d(t_i, t_{i+1}) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{8}$$

where (x_i, y_i) and (x_{i+1}, y_{i+1}) are the coordinate of the i^{th} city and $(i+1)^{\text{th}}$ city.

III. THE PROPOSED ALGORITHM CPA-HDM FOR SOLVING TSP

The CPA divides the population into several subgroups according to the fitness of individuals. In each subgroup, carnivorous plants and prey are applied to guide the population to explore the solution space in various directions, so that the algorithm has strong global searchability. The optimal individual is allowed to reproduce in the reproduction stage, which helps the convergence speed of the algorithm. However, CPA is real-coded, and the solutions may with decimal and repetitions, which are infeasible for solutions of TSP. Therefore, it is necessary to find a suitable decoding method to map continuous variables as legal TSP paths. In addition, to enhance the performance of CPA, several improvements are proposed to effectively balance the exploration and exploitation capabilities, and further improve the convergence speed and solution quality. These improvements will be discussed in the following subsections.

A. INITIALIZATION

Let the initial population $X^0 = (X_1^0, X_2^0, \ldots, X_i^0, \ldots, X_{nn}^0)$, $X_i^0 = (x_1, x_2, \ldots, x_m)$, and the upper and lower bounds of variable values are b and a, respectively, where nn is the

TABLE 1. The distance between five cities.

City	1	2	3	4	5
1	∞	9	12	1	7
2	9	∞	7	4	5
3	12	7	∞	3	1
4	1	4	3	∞	7
5	7	5	1	7	∞

TABLE 2. The example of HDM.

City number	City 1	City 2	City 3	City 4	City 5
X_k	1.3575	4.5155	3.4863	0.7845	0.0637
Tp	10.4861	-	13.0701	3.5429	1.2619
2					
Tp	8.1559	-	1.8672	6.2000	-
Tp	13.9814	-	-	2.6572	-
Order	5	1	3	4	2

population size, m is the city size, $a = 0, b = m$. The variables can be randomly and uniformly generated between a and b. Therefore, the i^{th} individual X_i^0 in the initial population can be initialized as

$$X_i^0 = m * rand(1, m) \quad i = 1, 2, \ldots, nn \tag{9}$$

where $rand$ (1, m) is an m dimensional random vector in the range [0, 1].

B. HEURISTIC DECODING METHOD

The decoding method plays a vital role in the solution quality and the population diversity of the algorithm with real-coded on solving TSP. The commonly used decoding methods include order-based arrangement and rounding [30], [32]. For the former, the rank of each continuous variable of an individual represents an index of a city, a legal path is obtained according to the sorting result. The latter is to round the corresponding real value of each individual and processed the repetition integer to make a feasible solution. However, the above two methods only consider the continuous variables, and the distance between cities is ignored, which may play a negative role in the solution quality after decoding for its randomicity.

Based on the above problems, a new decoding method that both considers the distance between cities and continuous variables is designed, which is heuristic and beneficial to keeping population diversity. This method is suitable for real-coded meta-heuristics and has broad applicability. The TSP with population size nn, city size m, and the k^{th} individual $X_k = (x_1, x_2, \ldots, x_m)$ are set as an example, the specific steps are as follows:

Step 1: A uniformly distributed random integer μ is generated between $[1, m]$, μ is set as a home city;

Step 2: The μ^{th} variable value of X_k is removed, which is recorded as X_k', $X_k' = (x_1, x_2, \ldots, x_{\mu-1}, x_{\mu+1}, \ldots, x_m)$, the distance between city μ and the rest of the cities are found, which is recorded as d, $d = (d_{\mu,1}, d_{\mu,2}, \ldots, d_{\mu,\mu-1}, d_{\mu,\mu+1}, \ldots, d_{\mu,m})$. Then, Tp is calculated according to (10).

$$Tp = d \otimes (X_k')^{0.5} \tag{10}$$

where $\otimes$ represents multiplying the variables at the same position of vector d and vector $(X_k')^{0.5}$;

Step 3: The minimum value in the vector Tp is determined, and take the city i corresponding to the minimum value as the next city to be visited;

Step 4: Let $\mu = i$. Step 2 and Step 3 are repeated until the order of visits for all cities is determined.

To facilitate an understanding of HDM, city size $m = 5$, $\mu = 2$, and $X_k = (1.3575, 4.5155, 3.4863, 0.7845, 0.0637)$ are set as an example, the distance between each city is defined in Table 1. The main steps of the HDM are shown in Table 2.

Table 2 shows that when $\mu = 2$, $X_k = (1.3575, 4.5155, 3.4863, 0.7845, 0.0637)$, and the route after HDM is $2 \rightarrow 5 \rightarrow 3 \rightarrow 4 \rightarrow 1$.

It can be seen from (10) that Tp is both related to the distance between cities and continuous variables of individuals. For TSP, if the distance between the city i ($i = 1, 2, \ldots, m-1$) and city $i+1$ is small, the probability that the route could be small will be increased, and the decoding integrates with the greedy idea of the nearest neighbor, which helps to enhance the solution quality. However, it may result in an increment in the probability of premature convergence, thus the continuous variables are incorporated in decoding. Therefore, the HDM applies in TSP helps to improve the solution quality and maintain population diversity.

C. GROWTH PHASE

The attraction probability γ of prey by carnivorous plants is constant at 0.8 in CPA. However, the concentration of scent released by carnivorous plants decreases with distance. Therefore, the closer the distance between prey and carnivorous plant, the greater the attraction and vice versa.

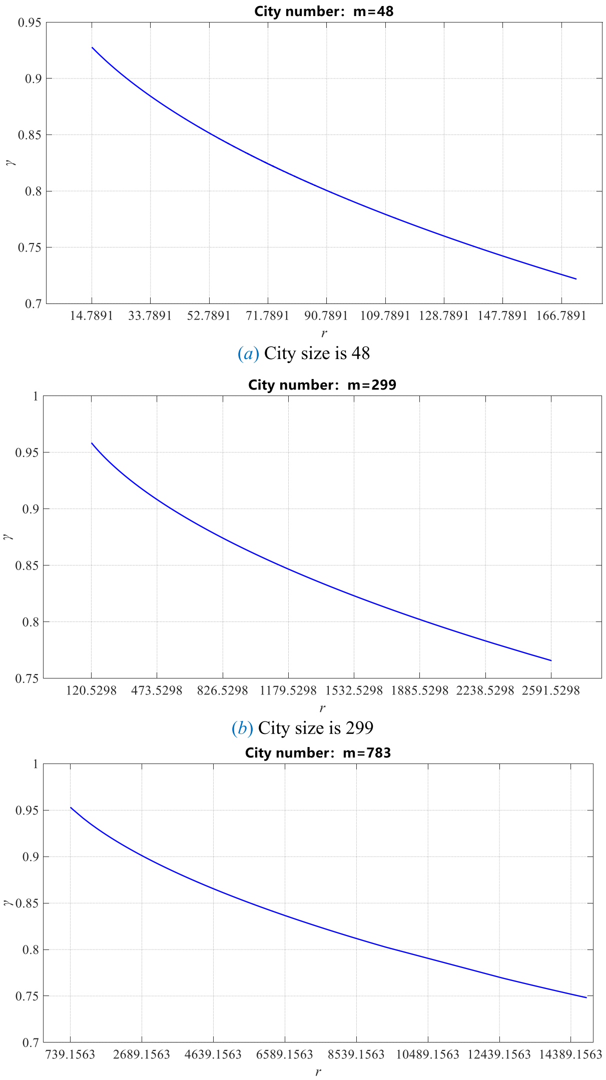
Thus, an adaptive attraction probability γ based on distance and number of cities is proposed, which is calculated as:

$$\gamma = e^{\frac{-r^{0.6}}{1.4*m}} \tag{11}$$

where m is the city number, r is the distance between xp_i and p_{iv}, r is calculated as

$$r = \|xp_i - p_{iv}\| = \sqrt{\sum_{j=1}^{m} (xp_i^j - p_{iv}^j)^2} \quad i = 1, 2, \cdots, n; \; v = 1, 2, \cdots n_1/n \tag{12}$$

where n is the number of carnivorous plants, n_1 is the number of prey, xp_i is the carnivorous plant in group i, p_{iv} is the v^{th} prey in group i, xp_i^j and p_{iv}^j are the j^{th} components of xp_i and p_{iv}, respectively.



(a) City size is 48

(b) City size is 299

(c) City size is 783

FIGURE 2. The trend of the attraction probability γ with distance r.

For a clearer understanding of AAP, $m = 48$, $m = 299$, and $m = 783$ are set as an instance, and the trend of γ with r is depicted in Fig. 2.

It can be observed in Fig. 2 that the farther the distance between carnivorous plants and prey, the less γ, the greater probability of prey performing position updating, the more search directions for the population, and the strong exploration ability of the algorithm; the closer prey to carnivorous plant, the more γ, the greater probability of carnivorous plant to grow, and the strong exploitation ability. Since the initial population is randomly generated and evenly distributed in the solution space, the distance between individuals is relatively far in the early iterations and close in the later iterations. Therefore, the prey is selected to update with high probability and the exploration ability of the CPA is strong in the early stage, the carnivorous plant is selected to grow with high probability and the exploitation ability is strong in the late stage.

The value of α is related to the search space of CPA, when α is close to 0, the xp_i - p_{iv} and p_{iu} - p_{iv} do not work at all, and the global search ability of the algorithm becomes weak. To address the above problems, the IGMOCP & IPUMOP are proposed as follows:

$$newxp_i = p_{iv} + \alpha \otimes (xp_i - p_{iv}) + \sigma \otimes (xp_1 - p_{iu}) \tag{13}$$

$$newp_i = \begin{cases} p_{iu} + \alpha \otimes (p_{iv} - p_{iu}) + \sigma \otimes (xp_i - p_{iw}) \\ \qquad\qquad if\ f(p_{iv}) < f(p_{iu}) \\ p_{iv} + \alpha \otimes (p_{iu} - p_{iv}) + \sigma \otimes (xp_i - p_{iw}) \\ \qquad\qquad if\ f(p_{iu}) < f(p_{iv}) \end{cases} \tag{14}$$

$$\alpha = \frac{m * gr * rand(1, m)}{r} \tag{15}$$

$$\sigma = (1 - (\frac{t}{t_{\max}})^{0.8}) * rand1(1, m) \tag{16}$$

where $\otimes$ represents multiplying the variables at the same position in two vectors, xp_i is the carnivorous plant in group i, xp_1 is the best individual in the population, p_{iv}, p_{iu}, p_{iw} are the v^{th}, u^{th}, w^{th} prey in group i, respectively. $rand\,(1, m)$ is an m dimensional random vector in the range [0.2, 1], $rand1(1, m)$ is an m dimensional random vector in the range $[-1, 1]$, gr is the growth rate, m is the city size, r in (13) is the distance between xp_i and p_{iv}, r in (14) is the distance between p_{iv} and p_{iu}, t_{max} is the maximum runtime, t is the current runtime.

The implementation method of (13) and (14) is: 1) λ is randomly and uniformly distributed generated in the range [0, 1]; 2) if $\lambda \leq \gamma$, the (13) is selected as the carnivorous plant growth method; else the (14) is selected as the prey position update method.

It can be observed from (13) that the attractiveness of the optimal carnivorous plant to the prey is added in the carnivorous plant growth model, which helps to make the prey move to the potential direction of the search space, improve the probability of the excellent offspring, and strengthen the exploitation ability. From (14), it can be known that the attractiveness of the carnivorous plant to the prey in the same group is added in the prey position update method, which not only helps to enhance the exploration ability but also improves the probability of the excellent offspring generated.

An optimization problem in 2 dimensions is taken as an example to compare the difference between the basic and improved update methods. Suppose the best carnivorous plant $xp_1 = (1.7, 2)^T$, the carnivorous plant in i^{th} group $xp_i = (1, 1)^T$, the prey in i^{th} group $p_{iu} = (0.2, 0.2)^T$, $p_{iv} = (2.2, 0.5)^T$, and $p_{iw} = (0.1, 0.1)^T$, 1000 numbers of α in (1) and (3) are randomly generated, 1000 numbers of α and σ in (13) and (14) are generated according to (15) and (16), respectively. The individuals' distribution in the search space obtained according to (1) and (13) is shown in Fig. 3(*a*) and Fig. 3(*b*), respectively. The individuals' distribution in the search space obtained according to (3) and (14) is shown in Fig. 4(*a*) and Fig. 4(*b*), respectively.

As is depicted in Fig. 3, the range of the abscissa of the offspring produced by (1) is $[-0.2, 2.2]$, and the range of the ordinate is [0.5, 1.5], all newly generated individuals are far away from the best carnivorous plant. However, the range

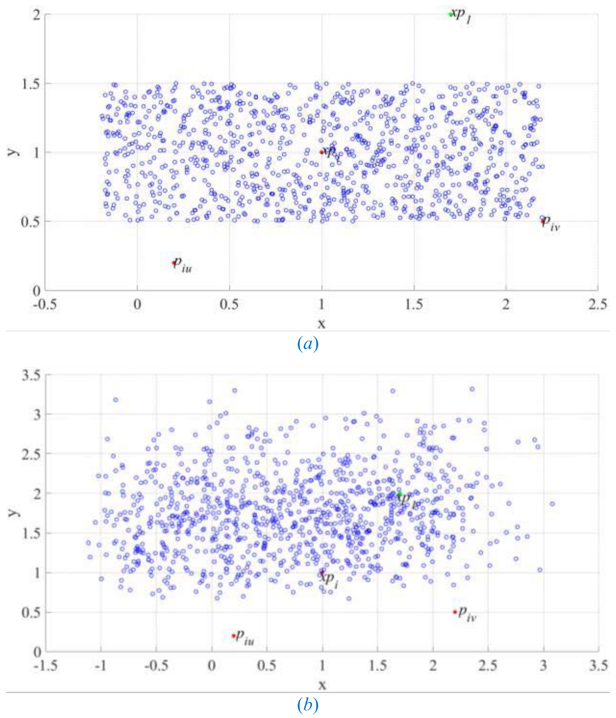


FIGURE 3. The individuals' distribution in the search space by the carnivorous plant growth model.

