mathematics

MDPI

Article

The Discrete Carnivorous Plant Algorithm with Similarity Elimination Applied to the Traveling Salesman Problem

Pan-Li Zhang †, Xiao-Bo Sun †, Ji-Quan Wang *, Hao-Hao Song, Jin-Ling Bei and Hong-Yu Zhang

College of Engineering, Northeast Agricultural University, Harbin 150030, China
* Correspondence: wang-jiquan@163.com
† These authors contributed equally to this work.

Abstract: The traveling salesman problem (TSP) widely exists in real-life practical applications; it is a topic that is under investigation and presents unsolved challenges. The existing solutions still have some challenges in convergence speed, iteration time, and avoiding local optimization. In this work, a new method is introduced, called the discrete carnivorous plant algorithm (DCPA) with similarity elimination to tackle the TSP. In this approach, we use a combination of six steps: first, the algorithm redefines subtraction, multiplication, and addition operations, which aims to ensure that it can switch from continuous space to discrete space without losing information; second, a simple sorting grouping method is proposed to reduce the chance of being trapped in a local optimum; third, the similarity-eliminating operation is added, which helps to maintain population diversity; fourth, an adaptive attraction probability is proposed to balance exploration and the exploitation ability; fifth, an iterative local search (ILS) strategy is employed, which is beneficial to increase the searching precision; finally, to evaluate its performance, DCPA is compared with nine algorithms. The results demonstrate that DCPA is significantly better in terms of accuracy, average optimal solution error, and iteration time.

Keywords: discrete carnivorous plant algorithm; traveling salesman problem; simple sorting grouping; similarity elimination; iterative local search

MSC: 68R05; 68W50; 68Q07

Citation: Zhang, P.-L.; Sun, X.-B.; Wang, J.-Q.; Song, H.-H.; Bei, J.-L.; Zhang, H.-Y. The Discrete Carnivorous Plant Algorithm with Similarity Elimination Applied to the Traveling Salesman Problem. *Mathematics* **2022**, *10*, 3249. https://doi.org/10.3390/math10183249

Academic Editor: Ioannis G. Tsoulos

Received: 24 August 2022
Accepted: 5 September 2022
Published: 7 September 2022

1. Introduction

Combinatorial optimization problems [1] explore extreme values in discrete areas, which commonly exists in numerous fields such as agriculture, industry, national defense, engineering, transportation, finance, energy, and communication. Many of these problems belong to NP-hard [2], in which the spatial scale, time scale, and complexity of the solution increase exponentially as the scale of the problem grows. Therefore, many current methods are not sufficient in solving this problem. The traveling salesman problem (TSP) is one of the most extensively studied combinatorial optimization problems in operation research. The TSP and its various variants are not only famous theoretical problems but also important practical application problems. It can be applied to various fields, including scheduling and drilling holes in printed circuit boards in manufacturing [3]; designing reasonable traffic roads to avoid traffic congestion [4]; planning the best transportation route to maximize the benefits to the company [5]; laying out the appropriate location for a router so that the information transmission efficiency is the strongest [6], and many other areas [7,8]. Usually, a minor improvement in solution quality or a reduction in execution time in these areas can save millions of dollars or significantly develop productivity, bringing enormous economic benefits to enterprises and society. In addition, the TSP is commonly adopted as a measurement tool of algorithms' performances. Therefore, the research of the TSP has high theoretical value and practical significance, and its solution method has attracted the attention of scholars in recent years.

The TSP has only simple loop constraints, so it is relatively simple. However, as a typical NP-hard problem, the calculation difficulty increases exponentially as the city number grows. The method for TSP can broadly divide into exact algorithms and heuristic algorithms. Exact algorithms, such as linear programming [9], dynamic programming [10], and branch and bound [11], find the optimal solution by traveling all solution spaces with exponential time complexity. Thus, the exact algorithms have been insufficient to satisfy the solution of a TSP with a large size. Heuristic algorithms, such as the nearest neighbor [12] and local search algorithms [13,14], can find the suboptimal solution or the optimal solution with a certain probability. However, they are prone to fall into local optimums. To solve such problems, some scholars have innovatively proposed an important branch of heuristic algorithms, called metaheuristic algorithms, which involve the ant colony algorithm (ACO) [15,16], particle swarm optimization (PSO) algorithm [17], genetic algorithm (GA) [18–20], differential evolution (DE) [21], sparrow search algorithm [22], artificial bee algorithm [23], etc. They solve large-scale instances by obtaining inspiration through the cognition of relevant behaviors, principles, and action mechanisms in biology, physics, chemistry, and other fields and designing and using heuristic rules to search the solution space. The metaheuristic algorithm is widely applied in optimization, as it can obtain a high-quality satisfactory solution in a reasonable time, which greatly enriches the algorithm system.