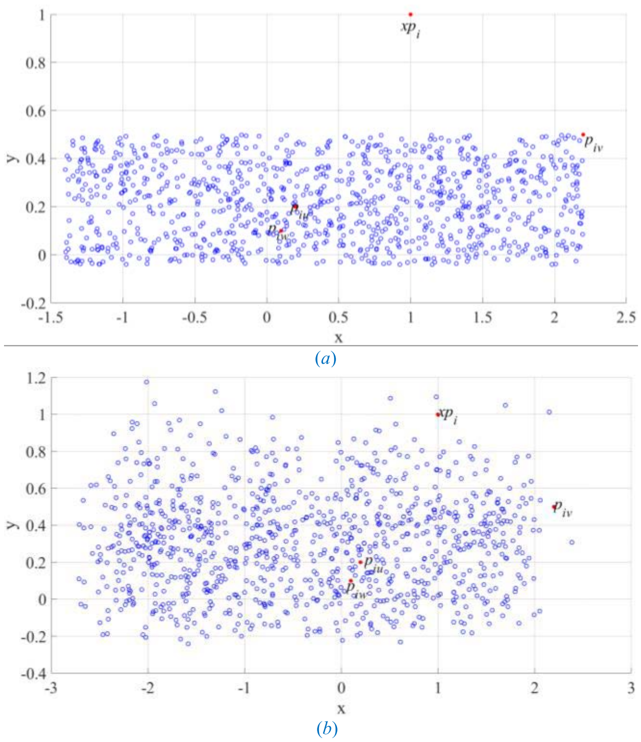


FIGURE 4. The individuals' distribution in the search space by the prey position update.

of the abscissa of the offspring produced by (13) is [-1.1, 3.1], and the range of the ordinate is [0.6, 3.4]. Compared with Fig. 3(a), the number of individuals near the optimal individual is increased in Fig. 3(b). In Fig. 4, the range of the abscissa of the offspring produced by (3) is [-1.4, 2.2], the range of the ordinate is [-0.05, 0.45], and all newly generated individuals are far away from the carnivorous plant in the i^{th} group. However, the range of the abscissa of the offspring produced by (14) is [-2.7, 2.3], and the range of the ordinate is [-0.21, 1.2]. Compared with Fig. 4(a), the number of individuals near the carnivorous plant is increased in Fig. 4(b). The analyses above show that the number of outstanding offspring increases and the search space is expanded with the IGMOCP & IPUMOP.

D. REPRODUCTION PHASE

Every carnivorous plant can prey and absorb nutrients for growth and reproduction in real-life. However, CPA only allowed the best carnivorous plant to reproduce, which is inconsistent with the law in nature. Besides, the range of β in CPA is [0, 1.8], when β is close to 0, the $xp_i - xp_j$ does not work at all, then the reproduction is difficult to generate excellent offspring, and the algorithm is prone to stick in the local optimal. In response to the above problems, the reproduction strategy is improved as each carnivorous plant is allowed to reproduce, and the reproduction of the best carnivorous plant is different from that of other carnivorous plants. The IRS is as follows

$$newxp_i = \begin{cases} xp_1 + \beta \otimes (xp_j - xp_i) & f(xp_j) < f(xp_i) \\ xp_1 + \beta \otimes (xp_i - xp_j) & f(xp_i) < f(xp_j) \end{cases} \tag{17}$$

$$newxp_i = xp_i + \beta \otimes (xp_1 - xp_j) \tag{18}$$

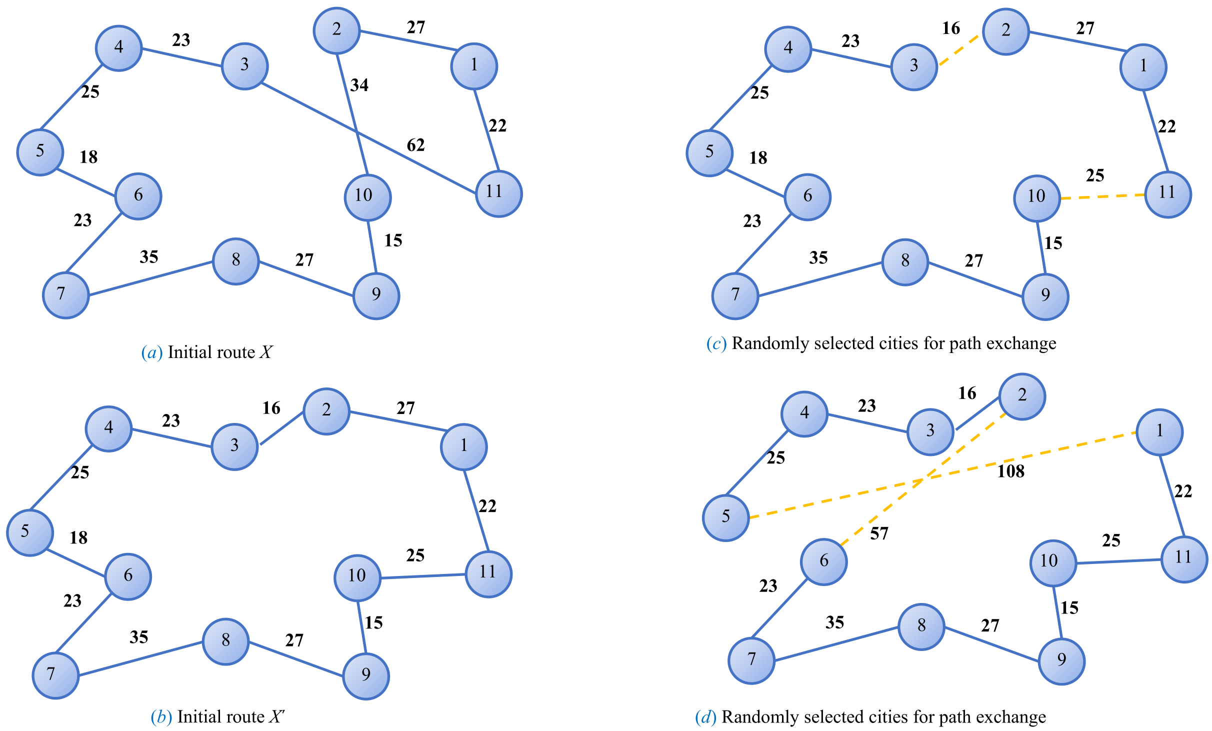
where $\otimes$ represents multiplying the variables at the same position in two vectors, xp_i and xp_j are the carnivorous plants in groups i and j, respectively, xp_1 is the best individual in the population, β is an m dimensional random vector in the range [0.5, 1.8].

The implementation method of IRS is: 1) δ is randomly and uniformly distributed generated in the range [0, 1]; 2) if $\delta \leq 0.6$, the (17) is selected as the reproduction method; else the (18) is selected.

The analysis of IRS shows that (17) has more exploitation ability than (18), when the reproduction method with strong exploitation capability is executed, the CPA can effectively balance the exploration and exploitation abilities. When the reproduction method with exploration capability is executed, the CPA can decrease the probability of falling into the local optimum.

E. ADAPTIVE COMBINATION PERTURBATION STRATEGY

The neighborhood 2-opt exchange and double-bridge exchange are employed as perturbation methods in this paper to find better individuals around nn^*rr best individuals locally (nn is the population size and rr is the selection ratio), and the local search algorithm 2-Opt is adopted to improve the quality of neighborhood solutions, if the new individual is better, it will replace the original solution. To avoid expensive computing, the maximum number of neighborhood solutions is limited to 10 in this paper, and the nn^*rr best individuals



Note: The city number is in the circle, the number on the line connecting the two circles is the distance between the two cities, and the yellow dotted line represents the path after reconnection.

FIGURE 5. **2-opt exchange.**

performing the perturbation are re-selected from the population in every I iterations.

1) NEIGHBORHOOD 2-OPT EXCHANGE

The idea of the 2-opt exchange [46] is to delete two non-adjacent edges randomly and connect the other two edges formed by the four points corresponding to the deleted edges, which can enhance population diversity. The 2-opt exchange is depicted in Fig. 5, where the initial path in Fig. 5(a) is $X = (1, 2, 10, 9, 8, 7, 6, 5, 4, 3, 11,1)$, in Fig. 5($b$) is $X' = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,1)$.

Suppose that the randomly selected cities in X are 2, 3, 10, and 11, the new route of X after reconnection is shown in Fig. 5(c), where $d(2, 3) + d(10, 11)$ is 41, $d(2, 3)$ represents the Euclidean distance between city 2 and city 3. $d(2, 10)$ $+d(3, 11)$ in X is 96, it can be noticed that the path length after the 2-opt exchange is better than route X.

However, suppose that the cities selected in X' are 1, 2, 5, and 6, the new route of X' after reconnection is shown in Fig. 5(d), where $d(1, 5) + d(2, 6)$ is 165, $d(1, 2) + d(5, 6)$ in X' is 45, the new route of X' after reconnection is worse than the initial route. The edges involved in the standard 2-opt exchange are chosen at random, and the low probability of excellent individuals is generated, which causes unnecessary search times in iteration. A neighborhood 2-opt exchange is employed in this paper, and $X = (x_1, x_2, \ldots, x_m)$ is set as an example to illustrate this operation, where m is the city size. The main steps are as follows:

Step 1: City a is randomly selected from m cities, a is the center of the circle with radius r_1 as the neighborhood, which is recorded as $U(a, r_1)$. The calculation of r_1 is shown in (19).

$$r_1 = \frac{2.5 * Z}{m} \tag{19}$$

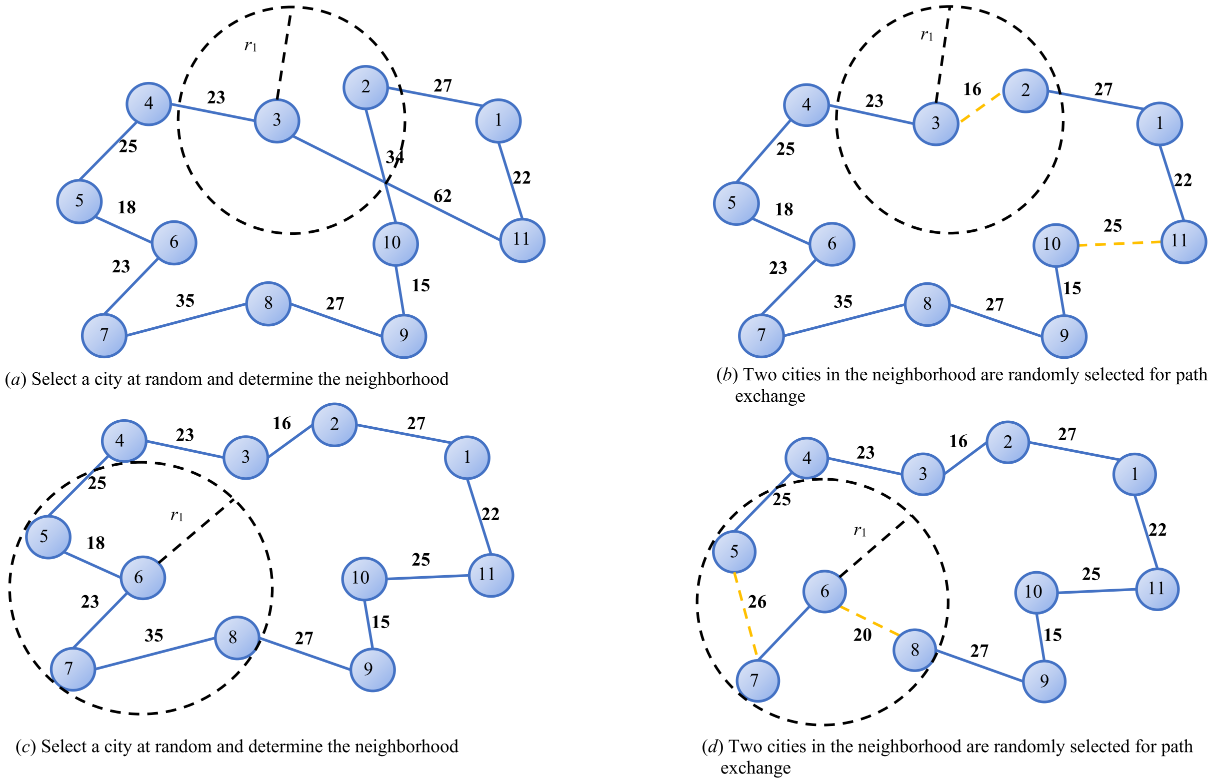
where Z is the path length of the best individual in the population.