Most metaheuristics algorithms were created for continuous optimization problems and cannot be directly applied to solve TSPs with discrete properties [24]. To explore the performance of such algorithms in the TSP, many scholars have conducted research and proposed a variety of discrete methods. References [25,26] employed swap sequence and swap operator, reference [27] adopted swap, shift, and symmetry operators to alternate the update method of the algorithm, and references [28,29] introduced the Hamming distance and a local search algorithm to updating individuals. However, among the above three update methods, the first two methods lack heuristics and exhibit poor convergence speed. The last one lacks exploration ability and easily to falls into stagnation. References [30,31] applied the rank-order and rounding decoding methods, respectively. Although the above two methods are simple to implement, they are not heuristic and may have a negative effect on the solution quality of its randomicity. Researchers such as Kenneth Sörensen [32] think that in the field of optimization computing it is important not to propose new algorithms but to establish universal optimization-algorithm-applicable rules and strategies and research common problems in optimization problems and optimization algorithms. Therefore, it is necessary to design a new universal discrete method or improve existing methods to enhance the algorithms' performance when solving the TSP.

The carnivorous plant algorithm (CPA) is a new swarm intelligence optimization algorithm proposed by Meng et al. [33], and the algorithm has been verified to successfully solve high-dimensional continuous problems. To the best of our knowledge, CPA has not been adopted to solve discrete optimization problems. In addition, the grouping method of CPA easily results in an imbalance in search ability among subgroups, which has a negative influence on the search performance of the algorithm. The growth of carnivorous plants or the updates of prey depend on the attraction probability, and the attraction probability in CPA is a constant that cannot improve the balance between exploration and the exploitation ability. Therefore, CPA performs convergence prematurely and is easy to trap into a local optimum. For optimization problems, when the algorithm finds a region with an extreme value, individuals continue to approach this extreme point. As the iteration time grows, the number of identical or similar individuals in the population increases. If the two individuals selected for the update are highly similar or identical, the possibility of excellent offspring being generated is reduced, which affects the convergence speed. The limitations of the discrete method for TSPs and the promising results achieved by CPA in continuous problems have formed the main motivation for this paper.

To address the above problems, in this work a new type of individual generation method is designed, using redesigned subtraction, multiplication, and addition operators

to satisfy the legitimacy of the TSP solution. The method combines set difference operation, optimal operation, three crossover operators, and symmetry transportation to maintain the exploration and exploitation abilities of the algorithm. Then, a simple sorting grouping method is proposed, which can promote the speed of grouping, reduces the rate of assimilation of the population, and heightens the exploitation ability. After that, an adaptive attraction probability is designed, which enables the prey to update the position in the early stage with a high probability and heightens the exploration ability. Carnivorous plants grow in the late stage with a high probability to strengthen the exploitation ability. In addition, to maintain population diversity and avoid the situation that affects the efficiency of the algorithm, the similarity-eliminating operation, comprising the same quantity of city sequences and route length, is designed. The former is applied to eliminate the individuals who participate in the update with high similarity, and the latter is applied to eliminate the same individuals in the population. Finally, the iteration local search algorithm (ILS) is introduced to jump out of the local optimum and find the global theoretical optimum.

The primary contributions of this work are:

(1) A new method of carnivorous plant growth and prey updating for TSP is designed to ensure that the algorithm can switch from continuous space to discrete space without losing information.
(2) A simple sorting grouping method is proposed to reduce the computational complexity of the algorithm.
(3) An adaptive attraction probability is presented to balance the exploitation and exploration ability.
(4) The similarity-eliminating operation is added to maintain population diversity and extend the search space.
(5) ILS is adopted to improve search precision and reduce the probability of individual stagnation.
(6) To verify the validity of the discrete carnivorous plant algorithm (DCPA), nine algorithms are conducted as comparison algorithms, and the results on 34 instances from TSPLIB show the superior performance of the proposed algorithm.

The remaining part of this work proceeds as follows: Section 2 gives a brief overview of the TSP and the CPA; Section 3 describes the formula of the TSP; Section 4 presents a discrete framework of the proposed DCPA; Section 5 is an account of the simulation experiment and data analysis; and Section 6 gives a conclusion.

2. Related Works

2.1. Literature Review

Researchers have proposed various algorithms over the years to solve the TSP, which can be broadly grouped into permutated coded algorithms and real coded algorithms.