Step 2: The number of cities u contained in neighborhood $U(a, r_1)$ needs to be determined. If $u \geq 2$, aa_1 and aa_2 are randomly selected in the neighborhood $U(a, r_1)$; if $u = 1$, let $aa_1 = a$, and aa_2 is randomly selected in $[1, 2, \ldots, a-1, a+1, \ldots, m]$;

Step 3: The city bb_1 is found adjacent to aa_1 in X, the city bb_2 adjacent to aa_2, and $bb_1 \neq aa_2, bb_2 \neq aa_1$. If $bb_1 = bb_2$, Step 1 and Step 2 are repeated until $bb_1 \neq bb_2$;

Step 4: Delete edges (aa_1, bb_1) and (aa_2, bb_2), connect edges (bb_1, aa_2) and (aa_1, bb_2).

The neighborhood 2-opt exchange is depicted in Fig. 6, where the initial path in Fig. 6(a) is $X = (1, 2, 10, 9, 8, 7, 6, 5, 4, 3, 11,1)$, in Fig. 6($c$) is $X' = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,1)$. Suppose that city 3 in X is randomly selected as the center of a circle and r_1 as the radius, the cities in



Note: The number in the circle represents the city number, the number on the line connecting the two circles represents the distance between the two cities, and the yellow dotted line represents the path after reconnection.

FIGURE 6. **The neighborhood 2-opt exchange.**

the neighborhood are 2 and 3. Thus, $aa_1 = 2$, $aa_2 = 3$, $bb_1 = 10$, and $bb_2 = 11$. The new route of X after Step 4 is shown in Fig. 6(*b*), where $d(2, 10) + d(3, 11) < d(2, 3) + d(10, 11)$, it can be noticed that the path length after the neighborhood 2-opt exchange is better than route X. City 6 in X' is assumed as the center of a circle, and the cities contained in the neighborhood with r_1 as the radius are 5, 6, 7, and 8. Suppose cities 6 and 7 are randomly selected from the neighborhood. Thus, $aa_1 = 6$, $aa_2 = 7$, $bb_1 = 5$, and $bb_2 = 8$. The new route after Step 4 is shown in Fig. 6(*d*), where $d(5, 7) + d(6, 8) < d(5, 6) + d(7, 8)$. The new path after the neighborhood 2-opt exchange is better than the initial route X.

It can be observed from the comparison of Fig. 5 and Fig. 6 that the two methods are likely to produce excellent individuals. However, the neighborhood 2-opt exchange improves the probability for it can reduce the probability of the long distance between the randomly selected cities.

2) DOUBLE-BRIDGE EXCHANGE

The idea of the double-bridge exchange [47] is to delete the four edges that are not adjacent in the current loop and then reconnect the edges. The double-bridge exchange can change the shape of the loop, which reduces the probability of search stagnation. The individual $X = (x_1, x_2, \ldots, x_m)$ is taken as an example to illustrate the operator (m is the city size), the main steps are as follows:

Step 1: A uniformly distributed random integer a_1 is generated in the range $[2, m-6]$;

Step 2: A uniformly distributed random integer a_2 is generated in the range $[2 + a_1, m-4]$;

Step 3: A uniformly distributed random integer a_3 is generated in the range $[2 + a_2, m-2]$;

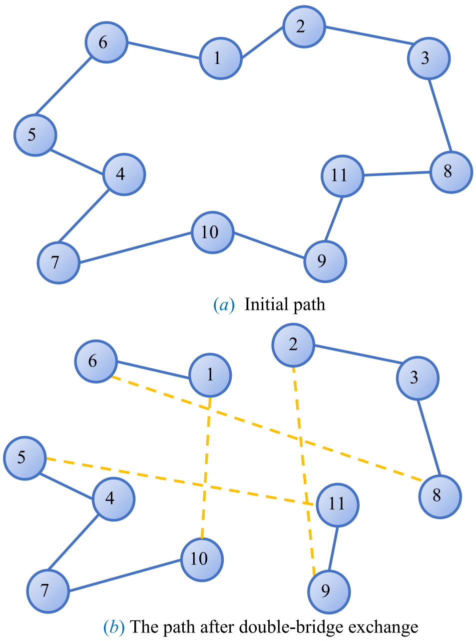
Step 4: A uniformly distributed random integer a_4 is generated in the range $[2 + a_3, m]$;

Step 5: Let $b_1 = a_1 - 1$, $b_2 = a_2 - 1$, $b_3 = a_3 - 1$, $b_4 = a_4 - 1$;

Step 6: $a_1, b_1, a_2, b_2, a_3, b_3, a_4$, and b_4 are the index of each component in individual X. The cities $aa_1, bb_1, aa_2, bb_2, aa_3, bb_3, aa_4$, and bb_4 corresponding to index $a_1, b_1, a_2, b_2, a_3, b_3, a_4$, and b_4 are found in X;

Step 7: The edges (aa_1, bb_1), (aa_2, bb_2), (aa_3, bb_3), (aa_4, bb_4) are deleted, and edges (aa_1, bb_3), (aa_2, bb_4), (aa_3, bb_1) and (aa_4, bb_2) are reconnected.

Fig. 7 illustrates the process of the double-bridge exchange, where the initial path in Fig. 7(*a*) is $X = (3, 2, 1, 6, 5, 4, 7, 10, 9, 11, 8, 3)$. Suppose that the four randomly generated integers are $a_1 = 3$, $a_2 = 5$, $a_3 = 9$, and $a_4 = 11$, thus the cities are $aa_1 = 1$, $aa_2 = 5$, $aa_3 = 9$, and $aa_4 = 8$ in X, $bb_1 = 2$, $bb_2 = 6$, $bb_3 = 10$, $bb_4 = 11$ according to the Step 6.



Note: The number in the circle represents the city number, and the yellow dotted line represents the path after reconnection.

FIGURE 7. **The double-bridge exchange.**

The new route after the double-bridge exchange is shown in Fig. 7(*b*).

3) THE COMBINATION STRATEGY OF NEIGHBORHOOD 2-OPT AND DOUBLE-BRIDGE EXCHANGE

The double-bridge exchange [48] affects eight cities on the route, which exhibits strong perturbations than the neighborhood 2-opt exchange. To make a balancing algorithm, the double-bridge exchange at an early stage and neighborhood 2-opt in the later phase should be selected, and an adaptive selection probability P_s is designed, which is calculated as

$$P_s = P_{\max} - (P_{\max} - P_{\min})\left(\frac{t}{t_{\max}}\right) \tag{20}$$

where $t_{\max}$ is the maximum runtime, t is the current runtime, $P_{\min}$ is the minimum selection rate, $P_{\max}$ is the maximum selection rate, and $P_{\min} = 0.25$, $P_{\max} = 0.7$ in this paper.

The implementation method of the two operators is as follows: 1) μ is randomly and uniformly distributed generated in the range [0, 1]; 2) if $\mu < P_s$, the double-bridge exchange is selected; else the neighborhood 2-opt exchange is selected.

F. LOCAL SEARCH ALGORITHM

The 2-Opt algorithm [46] is an effective algorithm for solving TSP. The main idea is that for each route in the population, the two non-adjacent edges of a given route are exchanged in turn, and preserve the path which can improve solution quality. The 2-Opt algorithm can effectively eliminate the crossed edge in the solution, the probability of crossed path exists is high in the early stage and decreases with iteration. Therefore, the algorithm plays a higher role in the early stage than in the later phase, and the time complexity of 2-Opt is $O(n^2)$, thus, an adaptive probability P is designed, which is calculated as

$$P = 0.3 + \frac{0.6}{e^{t/t_{\max}}} \tag{21}$$

where $t_{\max}$ is the maximum runtime and t is the current runtime.

The implementation method of the combination exchange strategy is as follows: 1) ε is randomly and uniformly distributed generated in the range [0, 1]; 2) if $\varepsilon < P$, the 2-Opt algorithm is executed; otherwise, do not execute the algorithm.

As is shown in (21), the P is decreased with the iteration time. Thus, the 2-Opt algorithm is executed in the early stage with a higher probability, which helps to eliminate the crossed path and significantly improve the solution quality, and the algorithm is executed in the late phase with a smaller probability, which helps to reduce the complexity of the algorithm.

The adaptive combination perturbation and local search algorithm are recorded as ACPLS, and the pseudo-code of ACPLS is presented in **Algorithm 2**.

Algorithm 2 ACPLS

Input: nn^*rr best individuals X, the maximum number of neighborhood solutions are 10;
Output: New nn^*rr individuals X;

For $i = 1$: nn^*rr
  If rand $< P^s$
    Execute 2-opt exchange on X_i, record the new individuals as X_i';
  elseif
    Execute double-bridge exchange on X_i, record the new individuals as X_i';
  End
  If rand $< P$
    Execute 2-Opt algorithm on X_i', record the new individuals as XX;
  End
  Calculate the fitness of XX, return the best XX to X_i;
End For

The CPA has strong exploration ability, and ACPLS exhibits strong exploitation ability. A hybrid algorithm integrating the ACPLS strategy into CPA can make a balance, which can promote the convergence speed and enhance the solution quality.

G. THE FRAMEWORK OF CPA-HDM

The flowchart of CPA-HDM is depicted in Fig. 8. It can be seen that CPA-HDM mainly consists of classification grouping, growth phase, reproduction phase, recombination phase, and ACPLS. Firstly, nn individuals are randomly generated with (9); Secondly, the classification and grouping phase is employed in Section II(*B*-1); Thirdly, the improved growth

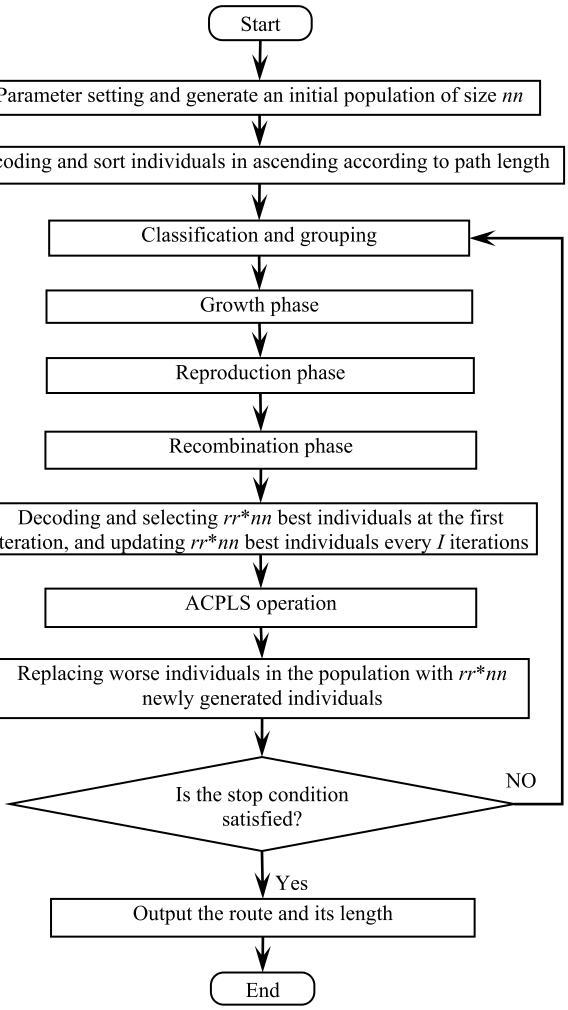


FIGURE 8. **Flowchart of the CPA-HDM.**

and reproduction phase are given in Sections III(C) and III(D); Fourthly, the recombination phase is carried out and its explanation is explained in Section II(B-4); Lastly, the ACPLS is employed and the detail is shown in **Algorithm 2**.

The HDM that both considers the distance between cities and continuous variables is proposed to map continuous variables as legal TSP paths, which helps to enhance the solution quality and population diversification. In the early phase of the iteration, the adaptive attraction probability γ with a small value is adopted, and the IPUMOP is selected with a high probability to reinforce the exploration ability of the algorithm; In the late phase, the IGMOCP is selected with a high probability because γ with a large value is adopted, which works for the exploitation ability; Then, the IRS is proposed in the reproduction phase to reduce the probability of sticking into the local optimum, and CPA integrates the ACPLS, which further amplifies the exploitation ability of the algorithm and reduces the probability of search stagnation. Finally, CPA-HDM evolutionary strategy can retain the best individuals in each iteration. Therefore, through the design of HDM, individuals' update method, and ACPLS operation in the search process of the algorithm, CPA-HDM can both possess exploration and exploitation capabilities.

The pseudo-code of CPA-HDM is shown in **Algorithm 3.**

Algorithm 3 CPA-HDM

Input: population size: nn; the population size of carnivorous plants xp: n, the population size of preys p: n_1, $Maxruntime$; iteration times: t; I; rr

Output: Best solution and the optimal value;

Randomly generate nn initial individuals by (9);
Do HDM in nn individuals to map the continuous variables into discrete ones, the detail is depicted in **Section III(*B*)**;
Calculate the fitness value and sort from small to large based on the fitness value;
While $runtime < Maxruntime$
  $t = t + 1$;
  Set the n best individuals as xp, the remaining n_1 individuals as p, and sort a group as depicted in **Section II(*B*-1)**);
  $Newxp$ and $Newp$ are updated by (13) - (14);
  $Newxp$ is updated by (17) - (18);
  Combined $Newxp$, $Newp$, and xp as a new population, which is recorded as A;
  Calculate the fitness value of A, and select nn best individuals, which is recorded as B;
  If $t = 1$
    Select rr*nn best individuals from B and record as C;
  elseif t mod $I = 0$
    Select rr*nn best individuals from B and record as C;
  End if
  Do **Algorithm 2**on C;
  Combined B and C, and select nn best individuals to continue iteration;
  Record the running time;
End while
Output the shortest route and its length;

IV. EXPERIMENTS AND ANALYSIS

To measure the performance of CPA-HDM and its improvements, three sets of experiments were produced in this study. The first set of experiments verifies the effectiveness of the HDM, ACPLS, AAP, IGMOCP& IPUMOP, and IRS; the second set of experiments applies to determine the optimal parameters combination of n, n_1, rr, and I; the third set of experiments discusses the superiority of CPA-HDM.

A. TERMINATION CONDITION OF THE EXPERIMENTS

The simulations of the involved algorithms were carried out with Matlab R2019b on a desktop with a 3.4 GHz CPU, and 31.9 GB RAM. The experiments on benchmark instances are taken from the TSPLIB. The termination conditions in

TABLE 3. The maximum runtime of different city size.

City number	Run time (s)
$m < 50$	30
$50 \leq m < 100$	50
$100 \leq m < 200$	100
$200 \leq m < 300$	200
$300 \leq m < 500$	400
$500 \leq m < 600$	500
$600 \leq m < 1000$	600
$m \geq 1000$	1500

TABLE 4. Experimental groups and details of algorithms.

Experiment	Algorithm	HD M	ACP LS	AAP	IGMOCP & IPUMOP	IRS	Other decoding methods
Experiment 1	Variant 1	N	N	N	N	N	OBA
	Variant 2	N	N	N	N	N	rounding
	ICPA	Y	N	N	N	N	N
Experiment 2	ICPA	Y	N	N	N	N	N
	ICPA-1	Y	Y	N	N	N	N
	ICPA-2	Y	Y	Y	Y	N	N
	ICPA-3	Y	Y	Y	Y	N	N
	CPA-HDM	Y	Y	Y	Y	Y	N

CPA-HDM are as follows: 1) the maximum running time, which is fairer than the maximum iterations number and the maximum number of fitness evaluations [49]; 2) the optimal value obtained is less than or equal to the theoretical optimal value before the maximum running time reaches. If one of the two conditions is met, the iteration can be stopped. Each experiment was performed with 20 independent runs, recording the best solution for each run. The maximum running time is shown in Table 3.

B. ALGORITHM PERFORMANCE EVALUATION METRICS AND METHOD

The minimum, maximum, average, and standard deviation values of the 20 shortest route lengths are recorded as Best, Worst, Mean, and SD, respectively. The deviation percentage of the Mean is recorded as $\mathrm{PD_{avg}}$, and the average time of 20 independent runs is recorded as t_{av}. The evaluation metrics mentioned above are adopted to measure the performance of the algorithm. $\mathrm{PD_{avg}}$ is calculated as follows:

$$PD_{avg} = \frac{Mean - BKS}{BKS} \times 100\% \tag{22}$$

where BKS is the theoretical optimal solution of the instance.

Friedman test [50], [51] is adopted in this paper to evaluate whether significant differences exist among participating algorithms. The results are computed using the following process.

Step 1: For each instance j ($j = 1, 2, \ldots, n$), rank the Mean of l participating algorithms from 1 (the smallest) to l (the largest). Marked these ranks as $rj\ i$ ($1 \leq i \leq l$, $1 \leq j \leq n$).

Step 2: For the i^{th} algorithm, the average rank of all instance R_i is calculated as:

$$R_i = \frac{1}{n}\sum_{j=1}^{n} r_i^j, \quad i = 1, 2, \cdots, l \tag{23}$$

Step 3: The R_i of l algorithms are sorted from small to large, and the final rank of l algorithms from 1 to l is obtained.

Step 4: Under the null hypothesis, the l participating algorithms perform similarly, The Friedman statistic χ^2 is computed as:

$$\chi^2 = \frac{12\sum_{i=1}^{l}(\sum_{j=1}^{n} r_i^j)^2}{nl\,(l+1)} - 3n(l+1) \tag{24}$$

Step 5: The $\chi^2_{\alpha(l-1)}$ is checked from the chi-square distribution table with the significance level α and $k-1$ degrees of freedom. If $\chi^2 > \chi^2_{\alpha(l-1)}$, the null hypothesis ($H_1$) is accepted, and the l participating algorithms are significantly different; otherwise, hypothesis (H_0) is accepted, and the l participating algorithms are similar.

To better verify the difference between the involved algorithm, Iman and Davenport [51] presented a better statistic F_{ID}, which is calculated as:

$$F_{ID} = \frac{(n-1)\,\chi^2}{n\,(l-1) - \chi^2} \tag{25}$$

where n represents the number of benchmark instances, and l represents the number of participating algorithms.

The $F_{[(l-1),(l-1)(n-1)]}$ is checked from the F distribution table with $l-1$ and $(l-1)(n-1)$ degrees of freedom. If $F_{ID} > F_{[(l-1),(l-1)(n-1)]}$, H_0 is rejected, and the l participating algorithms are significantly different; otherwise, H_1 is rejected, and the l participating algorithms are similar.

The Friedman tests only can detect significant differences over the whole multiple comparisons, being unable to find the concrete pairwise comparisons which produce significant differences. Thus, if the Friedman test shows that significant differences exist in l algorithms, the post hoc test needs to be employed to find out the concrete pairwise comparisons which produce significant differences. Holm's procedure is adopted in this paper and it can be divided into multiple comparisons with a control algorithm and multiple comparisons among all algorithms [52].

1) MULTIPLE COMPARISONS WITH A CONTROL ALGORITHM

The significant difference of the control algorithm will be contrasted against the rest of the $l-1$ participating algorithms in this situation. Suppose the control algorithm is the algorithm1 (Al_1), the adjusted p-value between Alg_1 and v^{th} algorithm (Al_v) is recorded as the APV_v (v is the rank value corresponding to the p-values sorted from small to large, $1 \leq v \leq l-1$, the p-value of each hypothesis obtained through the conversion of the results by the Friedman rank test by adopting a normal approximation [53]), Holm's procedure determines whether the two algorithms are significant by comparing the APV_v and the significance level α. If $APV_v < \alpha$, the Al1 and Al_v are significantly different. The APV_v is

TABLE 5. Experiment results of ICPA, Variant 1, and Variant 2.

Instance Name	BKS	Evaluation indicators	ICPA	Algorithms Variant 1	Variant 2
bayg29	9073	Best	**9073**	9199	10250
		Mean	**9089.5**	10368.45	11142.35
		SD	**28.46**	1033.42	423.79
		PD_{avg} (%)	**0.18**	14.28	22.81
att48	33522	Best	**33522**	38990	48211
		Mean	**33648**	50029	57773.6
		SD	**201.27**	6647.55	8242.65
		PD_{avg} (%)	**0.37**	49.24	72.35
eil51	426	Best	**427**	518	578
		Mean	**431.6**	681.75	755.5
		SD	**2.69**	120.98	116.82
		PD_{avg} (%)	**1.31**	60.04	77.35
berlin52	7542	Best	**7542**	10505	10116
		Mean	**7671.4**	11908.8	11257.75
		SD	**98.76**	992.96	746.8
		PD_{avg} (%)	**1.72**	57.90	49.27
st70	675	Best	**686**	1134	1265
		Mean	**697.45**	1539.65	1776.9
		SD	**7.77**	192.12	236.42
		PD_{avg} (%)	**3.33**	128.10	163.24
eil76	538	Best	**548**	1049	945
		Mean	**556.7**	1252.5	1226.9
		SD	**4.52**	134.1	189.12
		PD_{avg} (%)	**3.48**	132.81	128.05
pr76	108159	Best	**110078**	179428	178266
		Mean	**112336**	238243.8	204127.2
		SD	**1310**	38484.23	16251.94
		PD_{avg} (%)	**3.86**	120.27	88.73
rat99	1211	Best	**1225**	2433	2125
		Mean	**1283.35**	3433.35	2581
		SD	**34.34**	554.16	214.63
		PD_{avg} (%)	**5.97**	183.51	113.13
kroA100	21282	Best	**21627**	57609	56799
		Mean	**22177.1**	73763.45	79859.3
		SD	**335.44**	11977.02	11529.38
		PD_{avg} (%)	**4.21**	246.6	275.24
kroB100	22141	Best	**22457**	51218	56270
		Mean	**22740.85**	62285.95	74074.25
		SD	**213.24**	8727.11	12607.12
		PD_{avg} (%)	**2.71**	181.31	234.56
eil101	629	Best	**654**	1377	1353
		Mean	**664.9**	1794.95	1665.45
		SD	**8.48**	232.76	246.77
		PD_{avg} (%)	**5.71**	185.37	164.78
lin105	14379	Best	**14562**	35556	30182
		Mean	**14818.7**	51275.35	35700.05
		SD	**215.43**	51275.35	4305.08
		PD_{avg} (%)	**3.06**	256.60	148.28
pr107	44303	Best	**44794**	114196	94575
		Mean	**45378.15**	207310.8	128810.3
		SD	**415.71**	57086.88	16245.4
		PD_{avg} (%)	**2.43**	367.94	190.75
pr124	59030	Best	**59666**	279317	171069
		Mean	**60409.55**	337468.7	204036.1
		SD	**788.09**	41563.67	31704.77
		PD_{avg} (%)	**2.34**	471.69	245.65
bier127	118282	Best	**121702**	313592	242342
		Mean	**124119.9**	356746.3	296575.3
		SD	**1479.41**	25087.41	38643.79
		PD_{avg} (%)	**4.94**	201.61	150.74
ch130	6110	Best	**6344**	19469	17532
		Mean	**6477.95**	24700.8	25003.8
		SD	**87.02**	2068.23	2771.41
		PD_{avg} (%)	**6.02**	304.27	309.23
ch150	6528	Best	**6717**	25427	21171
		Mean	**6865.75**	30369.75	27319.3
		SD	**86.42**	2916.76	3771
		PD_{avg} (%)	**5.17**	365.22	318.49
d198	15780	Best	**16530**	64394	51531
		Mean	**17218.7**	89574.75	65028.75
		SD	**367.37**	15130.96	7124.80
		PD_{avg} (%)	**9.12**	467.65	312.10
kroA200	29368	Best	**30860**	135394	146620
		Mean	**32041.5**	186181.8	186864.9
		SD	**523.65**	21324.84	10226.26
		PD_{avg} (%)	**9.10**	533.96	536.29
pr226	80369	Best	**84740**	868730	566777
		Mean	**86975.6**	961475	737094.9
		SD	**1897.79**	59302.8	86801.77
		PD_{avg} (%)	**8.22**	1096.33	817.14

TABLE 5. *(Continued.)* Experiment results of ICPA, Variant 1, and Variant 2.

Note: the best is set in bold.

calculated as follows:

$$APV_v = \min\{R, 1\} \quad v = 1, 2, \cdots, l-1 \tag{26}$$

$$R = \max\{(l-u) * p_u\}\ 1 \leq u \leq v, \quad p_1 \leq p_2 \leq \cdots \leq p_v \leq \cdots \leq p_{l-1} \tag{27}$$

2) MULTIPLE COMPARISONS AMONG ALL ALGORITHMS

In this situation, the significant difference of each algorithm will be contrasted against the rest of the $l-1$ algorithms participating in the comparison, the possible pairwise comparison between algorithms is M, and $M = l*(l-1)/2$. The algorithm x is recorded as Al_x, the algorithm y is recorded as Al_y ($1 \leq x \leq l, 1 \leq y \leq l$). Suppose the rank of the p-value sorted from small to large between Al_x and Al_y among all pairwise comparisons is v ($1 \leq v \leq M$), and the adjusted p-value between Al_x and Al_y is recorded as APV_v. If $APV_v < \alpha$, the Al_x and Al_y are significantly different. The APV_v is calculated as follows:

$$APV_v = \min\{R, 1\} \quad v = 1, 2, \cdots, M \tag{28}$$

$$R = \max\{(M-u+1) * p_u\}\ 1 \leq u \leq v, \quad p_1 \leq p_2 \leq \cdots \leq p_v \leq \cdots \leq p_M \tag{29}$$

C. COMPARISONS AND ANALYSIS

1) VALIDATION OF THE IMPROVEMENT

For a better description, the CPA with order-based arrangement (OBA) decoding method [30] is recorded as Variant 1; the CPA with rounding method [33] is recorded as Variant 2; the CPA with HDM is recorded as ICPA; the ICPA with ACPLS is recorded as ICPA-1; the ICPA-1 with AAP is recorded as ICPA-2; the ICPA-2 with IGMOCP & IPUMOP is recorded as ICPA-3; the ICPA-3 with IRS is recorded as CPA-HDM.

Variant 1, Variant 2, and ICPA are adopted to verify the validation of HDM, which is marked as Experiment 1; ICPA, ICPA-1, ICPA-2, ICPA-3, and CPA-HDM are adopted to verify the validation of ACPLS, AAP, IGMOCP & IPUMOP, and IRS, which is marked as Experiment 2. The experimental groups and details are shown in Table 4. Y represents

TABLE 6. Experiment results of ICPA and its different versions.