Among the permutated coded algorithms, the ACO and GA are suitable for solving discrete optimization problems, as their update method can directly generate legitimate offspring. Stodola et al. [34] proposed an adaptive ant colony optimization algorithm with nod clustering (AACO-NC). In AACO-NC, three techniques are adopted to enhance the algorithm's performance. First, the node clustering principle is adopted to decrease the optimization time; second, the adaptive pheromone evaporation coefficient based on the population's diversity is employed to extend the search space; and third, a new termination condition based on entropy is proposed to decrease the inaccuracy of the iteration number setting. Yong W et al. [35] presented a hybrid genetic algorithm with two local optimization algorithms. One is applied to the local Hamiltonian paths to enhance the solution quality, and another is applied to prevent falling into the local optimum. Q. M et al. [36] proposed a hybrid genetic algorithm with a splitting algorithm when solving the traveling salesman problem with a drone.

Several algorithms are only applicable for continuous optimization problems. It is necessary to modify the update method to maintain the discrete characteristics when solving TSPs. Wang et al. [37] improved a discrete symbiotic organism search with an

excellent coefficient and self-escape strategy. The former helps to enhance the exploitation capability, and the latter helps to maintain population diversity. Eneko et al. [28] proposed a discrete water cycle algorithm where the Hamming distance is adopted to measure the difference between two individuals and the insert and 2-Opt operators are adopted as movement operators to avoid the local optimum. Kóczy et al. [38] presented a discrete version of the bacterial memetic evolutionary algorithm (DBMEA). In DBMEA, three nearest neighbor heuristic methods and a random creation method are employed to generate the initial population, two local search algorithms are introduced to improve the search precision, and a combined gene transfer operation maintains population diversity. Zhong et al. [39] developed a discrete pigeon-inspired optimization algorithm (DPIO). A new map and landmark operator with learning ability are employed in DPIO. The former helps to heighten the exploration ability, and the latter helps to enhance the exploitation ability. The metropolis acceptance criterion in the proposed algorithm helps to avoid stagnation. A discrete bat algorithm with Lévy flight (DBAL) was designed by Saji et al. [40], where a new updating method is proposed to solve TSP, and the crossover operator and three local search algorithms are employed to enhance the searching precision. A. Benyamin et al. [41] designed a discrete farmland fertility optimization algorithm (DFFA). In DFFA, the swap, inversion, and insertion operators are employed to expand the search space, the metropolis acceptance strategy is employed to prevent prematurely accepting a solution, and a crossover operator is adopted to maintain the features of the standard algorithm. G. H. Al-Gaphari et al. [42] proposed a crow-inspired algorithm with three new discrete methods to map the continuous variables into discrete variables, which are: the modular arithmetic and set theory, basic operators, and dissimilarity operation. Z. Zhang and J. Yang [43] developed a discrete cuckoo search algorithm with random work and a local adjustment operator for the TSP. The former is utilized to maintain population diversity, and the latter is adopted to promote the convergence rate.

Some algorithms need to introduce the mapping method to transform continuous variables into discrete variables. Samanlioglu et al. [44] improved a random-key genetic algorithm with the ranked-order value decoding for the TSP. Ezugwu et al. [45] adopted the rounding method and restructured symbiotic organism search by incorporating swap, insert, and inverse operators. Ali et al. [46] adopted the rank-order and best-matched decoding in a novel discrete differential evolution, where the *k*-means clustering-based repairing method is employed in the algorithm to improve the individuals' quality, and a combined mutation is introduced to maintain the population diversity. F. S. Gharehchopogh and B. Abdollahzadeh [47] employed random-key encoding in the Harris hawk optimization algorithm, where a mutation operator, the metropolis acceptance strategy, and a local search algorithm are adopted in the algorithm to maintain population diversity, prevent prematurely accepting a solution, and enhance the solution quality, respectively. Zhang et al. [30] applied the order-based arrangement to map continuous variables into discrete ones in the discrete sparrow search algorithm (DSSA). In the DSSA, Gaussian mutation and a swap operator are adopted to maintain population diversity, and the 2-Opt algorithm is adopted to enhance search precision.

2.2. Standard Carnivorous Plant Algorithm

CPA is a metaheuristic algorithm that simulates the whole process of carnivorous plants' attraction, predation, and digestion. The algorithm is mainly composed of four stages: the grouping, growth, reproduction, and recombination phases.

2.2.1. Grouping Phase

Let the population size be n and rank the individuals according to the fitness value from small to large. The best (n_1 individuals) are regarded as carnivorous plants, and the rest (n_2 individuals) are regarded as prey (n_1 can be arbitrarily determined, as long as $n_2 > n_1$, $n_2 + n_1 = n$ is satisfied and n_2 can be divisible by n_1). The population is divided into n_1 groups, and individuals in each subgroup are comprised of one carnivorous plant

and n_2/n_1 prey. Table 1 presents the grouping process in the CPA at a population size of 12, where the population before ordering is $X = (X_1, X_2, \ldots, X_{n1+n2})$, after ordering is $X' = (X'_1, X'_2, \ldots, X'_{n+n})$, and the fitness of individuals satisfies $F(X'_1) \leq F(X'_2) \leq \ldots \leq F(X'_{n1+n2})$. The number of carnivorous plants (n_1) is understood to be 3 in this example, so that the three best individuals in the population are regarded as carnivorous plants, and the remaining individuals are prey. In Table 1, prey X'_4, X'_5, and X'_6 are attracted by carnivorous plants X'_1, X'_2, and X'_3, respectively. Then, prey X'_7, X'_8, and X'_9 are attracted by carnivorous plants X'_1, X'_2, and X'_3, respectively. The process is repeated until the n_2-th prey is attracted by the n_1-th carnivorous plant.

Table 1. The grouping process at size 12 in the CPA.

Before Sorting	After Sorting	Group	Carnivorous Plants	Prey		
X_1	X'_1	Group 1	X'_1	X'_4	X'_7	X'_{10}
X_2	X'_2					
X_3	X'_3					
X_4	X'_4	Group 2	X'_2	X'_5	X'_8	X'_{11}
X_5	X'_5					
X_6	X'_6					
X_7	X'_7	Group 3	X'_3	X'_6	X'_9	X'_{12}
X_8	X'_8					
X_9	X'_9					
X_{10}	X'_{10}					
X_{11}	X'_{11}					
X_{12}	X'_{12}					

2.2.2. Growth Phase

The attraction rate is adopted in the CPA. While the attraction probability γ ($\gamma = 0.8$) is greater than a random number between [0, 1], the individual executes the growth formula of carnivorous plants:

$$Xnew_i = P_{iv} + \alpha \otimes (X_i - P_{iv}) \tag{1}$$

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

where $\otimes$ indicates that two vectors are multiplied by elements at the same position, X_i indicates the carnivorous plants in the i-th subgroup, P_{iv} indicates the v-th prey in the i-th subgroup, *rand* indicates a random vector with m-dimensional components and each component is uniformly distributed in the range [0, 1], m is the dimension number in the individual, and gr indicates the growth rate, which is equal to 2.

While γ is less than the random number, the individual executes the update formula of grey:

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

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

where $\otimes$ indicates that two vectors are multiplied by elements at the same position, P_{iu} and P_{iv} indicate the u-th and v-th prey in the i-th subgroup, respectively, *rand* indicates a random vector with m-dimensional components and each component is uniformly distributed in the range [0, 1], m is the dimension number in the individual, and $f(P_{iu})$ and $f(P_{iv})$ indicate the fitness values of the u-th and v-th prey, respectively.

2.2.3. Reproduction Phase

In the CPA, only the best individual can execute the reproduction operation. Its mathematical model is as follows:

$$Xnew_i = \begin{cases} X_1 + \beta * rand \otimes (X_v - X_i) & f(X_i) > f(X_v) \\ X_1 + \beta * rand \otimes (X_i - X_v) & f(X_i) < f(X_v) \end{cases} \quad (5)$$

$$\beta = \mu * rand \quad (6)$$

where $\otimes$ indicates that two vectors are multiplied by elements at the same position, X_1 indicates the best individual, X_i and X_v indicate carnivorous plants in the i-th and v-th subgroup, respectively, the "*rand*" is a random vector with m-dimensional components and each component is uniformly distributed in the range [0, 1], m is the dimension number in the individual, $f(X_v)$ and $f(X_i)$ indicate the fitness values of the i-th and v-th carnivorous plants, respectively, and μ indicates the reproduction rate, which is equal to 1.8.

2.2.4. Recombination Phase

Recombination refers to merging the population before and after updating into a large population and calculating and ranking the fitness from small to large. n best individuals are selected from the large population.

The procedure of standard carnivorous plant algorithms is summarized in Algorithm 1, where n is the population size, n_1 is the number of carnivorous plants, n_2 is the number of prey, *iter* is the current iteration number, and *Maxiter* is the maximum iteration number.

Algorithm 1. The standard CPA

Initialize the relevant parameters;
Generate n initial individuals in the population;
Calculate and sort the fitness value;
iter = 1;
While *iter* < *Maxiter*
  iter = *iter* +1;
  Regard the n_1 best individuals as carnivorous plants, the remaining n_2 individuals as prey, and sort into subgroups as shown in Table 1;
  Xnew and *Pnew* are updated with Equations (1) and (3);
  Xnew is updated with Equation (5);
  Xnew, *Pnew*, *X*, and *P* are combined to form a new population;
  The new population is sorted by its fitness, and n best individuals are selected;
End While
Output the best solution and its fitness value;