Instance Name	Instance BKS	Evaluation indicators	Algorithms ICPA	ICPA-1	ICPA-2	ICPA-3	CPA-HDM
eil101	629	Best	654	631	630	630	**629**
		Worst	681	643	643	644	**639**
		Mean	664.9	639.6	638.45	637.3	**631.75**
		SD	8.48	2.62	3.50	3.73	**3.12**
		PD_{avg} (%)	5.71	1.69	1.50	1.32	**0.44**
lin105	14379	Best	14562	**14379**	**14379**	**14379**	**14379**
		Worst	15348	14449	14447	**14434**	14456
		Mean	14818.7	14402.45	14396.65	14394.15	**14385**
		SD	215.43	24.07	20.36	19.44	**17.61**
		PD_{avg} (%)	3.06	0.16	0.12	0.11	**0.04**
pr107	44303	Best	44794	**44303**	**44303**	**44303**	**44303**
		Worst	46493	44665	44610	44596	**44431**
		Mean	45378.15	44496.95	44489.35	44458.55	**44334.7**
		SD	415.71	96.38	105.49	78.16	**40.44**
		PD_{avg} (%)	2.43	0.44	0.42	0.35	**0.07**
pr124	59030	Best	59666	**59030**	**59030**	**59030**	**59030**
		Worst	63335	59159	59092	59159	**59030**
		Mean	60409.55	59061.9	59048.55	59047.95	**59030**
		SD	788.09	43.28	27.34	78.16	**0.00**
		PD_{avg} (%)	2.34	0.05	0.03	0.03	**0.00**
bier127	118282	Best	121702	118536	118459	**118282**	**118282**
		Worst	125847	119638	119578	119293	**118978**
		Mean	124119.9	119111.8	119083.7	118949.6	**118414**
		SD	1479.41	280.3	291.7	218.76	**181.65**
		PD_{avg} (%)	4.94	0.70	0.68	0.56	**0.11**
ch130	6110	Best	6344	6138	6148	6120	**6110**
		Worst	6702	6214	6202	6204	**6173**
		Mean	6477.95	6172.10	6171.40	6168.95	**6136.45**
		SD	87.02	19.63	16.06	22.08	**18.71**
		PD_{avg} (%)	6.02	1.02	1.00	0.96	**0.43**
ch150	6528	Best	6717	6554	6544	6528	**6528**
		Worst	7050	6616	6629	6620	**6572**
		Mean	6865.75	6589.90	6597.85	6587.85	**6543.05**
		SD	86.42	18.89	24.7	28.31	**15.05**
		PD_{avg} (%)	5.17	0.95	1.07	0.92	**0.23**
d198	15780	Best	16530	15954	15919	15895	**15800**
		Worst	18063	16082	16075	16050	**15921**
		Mean	17218.7	16009.8	15996.7	15988.9	**15854.6**
		SD	367.37	40.99	48.99	44.44	**35.07**
		PD_{avg} (%)	9.12	1.46	1.37	1.32	**0.47**
kroA200	29368	Best	30860	29762	29670	29687	**29394**
		Worst	32962	30040	30072	30056	**29553**
		Mean	32041.50	29897.55	29887.15	29865.95	**29456.6**
		SD	523.65	77.06	99.88	113.25	**46.42**
		PD_{avg} (%)	9.10	1.80	1.77	1.70	**0.30**
pr226	80369	Best	84740	80421	80418	80421	**80369**
		Worst	91117	81023	80843	80731	**80655**
		Mean	86975.6	80694.65	80620.9	80616.3	**80422.9**
		SD	1897.79	160.8	129.38	116.98	**82.76**
		PD_{avg} (%)	8.22	0.41	0.31	0.31	**0.07**
MSD / MPD_{avg} (%)			586.9/5.6	76.4/0.87	76.7/0.83	72.3/0.76	**44/0.22**

Note: the best is set in bold.

that method is contained in the algorithm, and N represents that method is not contained in the algorithm. The maximum runtime and the details of the participating algorithms are shown in Table 3, the Best, Worst, Mean, SD, and PD_{avg} (%) are adopted as measurements in the following comparison.

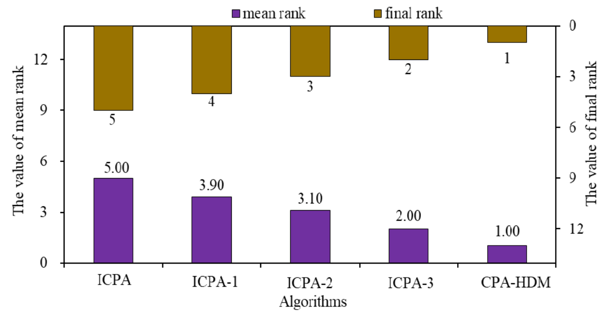


FIGURE 9. The rank of the five participating algorithms.

TABLE 7. The results of Friedman statistic when the algorithms number $l = 5$ and standard instances number $n = 10$.

α	χ^2	$\chi^2_{\alpha(l-1)}$	F_{ID}	$F_{[(l-1),(l-1)(n-1)]}$	H_0	H_1
0.05	39.28	9.49	491	2.63	Reject	Accept

TABLE 8. Four-factor four-level orthogonal experiment.

Experiment number	n	n_1	rr	I
1	5	120	0.3	5
2	10	80	0.4	10
3	20	80	0.2	3
4	10	100	0.2	8
5	10	120	0.3	5
6	5	80	0.4	8
7	20	160	0.4	10
8	5	80	0.1	3
9	5	160	0.2	10
10	20	120	0.1	5
11	5	160	0.1	3
12	10	120	0.2	10
13	20	100	0.3	8
14	10	100	0.3	8
15	5	100	0.4	5
16	10	160	0.1	3

The ICPA, Variant 1, and Variant 2 are adopted with 20 instances from TSPLIB to compare the performance of HDM. To achieve a fair comparison, the same parameters are set in the participating algorithms. The population size $nn = 120$, the number of carnivorous plants $n = 20$, and the number of carnivorous prey $n_1 = 100$. The performance of HDM and the other two decoding methods considered for comparison are displayed in Table 5.

It can be observed from Table 5 that the Best, Mean, SD, and PD_{avg} (%) values obtained by ICAP are all better than Variant 1 and Variant 2 in 20 instances, which verified that the HDM can direct the produced discrete solutions towards optimality compared with the other two decoding methods.

To show the effect of the ACPLS, AAP, IGMOCP & IPUMOP, and IRS on the overall performance of the proposed algorithm, several self-comparisons between different versions of ICPA are conducted with 20 instances. The parameters $nn = 120$, $n = 20$, $n_1 = 100$, $rr = 0.2$, $I = 5$, the MSD

TABLE 9. Results of orthogonal experimental design.

Experiment	pr107	pr144	kroB150	d198	kroA200	kroB200	tsp225	a280	pr299	lin318	pr442
1	44341.4	58649.4	26164.87	15840.33	29473.07	29615.40	3942.93	2593.33	48550.93	42478	51546.27
2	44326.8	58638.93	26198.07	15809.67	29476.2	29547.73	3946.47	2591.27	48518.53	42376.60	51572.6
3	44380.87	58736.4	26275.87	15892	29536.33	29701.47	3940.67	2594.47	48586.73	42514.80	51808.27
4	44344.67	58651	26197.27	15874.07	29502.4	29617.27	3947.93	2599.20	48619.6	42530	51568.27
5	44323.27	**58537**	26188.9	15822.4	29436.8	29503.80	3955.7	2590.87	48493.05	**42242.73**	51377.27
6	44330	58575.53	26201.2	15835.93	29467.33	29593.07	3939.53	2596.47	48586.13	42292.80	**51317.13**
7	**44310.93**	58582.4	26193.33	**15808.13**	**29406.73**	29469.07	**3934.53**	2589.8	48634.4	42283.07	51416.6
8	44580.93	58847.73	26393.67	15947.47	29612.80	29800.07	3973.07	2619.13	49267.93	42718	51918.6
9	44340.67	58577.47	26156.20	15851.93	29498.00	29520.87	3936.93	2601	48538.87	42334.67	51385.27
10	44383.87	58705.47	26283.87	15904.27	29654.53	29686.07	3965.93	2604.93	48584.4	42613.53	51643.07
11	44483.67	58674.07	26271.2	15889.60	29572.47	29679.27	3965.87	2615.13	48842.8	42529.87	51774.4
12	44431	58759.47	26306.4	15855.67	29577.13	29652.33	3960.80	2620.27	48645.4	42435.53	51738.47
13	44341	58622.13	26178.73	15818.20	29518.07	29563.87	3940.73	2593.33	48532.33	42272.07	51277.8
14	44330.73	58649.73	26176.67	15824.07	29469.47	29597.33	3944.20	2592.4	**48416.8**	42368.87	51397.73
15	44331.47	58639.93	**26146.87**	15818.47	29435.33	29504.47	3946.47	**2585.4**	48421.87	42316.87	51378.33
16	44421.40	58709.73	26265.8	15876.80	29554.2	29794.07	3971.53	2616.67	48601.27	42475.53	51554.4

Note: the best is set in bold

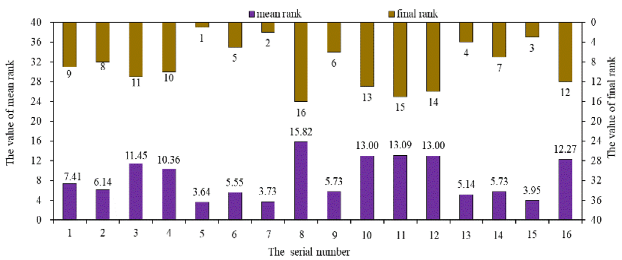


FIGURE 10. The rank of each experiment scheme.

is the average of SD, and the MPD_{avg} (%) is the average of PD_{avg} (%). The results are displayed in Table 6.

The results from Table 6 can be observed that the ICPA performs poorly with the largest MSD and MPD_{avg} (%) values, and each added component gets lower MPD_{avg} (%) values in comparison with its previous version, which demonstrates the ACPL, AAP, IGMOCP & IPUMOP, and IRS have a positive effect on the performance of the ICPA.

As is depicted in Fig. 9, the mean rank of the Friedman rank test is ICPA > ICPA-1 > ICPA-2 > ICPA-3 > CPA-HDM, and the final rank is CPA-HDM < ICPA-3 < ICPA-2 < ICPA-1 < ICPA, the lower the final rank, the superior the performance of the algorithm, which verifies that each added component can enhance the algorithms' performance.

At the significant level of 0.05, the results of Friedman statistic χ^2 and F_{ID} are summarized in Table 7, where χ^2 is 39.28 and F_{ID} is 491. The critical value of $\chi 2$ 0.05[4] is 9.49 with degrees of freedom $l-1=4$, $F_{(4,36)}$ is 2.63 with $l-1=4$ and $(l-1)(n-1)=36$ degrees of freedom.

The results in Table 7 show that $\chi^2 > \chi^2_{0.05[4]}$, $F_{ID} > F_{(4,36)}$ at $\alpha = 0.05$. Thus, the differences among the five comparison algorithms are significant. The analyses above confirm that the ACPLS, AAP, IGMOCP & IPUMOP, and IRS can make a positive impact on the proposed algorithm.

TABLE 10. The results of Friedman statistic when the algorithms number $l = 16$ and standard instances number $n = 11$.

α	χ^2	$\chi^2_{\alpha(l-1)}$	F_{ID}	$F_{[(l-1),(l-1)(n-1)]}$	H_0	H_1
0.05	121.77	25	28.17	1.73	Reject	Accept

TABLE 11. Parameter settings of participating algorithms.

Algorithm	Reference	Parameters
CPA-HDM	-	nn=130, n=10, n_1=120, I=5, rr=0.3
D-GWO	[23]	population size: 50
DSFLA	[40]	α= [1,5], β= [1,5], Q=1000, ρ=0.1, N=20, m=10
DBAL	[38]	population size: 50, r_i: 0.5, A_i:0.5
AGBSO3	[54]	*cluster_num*: 5, *p_replace*: 0.3, *p_one*: 0.6, *p_one_center*: 0.45, *p_two_center*: 0.5, *MM*: 100
ABC	[28]	E_b=20, O_b=0.5E_b+1, $Mait3$=10, lim=5
PACO-3Opt	[55]	Q=30, ρ=0.1, I=5, α=3, β=2, m=10

2) THE OPTIMAL PARAMETERS COMBINATION OF CPA-HDM THROUGH ORTHOGONAL EXPERIMENTS

The performance of CPA-HDM is related to the number of carnivorous plants n, the number of prey n_1, the proportion rr of the population to execute ACPLS, and the individuals performing the ACPLS are re-selected from the population every I iterations. Therefore, to determine the optimal parameters combination of n, n_1, rr, and I, the orthogonal experiment in Table 8 is designed.

The maximum runtime is given in Table 3, Table 9 summarizes the results of the mean best value by solving 11 instances in 16 groups.

TABLE 12. Experiment results of CPA-HDM and the other six participating algorithms.

Name	BKS	Evaluation indicators	CPA-HDM	D-GWO	DSFLA	DBAL	AGBSO3	ABC	PACO-3Opt
bayg29	9073	Best	**9073**	**9073**	**9073**	**9073**	**9073**	**9073**	**9073**
		Worst	**9073**	**9073**	**9073**	**9073**	9120	**9073**	**9073**
		Mean	**9073**	**9073**	**9073**	**9073**	9078.25	**9073**	**9073**
		SD	**0.00**	**0.00**	**0.00**	**0.00**	11.47	**0.00**	**0.00**
		PD_{avg} (%)	**0.00**	**0.00**	**0.00**	**0.00**	0.06	**0.00**	**0.00**
att48	33522	Best	**33522**	**33522**	**33522**	**33522**	33522	**33522**	**33522**
		Worst	**33522**	33599	**33522**	**33522**	34228	33599	**33522**
		Mean	**33522**	33539.4	**33522**	**33522**	33743.7	33525.85	**33522**
		SD	**0.00**	28.81	**0.00**	**0.00**	190.51	16.80	**0.00**
		PD_{avg} (%)	**0.00**	0.05	**0.00**	**0.00**	0.66	0.01	**0.00**
eil51	426	Best	**426**	**426**	**426**	**426**	**426**	**426**	**426**
		Worst	**427**	428	428	**427**	439	**427**	428
		Mean	426.1	426.8	426.35	**426.05**	429.9	426.35	426.75
		SD	0.30	0.68	0.75	**0.22**	3.69	0.48	0.78
		PD_{avg} (%)	0.02	0.19	0.08	**0.01**	0.92	0.08	0.18
pr76	108159	Best	**108159**	**108159**	**108159**	**108159**	**108159**	**108159**	**108159**
		Worst	108202	108936	108866	**108159**	112261	109295	109043
		Mean	108161.15	108361.8	108453	**108159**	110506.2	108297	108465.5
		SD	13.09	298.21	195.68	**0.00**	1165.61	329.70	332.82
		PD_{avg} (%)	0.00	0.19	0.27	**0.00**	2.17	0.13	0.28
rat99	1211	Best	**1211**	1217	1212	**1211**	**1211**	**1211**	**1211**
		Worst	1222	1236	**1213**	1217	1279	1255	1224
		Mean	**1212.2**	1224.7	1212.35	1213.25	1234	1223.6	1215.15
		SD	2.44	6.48	**0.48**	1.85	19.92	11.92	4.14
		PD_{avg} (%)	**0.10**	1.13	0.11	0.19	1.90	1.04	0.34
kroA100	21282	Best	**21282**	**21282**	**21282**	**21282**	**21282**	**21282**	**21282**
		Worst	21294	21415	**21282**	21292	21821	21571	**21282**
		Mean	21282.6	21319.3	**21282**	21282.5	21415.4	21382.2	**21282**
		SD	2.62	36.98	**0.00**	2.18	127.15	78.28	**0.00**
		PD_{avg} (%)	**0.00**	0.18	**0.00**	**0.00**	0.63	0.47	**0.00**
kroB100	22141	Best	**22141**	**22141**	22193	**22141**	22199	**22141**	**22141**
		Worst	**22199**	22336	22284	22232	22737	22364	22246
		Mean	**22146.6**	22222.85	22243.8	22165.15	22376.3	22192.65	22204.4
		SD	**15.68**	45.55	20.01	24.34	136.09	65.96	31.74
		PD_{avg} (%)	**0.03**	0.37	0.46	0.11	1.06	0.23	0.29
lin105	14379	Best	**14379**	**14379**	**14379**	**14379**	**14379**	**14379**	**14379**
		Worst	14449	14434	14426	**14379**	14659	14486	14448
		Mean	14382.5	14396.75	14391.75	**14379**	14470.2	14385.05	14396
		SD	15.28	19.13	16.90	**0.00**	91.30	23.40	27.55
		PD_{avg} (%)	0.02	0.12	0.09	**0.00**	0.63	0.04	0.12
pr107	44303	Best	**44303**	**44303**	**44303**	**44303**	**44303**	**44303**	44347
		Worst	44387	44573	**44339**	44486	44998	44541	44577
		Mean	44323.27	44422	**44306.6**	44351.95	44652.85	44372.05	44476.05
		SD	42.55	81.41	**10.83**	48.19	162.42	84.40	73.17
		PD_{avg} (%)	0.05	0.27	**0.01**	0.11	0.79	0.16	0.39
pr124	59030	Best	**59030**	**59030**	**59030**	**59030**	**59030**	**59030**	**59030**
		Worst	59076	59397	**59030**	**59030**	60516	59779	59385
		Mean	59032.3	59171.15	**59030**	**59030**	59498.75	59198	59081.6
		SD	10.04	117.05	**0.00**	**0.00**	435.20	218.21	72.74
		PD_{avg} (%)	**0.00**	0.24	**0.00**	**0.00**	0.79	0.28	0.09
bier127	118282	Best	**118282**	118454	118423	**118282**	119652	118308	**118282**
		Worst	**118728**	119773	119110	118845	123396	119772	119833
		Mean	**118366.4**	119246.4	118772.4	118465.9	120980.7	118681.1	118938.6
		SD	**156.70**	338.29	194.67	191.30	881.53	327.86	456.58
		PD_{avg} (%)	**0.07**	0.82	0.41	0.16	2.28	0.34	0.56
ch130	6110	Best	**6110**	6143	**6110**	**6110**	6140	**6110**	6134
		Worst	6170	6234	6161	6173	6284	**6151**	6177
		Mean	**6129.4**	6184.15	6137.55	6134.2	6236.2	6132.2	6152.8
		SD	16.82	25.56	**14.63**	15.19	37.23	24.56	13.69
		PD_{avg} (%)	**0.32**	1.21	0.45	0.40	2.07	0.36	0.70
pr144	58537	Best	**58537**	**58537**	**58537**	**58537**	**58537**	**58537**	**58537**
		Worst	**58537**	58736	58603	**58537**	59364	58848	**58537**
		Mean	**58537**	58571.9	58553.55	**58537**	58700.8	58603	**58537**
		SD	**0.00**	55.87	20.58	**0.00**	211.59	118.72	**0.00**
		PD_{avg} (%)	**0.00**	0.06	0.03	**0.00**	0.28	0.11	**0.00**
		Best	**26524**	26680	26703	**26524**	26730	**26524**	26577
		Worst	26807	27103	27014	**26716**	27543	27091	26901

TABLE 12. ***(Continued.)*** **Experiment results of CPA-HDM and the other six participating algorithms.**

kroA150	26524	Mean	**26601.75**	26869.65	26845.25	26637.8	27094.95	26665.8	26774.85
		SD	69.46	127.41	71.80	**57.54**	259.08	144.46	79.24
		PD$_{avg}$ (%)	**0.29**	1.30	1.21	0.43	2.15	0.53	0.95
		Best	**26130**	26233	26324	**26130**	26311	26135	**26130**
		Worst	26316	26610	26478	**26283**	26871	26449	26382
kroB150	26130	Mean	**26188.9**	26382.45	26381.35	26216.9	26462.65	26258.4	26238.6
		SD	70.62	131.33	**41.52**	41.96	133.78	98.21	81.23
		PD$_{avg}$ (%)	**0.23**	0.97	0.96	0.33	1.27	0.49	0.42
		Best	2331	2371	2337	2336	2339	**2326**	2334
		Worst	2358	2416	2361	**2357**	2395	2387	2353
rat195	2323	Mean	2344.1	2395.55	2346.5	2349.7	2366.05	**2343.35**	2345.35
		SD	7.98	12.03	**4.92**	7.63	16.93	12.87	5.75
		PD$_{avg}$ (%)	0.91	3.12	1.01	1.15	1.85	**0.88**	0.96
		Best	**15781**	15804	15929	15804	15987	15798	15832
		Worst	15886	15919	16071	**15872**	16459	16259	15896
d198	15780	Mean	**15822.4**	15879.4	16011.25	15845.45	16250.25	15911.3	15860.6
		SD	30.56	31.97	43.56	**11.15**	139.94	90.85	24.21
		PD$_{avg}$ (%)	**0.27**	0.63	1.47	0.41	2.98	0.83	0.51
		Best	**3920**	3986	3938	3953	3936	3922	3943
		Worst	3995	4048	**3979**	3993	4026	4004	3983
tsp225	3916	Mean	**3955.7**	4023.7	3958	3972.55	3982.45	3957.4	3964.8
		SD	21.87	16.58	**10.78**	11.15	30.32	21.33	12.46
		PD$_{avg}$ (%)	**1.01**	2.75	1.07	1.44	1.70	1.06	1.25
		Best	**49135**	49312	49151	**49135**	**49135**	**49135**	**49135**
		Worst	49431	50276	**49220**	49248	50225	50261	49628
pr264	49135	Mean	49212.8	49767.85	49203.15	**49169.45**	49860.2	49464.05	49271.95
		SD	107.67	209.50	56.03	**34.31**	319.27	257.81	129.73
		PD$_{avg}$ (%)	0.16	1.29	0.14	**0.07**	1.48	0.67	0.28
		Best	**48195**	48540	48551	48315	48710	48337	48580
		Worst	49422	49451	49262	49159	49918	**49116**	49146
pr299	48191	Mean	**48493.05**	49084	48795.2	48861.15	49163.8	48696	48904.4
		SD	258.27	217.92	224.88	180.25	345.95	264.01	**142.54**
		PD$_{avg}$ (%)	**0.63**	1.85	1.25	1.39	2.02	1.05	1.48
		Best	42204	42770	42330	42394	**42197**	42517	42458
		Worst	**42616**	43347	42873	42734	43792	43393	42870
linhp318	41345	Mean	**42417.4**	43067.75	42545.9	42566.55	43204.5	42924.4	42643.4
		SD	151.80	460.07	142.90	**79.83**	487.89	210.88	108.95
		PD$_{avg}$ (%)	**2.59**	4.17	2.90	2.95	4.50	3.82	3.14
		Best	**15423**	15593	15432	15507	15486	15550	15519
		Worst	15666	15874	**15591**	15647	15858	15839	15677
rd400	15281	Mean	15536.75	15748.3	**15497.85**	15583.95	15707.8	15685.6	15605.1
		SD	70.18	74.95	49.26	**43.02**	103.22	79.49	45.38
		PD$_{avg}$ (%)	1.67	3.06	**1.42**	1.98	2.79	2.65	2.12
		Best	**35461**	35840	35621	35604	36162	35639	35559
		Worst	**35885**	36218	36150	35983	37859	36260	35977
d493	35002	Mean	**35669.25**	36066.85	35844	35837.9	36966.95	35853.9	35796.9
		SD	148.93	**92.41**	158.77	94.62	635.57	172.98	112.78
		PD$_{avg}$ (%)	**1.91**	3.04	2.41	2.39	5.61	2.43	2.27
		Best	**87736**	88724	87852	88262	88632	88392	88247
		Worst	89178	90193	**89147**	89099	91704	89411	89176
att532	86729	Mean	**88255.25**	89466.55	88596.7	88760.7	89700.85	88951.95	88873.7
		SD	373.66	351.49	459.70	261.01	663.29	324.61	**224.42**
		PD$_{avg}$ (%)	**1.76**	3.16	2.15	2.34	3.43	2.56	2.47
		Best	**6864**	7046	6867	6955	6948	6936	6900
		Worst	**6946**	7107	7005	7062	7101	7075	7001
rat575	6773	Mean	**6910.35**	7070.45	6914.45	7022.6	6998.35	7014.6	6953.75
		SD	29.77	**16.66**	40.82	24.97	39.93	35.38	23.09
		PD$_{avg}$ (%)	**2.03**	4.39	2.09	3.69	3.33	3.57	2.67
		Best	**42471**	43137	42533	43025	42724	43269	42727
		Worst	**43201**	44027	43330	43682	43593	43875	43232
u724	41910	Mean	**42848.6**	43666.8	42933.95	43440.95	43352.05	43585.55	43035.5
		SD	167.70	181.75	227.89	149.52	249.43	172.12	**124.19**
		PD$_{avg}$ (%)	**2.24**	4.19	2.44	3.65	3.44	4.00	2.69
		Best	9023	9130	**8941**	9173	9095	9061	9038
		Worst	9148	9265	9173	9287	9238	9276	**9140**
rat783	8806	Mean	**9058.7**	9210.05	9063.55	9224.8	9190.45	9184.75	9084.35
		SD	33.19	36.09	61.51	27.63	34.96	53.65	**28.80**
		PD$_{avg}$ (%)	**2.87**	4.59	2.92	4.76	4.37	4.30	3.16
vm1084	239297	Best	**243221**	246094	245596	246638	247574	246238	246960
		Worst	**247618**	251440	250417	249591	253424	250081	250345
		Mean	**245449.1**	249027.1	248359.9	248057.3	250674.3	248415.5	248338.3
		SD	1243.78	1480.46	1232.48	**828.17**	1658.14	1304.95	1059.47
		PD$_{avg}$ (%)	**2.57**	4.07	3.79	3.66	4.75	3.81	3.78
MSD / MPD$_{avg}$ (%)			109.32/**0.78**	160.52/1.69	117.91/1.13	**76.29**/1.04	306.84/2.14	162.28/1.28	114.84/1.11

Note: the best is set in bold

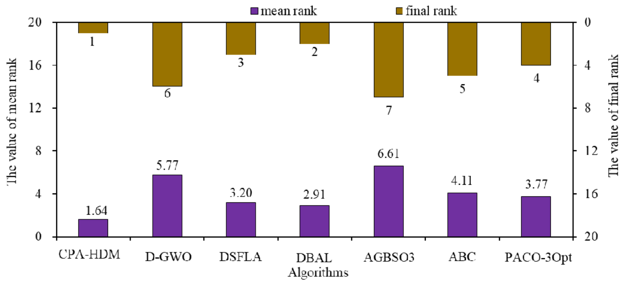


FIGURE 11. The rank of the seven participating algorithms.