3. Problem Formulation

The TSP refers to a salesman starting from a city, passing through cities 1, 2, . . . , m to deliver goods, and eventually returning to the starting city. Since the distance from each city to other cities is different and the total path lengths corresponding to the sequences of different routes are different, it is necessary to design the sequence of cities and ensure that the final journey of the salesman is the shortest and each city is only visited once. If the city size is slight, the problem is easy to calculate, but with the city size expansion, it grows complex. The TSP can be also defined as an undirected weighted complete graph, $G = (V, A)$, where V is the set of m vertices and A is the set of edges. For each edge, $(i, j) \in A(i, j \in V)$. The modeling of the TSP can be expressed as:

$$Minf(c) = \sum_{i=1}^{m} \sum_{j=1}^{m} z_{ij} c_{ij} \quad (7)$$

where f indicates the distance traveled by the salesman in the TSP, c_{ij} represents the distance between city i and city j, z_{ij} indicates the decision variable, and its formula is as follows:

$$z_{ij} = \begin{cases} 1 & \textit{if city } i \textit{ is visited next to } j \\ 0 & \textit{Otherwise} \end{cases} \tag{8}$$

The calculation formula of c_{ij} is:

$$c_{ij} = \sqrt{(h_i - h_j)^2 + (y_i - y_j)^2} \tag{9}$$

where (h_i, y_i) and (h_j, y_j) indicate the coordinates of the i-th and j-th city, respectively.

4. The DCPA for TSP

A discrete carnivorous plant algorithm (DCPA) for the TSP is proposed in this section, where a novel individual-generated method and several improved components are introduced.

4.1. Individual-Generated Method

The method of generating initial solutions by the CPA is unsuitable for the TSP. Variables in the CPA are continuous, while the solutions in the TSP are discrete. For the TSP with m cities, each route is composed of m random integers without repetition, which are between [1, m]. Therefore, the encoding method of the DCPA employs permutation encoding.

The CPA is suitable for continuous optimization problems. For combinatorial optimization problems such as the TSP, the solution update method should be redesigned, so that it cannot only conform to the properties of TSP but also retain the good characteristics of the standard update method. Therefore, the subtraction, multiplication, and addition operations in CPA are redefined. The details are presented from Sections 4.1.1–4.1.3.

4.1.1. Redefining the Subtraction Operation

From Section 2.2.1, the population in the algorithm has been divided into carnivorous plants and prey. Randomly select an individual from the carnivorous plants and regard it as X_i. Randomly select an individual from the prey and regard it as P_{iv}. Let $U = X_i - P_{iv}$, $X_i = (x_i^1, x_i^2, \ldots, x_i^m)$, and $P_{iv} = (x_{iv}^1, x_{iv}^2, \ldots, x_{iv}^m)$. t is set as the index of X_i ($t \in [1, 2, \ldots, m]$), x_i^m indicates the m-th city in X_i, x_{iv}^m indicates the m-th city in P_{iv}, and $rr = 1$. The main steps of $X_i - P_{iv}$ are:

Step 1: Let $t = rr$. Then, $X_i(t) = x_i^t$, and the index, ε, is found of x_i^t in P_{iv}. If $\varepsilon = m$, then $\varepsilon = 0$;

Step 2: $X_i(t+1) = x_i^{t+1}$. x_{iv}^u is found on $P_{iv}(\varepsilon + 1)$ in P_{iv};

Step 3: If $x_i^{t+1} = x_{iv}^u$, $s_t = (0, 0)$. Otherwise, $s_t = (x_i^{t+1}, x_{iv}^u)$;

Step 4: Let $r = r + 1$. If $t > m - 1$, turn to Step 5. Otherwise, Step 1–Step 4 are repeated;

Step 5: Let $t = m$. Then, $X_i(t) = x_i^m$, and the index, θ, is found of x_i^m in P_{iv}. If $\theta = m$, then $\theta = 0$. $X_i(1) = x_i^1$ and x_{iv}^r is found on $P_{iv}(\theta+1)$ in P_{iv}. If $x_i^1 = x_{iv}^r$, $s_t = (0, 0)$. Otherwise, $s_t = (x_i^1, x_{iv}^r)$;

Step 6: Let $S = \{s_1, s_2, \ldots, s_m\}$;

Step 7: The variables in X_i, which are different from those in S, are assigned to 0 to obtain a new vector, X_i^0, and $U = X_i^0$;

For a clearer explanation, set an example with $X_i = (6, 3, 5, 1, 2, 4)$ and $P_{iv} = (5, 1, 2, 3, 4, 6)$. From Step 1 to Step 5, $s_1 = (6, 3)$, $s_2 = (3, 5)$, $s_3 = (0, 0)$, $s_4 = (0, 0)$, $s_5 = (2, 4)$, and $s_6 = (0, 0)$ can be obtained. Then, $S = (6, 3, 3, 5, 0, 0, 0, 0, 2, 4, 0, 0)$ and $U = (6, 3, 5, 0, 2, 4)$ according to Step 6–Step 7.

It can be seen from the above description that the redefined subtraction operation employed the concept of the set difference operation, which can make the offspring have the inherent characteristics of the better individuals in the parents and make the subtraction have an inheritance.

4.1.2. Redefining the Multiplication Operation

r is a random number between [0, 2]. If $r \geq 1$, $U' = U$, and U is obtained by Section 4.1.1; otherwise, the calculation steps of U' are:

Step 1: Let U' be a zero vector with an m-dimensional;

Step 2: $V' = (l_1, l_2, \ldots, l_{cc})$, which is a vector after deleting zero elements in U, where cc is the number of nonzero components in U. The distance between adjacent cities in V' is marked as d, where $d = (d_{l1,\,l2}, d_{l2,\,l3}, \ldots, d_{lcc-1,\,lcc}, d_{lcc,\,1})$, and the vector D is calculated with Equation (10):

$$D = d \otimes q \tag{10}$$

where $\otimes$ indicates that two vectors are multiplied by elements at the same position, q is a random vector with a cc-dimensional and each component is uniformly distributed in the range [0, 2] when the offspring are generated by Equations (1) and (3), and each component is uniformly distributed in the range [0, 1.8] with a cc-dimensional when the offspring are generated by Equation (5).

Step 3: The smallest $\lfloor r * cc \rfloor$ elements are selected in D ($\lfloor\ \rfloor$ represents round down), and their positions are taken as the index. Keep these components corresponding to the index in X' and assign them to U' according to the same position.

For a clearer explanation, set an example with U = (6, 3, 5, 0, 2, 4). If the random number r = 1.5, U' = (6, 3, 5, 0, 2, 4); if r = 0.43, V' = (6, 3, 5, 2, 4) and cc = 5. Let U' = (0, 0, 0, 0, 0, 0), d = (15, 20, 31, 11, 35), and the random vector q = (1.91, 0.97, 1.60, 0.28, 0.84). Then, D = (28.65, 19.40, 49.60, 3.08, 29.40), $\lfloor r * cc \rfloor$ = 2, the index of the smallest two elements in D are 2 and 4, and the corresponding values of index 2 and 4 in V' are 3 and 2. Then, 3 and 2 are assigned to U' according to index 2 and 4, and U'= (0, 3, 0, 2, 0, 0).

From the above description, it can be seen that the redefined multiplication operation enables the offspring to inherit the good characteristics of the parent. In Equation (10), a random vector and the distance between cities are introduced in D. The former can reduce the probability of falling into local optimum, and the latter can enhance the exploitation ability.

4.1.3. Redefining the Addition Operation

Crossover is an operation that generates new potential offspring by reconstructing parents, which helps to enhance the global search ability. The addition operation in the CPA is replaced by a crossover operation so that the offspring can not only satisfy the characteristics of the TSP but also retain the attraction of carnivorous plants to prey to improve the exploitation ability. If there is no zero vector in the parents who participate in the crossover, three crossover operators are applied to realize the addition operation; otherwise, the addition operation is realized by a symmetry transformation.

For the convenience of describing the addition operation, the P_{iv} and U' (P_{iv} is randomly selected in the population, and U' is obtained by multiplication) with eight cities are set as an example, and the distances between eight cities are shown in Table 2.

Table 2. The distances between eight cities.

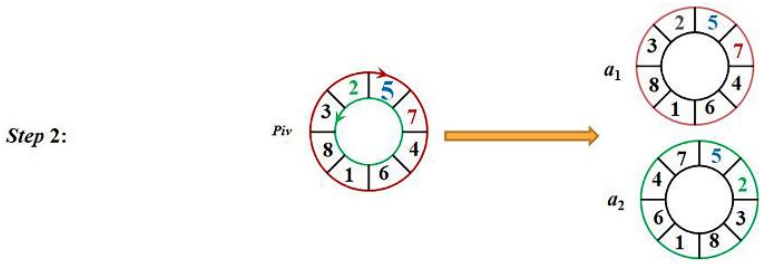
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	∞	8	11	2	10	7	4	1
2	9	∞	7	3	6	2	1	3
3	12	7	∞	5	1	6	7	9
4	1	4	3	∞	3	8	6	5
5	7	5	1	2	∞	10	6	1
6	5	3	6	8	11	∞	4	2
7	3	1	4	5	8	5	∞	1
8	2	2	9	6	1	2	1	∞