Fig. 9 shows the average ranking and the final ranking of the Friedman test, in which can be observed that Experient 5 ranks first among 16 experiments. From the perspective of the Friedman statistic, it can be noticed from Table 10 that χ^2 and F_{ID} are greater than $\chi^2_{\alpha[15]}$ and $F_{(15,150)}$ at the significant level of 0.05, which shows a significant difference between the compared experimental combinations. Based on the findings obtained so far, it can confirm that Experient 5 is superior to the other remaining 15 experiments. Therefore, the optimal parameter combination is $n = 10$, $n_1 = 120$, $rr = 0.3$, and $I = 5$.

3) CPA-HDM COMPARED WITH OTHER PARTICIPATING ALGORITHMS AND ANALYSIS

To demonstrate the performance of CPA-HDM, 28 instances with cities from 29 up to 1084, have been selected and compared with six algorithms in the literature, namely: D-GWO [23], DSFLA [40], DBAL [38], agglomerative greedy brain storm optimization algorithm (AGBSO3) [54], a parallel cooperative hybrid method based on 3-Opt and ant colony (PACO-3Opt) [55], and ABC [28]. Among them, the D-GWO, DSFLA, DBAL, PACO-3Opt, and ABC are memetic algorithms. The parameters of the participating algorithms are shown in Table 11.

The comparison results of these algorithms are presented in Table 12. The maximum runtime is shown in Table 3, the details of the participating algorithms are shown in Table 11, and the Best, Worst, Mean, SD, PD_{avg} (%), MSD, and MPD_{avg} (%) values are set as the measurement of accuracy and stability. The MSD is the average of SD, the MPD_{avg} is the average of PD_{avg} (%).

The results, which are given in Table 12, show the efficiency of CPA-HDM, as it could achieve higher accuracy for most instances and better MPD_{avg} (%) values than the other six algorithms, it only fails to find the lower Best value in rat195 and rat783. The CPA-HDM gets the 16 theoretical optimal solutions on 28 benchmark instances, and the PD_{avg} (%) is no more than 0.91% on 19 instances. The MPD_{avg} (%) is 0.78%, which is 0.91%, 0.35%, 0.26%, 1.36%, 0.5%, and 0.33% superior to the D-GWO, DSFLA, DBAL, AGBSO3, ABC, and PACO-3Opt, respectively.

The Friedman rank test is carried out and the results are depicted in Fig. 11. The mean rank is AGBSO3 > D-GWO > ABC > PACO-3Opt > DSFLA > DBAL > CPA-HDM, and the final rank is AGBSO3 < D-GWO < ABC < PACO-3Opt < DSFLA < DBAL < CPA-HDM, which illustrates that CPA-HDM is superior to the other algorithms.

TABLE 13. The results of Friedman statistic when the algorithms number *l* = 7 and standard instances number *n* = 28.

α	χ^2	$\chi^2_{\alpha(l-1)}$	F_{ID}	$F_{[(l-1),(l-1)(n-1)]}$	H_0	H_1
0.05	107.83	12.59	48.39	2.16	Reject	Accept

TABLE 14. Unadjusted and adjusted *p*-values by Holm's post hoc test (CPA-HDM is the control algorithm).

v	Algorithm	Unadjusted *p*-value	Adjusted *p*-value
1	AGBSO3	0E-0	0E-0
2	D-GWO	9.02E-13	4.51E-12
3	ABC	2.00E-05	8.00E-05
4	PACO-3Opt	2.33E-04	6.99E-04
5	DSFLA	7.13E-03	1.43E-02
6	DBAL	2.81E-02	2.81E-02

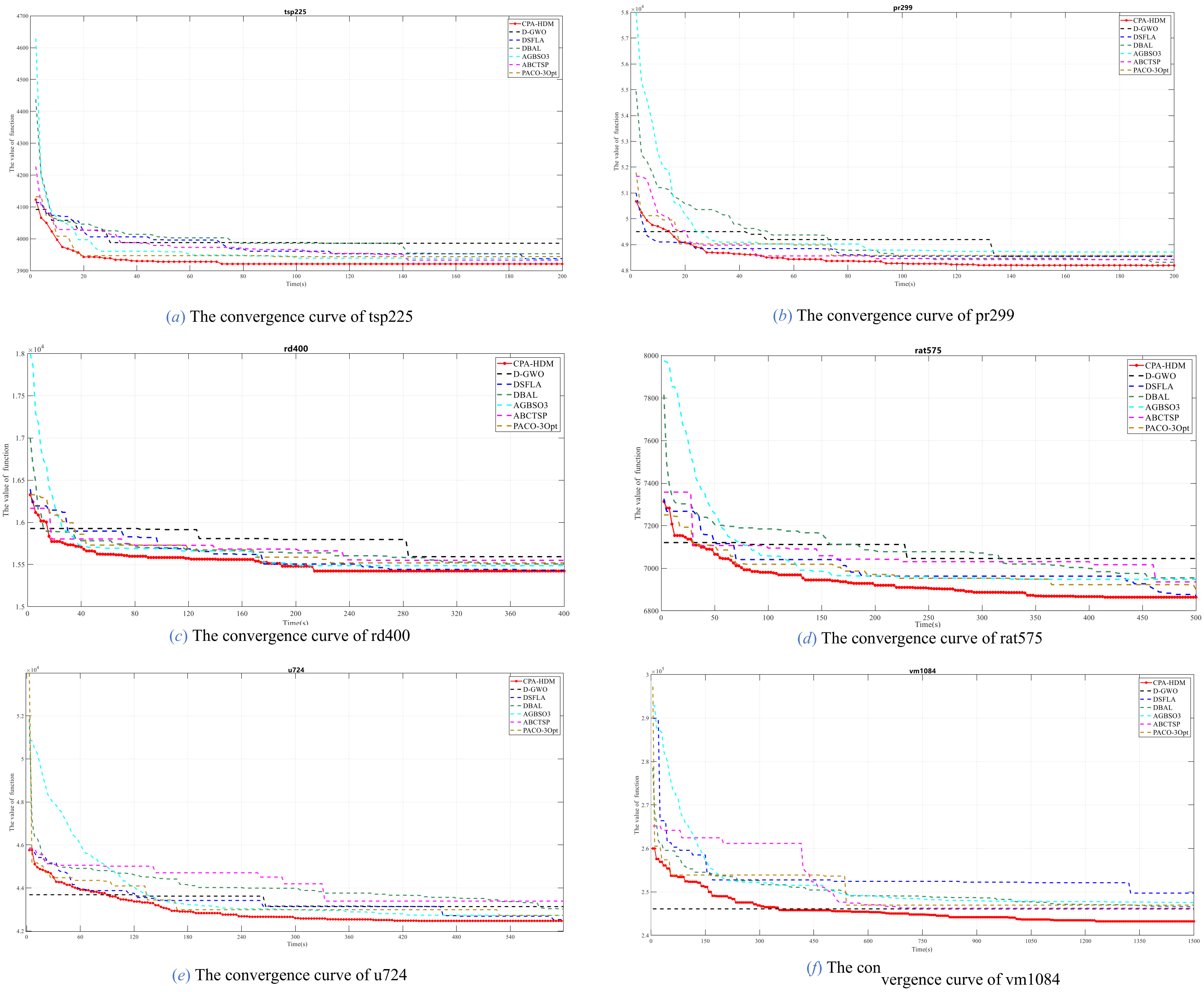
TABLE 15. Computation time (s) of algorithms when city size is less than 300.

Instance	Algorithms						
	CPA-HDM	D-GWO	DSFLA	DBAL	AGBSO3	ABC	PACO-3Opt
bayg29	**0.32**	0.94	1.86	0.68	14.73	0.42	1.55
att48	1.47	19.99	**0.92**	4.11	27.08	4.56	2.63
eil51	**14.95**	40.43	35.22	14.97	46.24	19.52	37.61
pr76	**10.89**	33.02	48.86	17.20	47.38	21.03	38.56
rat99	**31.01**	50.79	50.20	46.88	48.28	48.23	44.93
kroA100	18.64	84.12	20.51	16.82	90.83	34.98	**8.56**
kroB100	**33.94**	99.27	100.21	42.30	100.01	35.10	90.96
lin105	**21.21**	72.09	56.08	23.55	80.08	27.10	40.15
pr107	57.73	94.46	**37.09**	80.79	96.21	71.01	100.02
pr124	10.19	92.97	24.48	**9.31**	86.55	81.89	78.45
bier127	**61.55**	101.69	100.31	97.84	100.01	100.5	96.33
ch130	87.85	102.72	96.94	95.63	100.01	**86.62**	100.04
pr144	**6.52**	72.80	80.24	32.88	79.35	74.28	12.38
kroA150	91.24	102.74	100.38	98.36	100.02	**89.09**	100.08
kroB150	95.94	103.15	100.31	97.31	100.02	92.62	**90.38**
pr264	189.59	202.29	200.24	**156.5**	189.95	188.6	193.28
T_{avg}	**45.82**	79.59	65.87	52.20	81.67	60.97	64.74

Note: set the best in bold

The results of Friedman statistic χ^2 and F_{ID} are shown in Table 13. The value of χ^2 is 107.83, and F_{ID} is 48.39. When the degrees of freedom $l - 1 = 6$ and the significant level $\alpha = 0.05$, the critical value of $\chi^2_{\alpha[6]}$ is 12.59, and the critical value of $F_{(6,162)}$ at $(l-1)(n-1) = 162$ is 2.16. It can be observed that $\chi^2 > \chi 2\alpha$ [6], $F_{ID} > F_{(6,162)}$ at $\alpha = 0.05$, which verifies that the differences among the algorithms are significant, and the outperformance of CPA-HDM has been confirmed.

For further statistical analysis, Holm's procedure is employed to evaluate the practical difference between CPA-HDM and the other six algorithms at a 95% confidence level. The unadjusted and adjusted *p*-values returned



(*a*) The convergence curve of tsp225

(*b*) The convergence curve of pr299

(*c*) The convergence curve of rd400

(*d*) The convergence curve of rat575

(*e*) The convergence curve of u724

(*f*) The convergence curve of vm1084

FIGURE 12. The convergence cure of the involved algorithms.

by Holm's procedure for multiple comparisons are summarized in Table 14. The results in Table 14 show that all the p-values are lower than 0.05, which indicates that CPA-HDM is significantly different from all participating algorithms.

For demonstrating the convergence performance of CPA-HDM, the convergence analysis of the participant algorithms is analyzed with t_{av} and convergence curves. The average computation time t_{av} of 16 instances in 20 times runs is shown in Table 15, and the graphical representation of the convergence analysis is depicted in Fig. 13. The T_{avg} in Table 15 denotes the average time of t_{av}, the x-axis in Fig. 12 is taken as the running time of the algorithm and the y-axis is taken as the length of the route.

Table 15 shows that CPA-HDM wins on 8 instances, whereas DSFLA performs better on att48 and pr107, DBAL performs better on pr124 and pr264, ABC performs better on ch130 and kroA150, and PACO-3Opt performs better on kroA100 and kroB150. The T_{avg} of CPA-HDM achieved the shortest among seven algorithms, which means that the proposed algorithm exhibits superior performance.

The figures depicted in Fig. 12 also confirm the efficiency of the CPA-HDM, as it achieved the fastest convergence speed among various comparison algorithms on six instances. Although CPA-HDM converges slower than D-GWO on six instances and DSFLA on pr299 in the early stage, it could keep the convergence speed at the highest level among the involved algorithms in the middle and later iteration for its balance the exploration and exploitation ability.

Based on the analyses above, it is clear that the CPA-HDM exhibits superior accuracy and high convergence speed in solving TSP, which has an outstanding performance with different scale instances compared with the other six algorithms.

V. CONCLUSION AND FUTURE WORK

A real-coded CPA with a heuristic decoding method, named CPA-HDM, is proposed in this paper to solve TSP, which incorporates several useful components. The CPA-HDM presents a heuristic decoding method to map continuous variables to discrete ones, the method both considers the continuous variables and the distance between cities. After that, the AAP, the IGMOCP & IPUMOP are proposed in the growth phase to address the poor performance of balancing the exploration and exploitation ability in CPA. Also, the IRS is redesigned, which allows all carnivorous plants to reproduce and reduces the possibility of search stagnation. Finally, the ACPLS is incorporated into the algorithm, the double-bridge exchange is employed in the early iteration, the neighborhood 2-opt exchange is employed in the late iteration, and the local search algorithm is employed to promote the convergence speed.

To verify the performance of the proposed algorithm and its improvements, several instances from TSPLIB have been solved. The results show that the CPA-HDM could converge quickly towards the optimal solutions for most of the instances selected. From the statistics, the following conclusions can be drawn: 1) the proposed decoding method can direct the produced discrete solutions toward optimality compared with the order-based arrangement and rounding method; 2) Using ACPLS, AAP, IGMOCP& IPUMOP, and IRS exhibit a positive role in the performance of the algorithm; 3) The CPA-HDM performs highest solution precision, robust, and convergence speed of its balanced exploration and exploitation capabilities compared with the other participating algorithms.

The CPA-HDM is only proven on the symmetric TSP, some practical applications are still needed to test its effectiveness. Besides, to further improve the performance of CPA and expand its application fields, the permutation-coded carnivorous plant algorithm, decoding method of the real-coded carnivorous plant algorithm for solving TSP, the research of hybrid carnivorous plant algorithm, and applied research, such as the flexible job-shop scheduling, route optimization, and cross-region work problem of agricultural machinery with time window can be carried out in the future.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their valuable and constructive comments that greatly helped improve the quality and completeness of this paper.

REFERENCES

[1] K. Savla, E. Frazzoli, and F. Bullo, "Traveling salesperson problems for the Dubins vehicle," *IEEE Trans. Autom. Control*, vol. 53, no. 6, pp. 1378–1391, Jul. 2008.

[2] Y. Zhang, Z. Song, J. Yuan, Z. Deng, H. Du, and L. Li, "Path optimization of gluing robot based on improved genetic algorithm," *IEEE Access*, vol. 9, pp. 124873–124886, 2021.

[3] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà, "A hybrid genetic algorithm for the traveling salesman problem with drone," *J. Heuristics*, vol. 26, no. 2, pp. 219–247, Apr. 2020.

[4] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transp. Sci.*, vol. 52, no. 4, pp. 965–981, Aug. 2018.

[5] D. Whitley, T. Starkweather, and D. Shaner, *The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination*, 1991.

[6] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.

[7] T. Eldos, A. Kanan, W. Nazih, and A. Khatatbih, "Adapting the ant colony optimization algorithm to the printed circuit board drilling problem," *World Comput. Sci. Inf. Technol. J.*, vol. 3, no. 5, pp. 100–104, 2013.

[8] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theor. Comput. Sci.*, vol. 4, pp. 237–244, Jun. 1977.

[9] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 2015.

[10] M. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," *Oper. Res. Lett.*, vol. 6, no. 1, pp. 1–7, Mar. 1987.

[11] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *J. Oper. Res. Soc. Amer.*, vol. 2, no. 4, pp. 393–410, 1954.

[12] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "An analysis of several heuristics for the traveling salesman problem," *SIAM J. Comput.*, vol. 6, no. 3, pp. 563–581, Sep. 1977.

[13] D. S. Johnson, "Local optimization and the traveling salesman problem," in *Proc. Int. Colloq. Automata, Lang., Program.*, vol. 443, Jun. 1990, pp. 446–461.

[14] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local Search Combinat. Optim.*, vol. 1, no. 1, pp. 215–310, 1997.

[15] Y. Deng, J. Xiong, and Q. Wang, "A hybrid cellular genetic algorithm for the traveling salesman problem," *Math. Problems Eng.*, vol. 2021, pp. 1–16, Feb. 2021.

[16] J. Wang, O. Ersoy, M. He, and F. Wang, "Multi-offspring genetic algorithm and its application to the traveling salesman problem," *Appl. Soft Comput.*, vol. 43, pp. 415–423, Jun. 2016.

[17] M. Mi, X. Huifeng, Z. Ming, and G. Yu, "An improved differential evolution algorithm for TSP problem," in *Proc. Int. Conf. Intell. Comput. Technol. Automat.*, vol. 1, May 2010, pp. 544–547.

[18] O. Eneko, S. J. Del, S. Ali, B. M. Nekane, and C. D. Avid, "A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem," *Appl. Soft Comput.*, vol. 71, Oct. 2018, Art. no. S1568494618303818.

[19] S. R. Balachandar and K. Kannan, "Randomized gravitational emulation search algorithm for symmetric traveling salesman problem," *Appl. Math. Comput.*, vol. 192, no. 2, pp. 413–421, Sep. 2007.

[20] K. Yang, X. You, S. Liu, and H. Pan, "A novel ant colony optimization based on game for traveling salesman problem," *Int. J. Speech Technol.*, vol. 50, no. 12, pp. 4529–4542, Dec. 2020.

[21] Q. Liu, S. Du, B. J. van Wyk, and Y. Sun, "Niching particle swarm optimization based on Euclidean distance and hierarchical clustering for multimodal optimization," *Nonlinear Dyn.*, vol. 99, no. 3, pp. 2459–2477, Feb. 2020.

[22] Y. Saji and M. E. Riffi, "A novel discrete bat algorithm for solving the travelling salesman problem," *Neural Comput. Appl.*, vol. 27, no. 7, pp. 1853–1866, 2016.

[23] K. Panwar and K. Deep, "Discrete Grey Wolf optimizer for symmetric travelling salesman problem," *Appl. Soft Comput.*, vol. 105, no. 11, Jul. 2021, Art. no. 107298.

[24] K. M. Ong, P. Ong, and C. K. Sia, "A carnivorous plant algorithm for solving global optimization problems," *Appl. Soft Comput.*, vol. 98, Jan. 2021, Art. no. 106833.

[25] M. Mukherjee and B. K. S. Roy, "Optimal μPMU placement in distribution network using binary carnivorous plant algorithm," in *Proc. IEEE 2nd Int. Conf. Smart Technol. Power, Energy Control (STPEC)*, Dec. 2021, pp. 1–6.

[26] M. Akhand, S. I. Ayon, S. A. Shahriyar, N. Siddique, and H. Adeli, "Discrete spider monkey optimization for traveling salesman problem," *Appl. Soft Comput.*, vol. 86, Jan. 2019, Art. no. 105887.

[27] I. Khan, M. K. Maiti, and K. Basuli, "Multi-objective traveling salesman problem: An ABC approach," *Appl. Intell.*, vol. 50, no. 11, pp. 3942–3960, Nov. 2020.

[28] I. Khan and M. K. Maiti, "A swap sequence based artificial bee colony algorithm for traveling salesman problem," *Swarm Evol. Comput.*, vol. 44, pp. 428–438, Feb. 2019.

[29] E. Osaba, X.-S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo, "An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems," *Eng. Appl. Artif. Intel.*, vol. 48, pp. 59–71, Feb. 2016.

[30] Z. Zhang and Y. Han, "Discrete sparrow search algorithm for symmetric traveling salesman problem," *Appl. Soft Comput.*, vol. 118, Mar. 2022, Art. no. 108469.

[31] F. S. Gharehchopogh and B. Abdollahzadeh, "An efficient Harris Hawk optimization algorithm for solving the travelling salesman problem," *Cluster Comput.*, vol. 25, pp. 1981–2005, May 2021.

[32] A. E.-S. Ezugwu, A. O. Adewumi, and M. E. Frîncu, "Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem," *Expert Syst. Appl.*, vol. 77, pp. 189–210, Jul. 2017.

[33] A. E.-S. Ezugwu and A. O. Adewumi, "Discrete symbiotic organisms search algorithm for travelling salesman problem," *Expert Syst. Appl.*, vol. 87, pp. 70–78, Nov. 2017.

[34] Y. Wang, Y. W. Wu, and N. Xu, "Discrete symbiotic organism search with excellence coefficients and self-escape for traveling salesman problem," *Comput. Ind. Eng.*, vol. 131, pp. 269–281, May 2019.

[35] L. T. Kóczy, P. Földesi, and B. Tüu-Szabó, "Enhanced discrete bacterial memetic evolutionary algorithm—An efficacious metaheuristic for the traveling salesman optimization," *Inf. Sci.*, vols. 460–461, pp. 389–400, Sep. 2018.

[36] J. Zhang, L. Hong, and Q. Liu, "An improved whale optimization algorithm for the traveling salesman problem," *Symmetry*, vol. 13, no. 1, p. 48, Dec. 2020.

[37] C. Wu, X. Fu, J. Pei, and Z. Dong, "A novel sparrow search algorithm for the traveling salesman problem," *IEEE Access*, vol. 9, pp. 153456–153471, 2021.

[38] Y. Saji and M. Barkatou, "A discrete bat algorithm based on Lévy flights for Euclidean traveling salesman problem," *Expert Syst. Appl.*, vol. 172, Jun. 2021, Art. no. 114639.

[39] M. Gunduz and M. Aslan, "DJAYA: A discrete Jaya algorithm for solving traveling salesman problem," *Appl. Soft Comput.*, vol. 105, Jul. 2021, Art. no. 107275.

[40] Y. Huang, X. N. Shen, and X. You, "A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem," *Appl. Soft Comput.*, vol. 102, no. 2, Apr. 2021, Art. no. 107085.

[41] W. Han, Q. Deng, G. Gong, L. Zhang, and Q. Luo, "Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint," *Expert Syst. Appl.*, vol. 168, Nov. 2021, Art. no. 114282.

[42] C. Yu, P. Andreotti, and Q. Semeraro, "Multi-objective scheduling in hybrid flow shop: Evolutionary algorithms using multi-decoding framework," *Comput. Ind. Eng.*, vol. 147, Jul. 2020, Art. no. 106570.

[43] F. Samanlioglu, W. G. Ferrell, and M. E. Kurz, "A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem," *Comput. Ind. Eng.*, vol. 55, no. 2, pp. 439–449, 2008.

[44] I. M. Ali, D. Essam, and K. Kasmarik, "A novel design of differential evolution for solving discrete traveling salesman problems," *Swarm Evol. Comput.*, vol. 52, Feb. 2020, Art. no. 100607.

[45] S. K. R. Kanna, K. Sivakumar, and N. Lingaraj, "Development of deer hunting linked earthworm optimization algorithm for solving large scale traveling salesman problem," *Knowl.-Based Syst.*, vol. 227, Sep. 2021, Art. no. 107199.

[46] G. A. Croes, "A method for solving traveling-salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, Dec. 1958.

[47] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, Oct. 1973.

[48] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Discrete cuckoo search algorithm for the travelling salesman problem," *Neural Comput. Appl.*, vol. 24, nos. 7–8, pp. 1659–1669, Apr. 2014.

[49] H. Song, H. Song, J. Wang, L. Song, H. Zhang, J. Bei, J. Ni, and B. Ye, "Improvement and application of hybrid real-coded genetic algorithm," *Appl. Intell.*, vol. 2022, pp. 1–39, Apr. 2022.

[50] M. Friedman, "A comparison of alternative tests of significance for the problem of *m* rankings," *Ann. Math. Statist.*, vol. 11, pp. 86–92, Mar. 1940.

[51] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the Friedman statistic," *Commun. Statist.-Theory Methods*, vol. 9, no. 6, pp. 571–595, 1980.

[52] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandin. J. Statist.*, vol. 6, pp. 65–70, Jan. 1979.

[53] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[54] C. Wu and X. Fu, "An agglomerative greedy brain storm optimization algorithm for solving the TSP," *IEEE Access*, vol. 8, pp. 201606–201621, 2020.

[55] G. Şaban, M. Mostafa, B. O. Kaan, and H. Kodaz, "A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem," *Soft Comput.*, vol. 22, no. 5, pp. 1669–1685, Mar. 2018.

JIQUAN WANG received the B.E. and M.E. degrees in agricultural electrification and automation from Northeast Agricultural University, China, in 1996 and 2004, respectively, and the Ph.D. degree in agricultural equipment engineering technology from Shenyang Agricultural University, China, in 2011.

He is currently a Professor with the Engineering College, Northeast Agricultural University. He published over 40 papers in domestic and international academic journals and conference proceedings. His current research interests include genetic algorithm theory and its application, neural network theory and its application, parallel computing and image recognition, and intelligent optimization algorithm.

PANLI ZHANG received the B.E. degree in logistic management from Chongqing Jiaotong University, China, in 2017, and the M.E. degree in industrial engineering from Northeast Agricultural University, China, in 2019, where she is currently pursuing the Ph.D. degree with the Engineering College. Her current research interests include genetic algorithm theory and its application and deep learning theory and its application.

HONGYU ZHANG received the Management degree from the Hebei University of Architecture, in 2019. She is currently pursuing the master's degree with the Engineering College, Northeast Agricultural University. Her current research interests include genetic algorithm theory and its application, image segmentation, and recognition.

HAOHAO SONG received the B.E. degree in logistic management from the North China University of Water Resources and Electric Power, China, in 2019, and the M.E. degree in agricultural system engineering and management engineering from Northeast Agricultural University, China, in 2022, where she is currently pursuing the Ph.D. degree with the Engineering College. Her current research interest includes genetic algorithm theory and its application.

JINLING BEI received the B.E. degree in industrial engineering from Northeast Agricultural University, China, in 2019, where she is currently pursuing the master's degree with the Engineering College. Her current research interests include genetic algorithm theory and its application, image segmentation, and recognition.

WENFENG SUN received the B.E. degree in agricultural electrification and automation, the M.E. degree in agricultural mechanization and automation, and the Ph.D. degree in agricultural systems engineering and management engineering from Northeast Agricultural University, China, in 1995, 2009, and 2022, respectively.

He is currently a Researcher with the Engineering College, Northeast Agricultural University. He published over 40 papers in domestic and international academic journals and conference proceeding. His current research interests include agricultural systems engineering and research on high-efficiency plant protection machinery.

XIAOBO SUN received the B.E. degree in agricultural mechanization and automation and the M.E. and Ph.D. degrees in agricultural mechanization engineering from Northeast Agricultural University, China, in 2016, 2019, and 2022, respectively.

He is currently a Researcher with the Engineering College, Northeast Agricultural University. He published over ten papers in domestic and international academic journals and conference proceedings. His current research interest includes intelligent agricultural equipment and technology.