1. Partial heuristic crossover operator

When $||U'|| \neq 0$ and U' involves zero elements, the offspring generated by the traditional crossover operator may not be a legal TSP route. To fix this problem, a partial heuristic crossover operator is proposed. Suppose P_{iv} = (5 7 4 6 1 8 3 2) and U' = (4 5 0 0 0 0 2 8), the main steps to generate offspring O are as follows:

Step 1: $O = U'$;

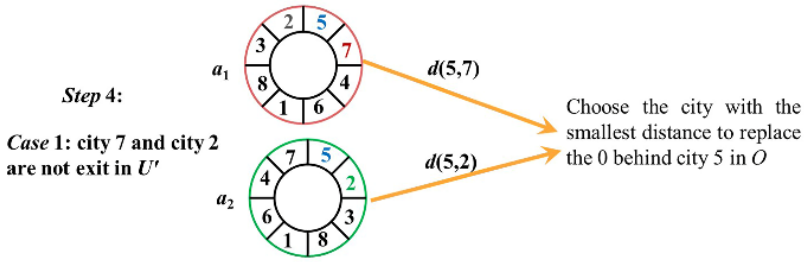
Step 2: The nonzero cities u_1 and u_2, which are nearest to 0, are found in O. Suppose u_1 is randomly selected and the position of u_1 in P_{iv} is taken as the starting point, traverse clockwise and counterclockwise, respectively (the red color indicates clockwise, and the green color indicates counterclockwise), and two new individuals, a_1 and a_2, with first component, u_1, are generated;



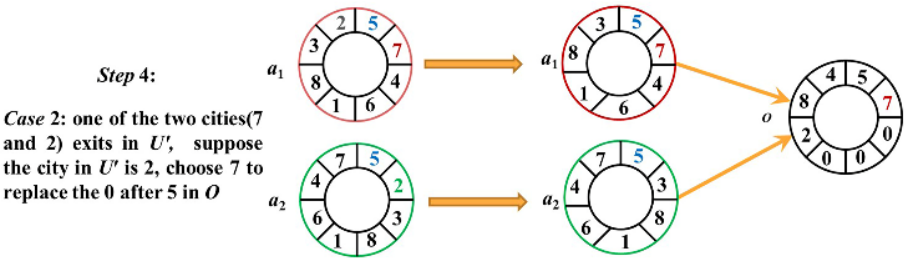
Step 3: The cities v_1 and v_2, which are nearest to u_1, are found in a_1 and a_2;

Step 4: Determine whether cities v_1 and v_2 are included in U', and three situations may exist at this time:

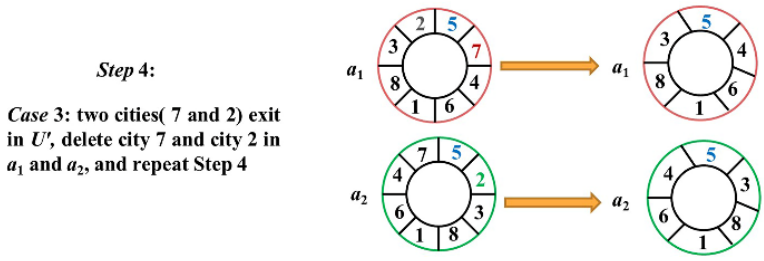
Case 1: If v_1 and v_2 are not included in U', the distances between u_1 and v_1, and u_1 and v_2 are compared, and the city with the shortest distance is selected to replace element 0 after u_1 in O. Then, turn to Step 5.



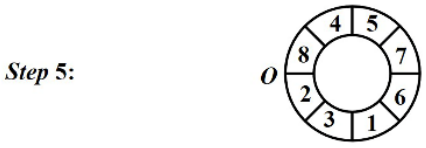
Case 2: If only one of v_1 and v_2 is included in U', the included city is deleted in a_1 and a_2 and the nonexistent city is selected to replace the element 0 after u_1 in O. Then, turn to Step 5.



Case 3: If both v_1 and v_2 are included in U', v_1 and v_2 are deleted in a_1 and a_2, and cities e_1 and e_2 adjacent to u_1 are found in a_1 and a_2. Let $v_1 = e_1$ and $v_2 = e_2$. Then, Step 4 is repeated.



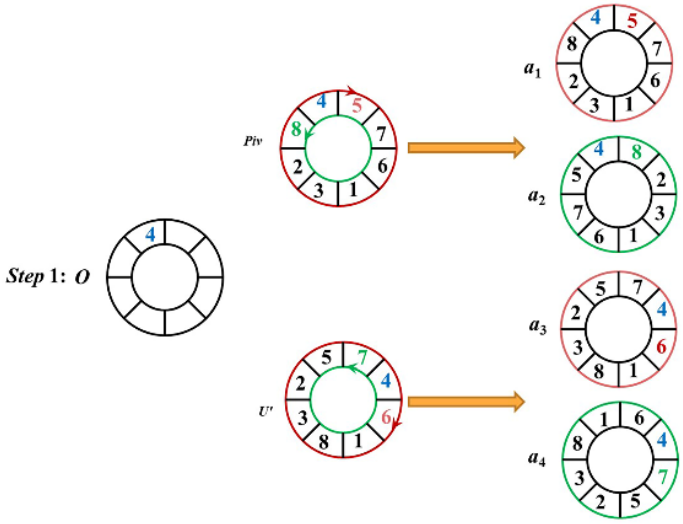
Step 5: If element 0 still exists in O, turn to Step 2; otherwise, output O.



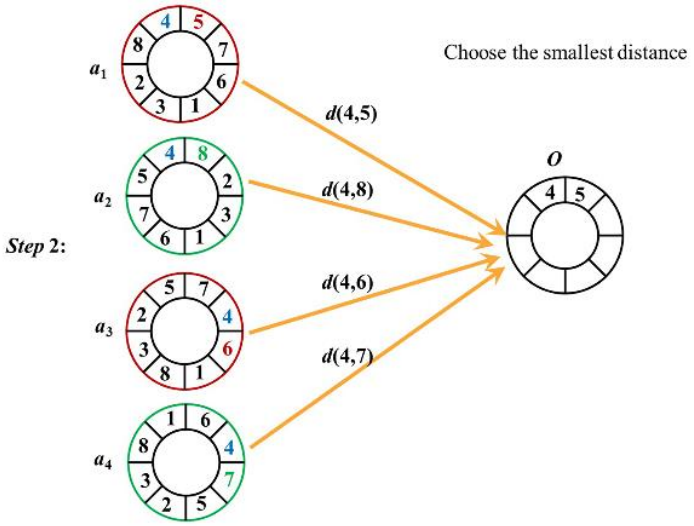
2. Bidirectional heuristic crossover operator

For the TSP, if the distance between the t-th ($t = 1, 2, \ldots, m-1$) visited city and the $(t+1)$-th visited city is the smallest, the chances of a short route will increase. The bidirectional heuristic crossover operator [48] can select the city with the shortest distance from the cities next to the t-th city in the parents as the $(t+1)$-th visited city, which can enhance the search precision. $P_{iv} = (5, 7, 6, 1, 3, 2, 8, 4)$ and $U' = (7, 4, 6, 1, 8, 3, 2, 5)$ are randomly generated and set as an example. The main steps of this operator are:

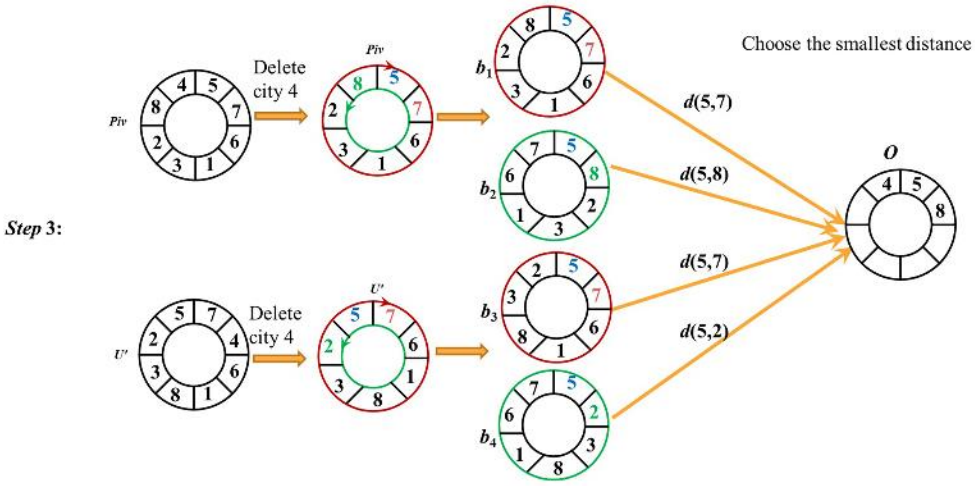
Step 1: Assuming that an integer 4 between [1, 8] is randomly generated, traverse clockwise and counterclockwise in P_{iv} and U' with the starting city 4 (the red color represents clockwise, and the green color represents counterclockwise), respectively. Then, four new individuals a_1, a_2, a_3, and a_4 are generated;



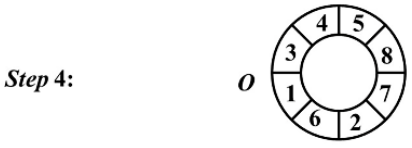
Step 2: The cities 5, 8, 6, and 7 adjacent to city 4 are found in a_1, a_2, a_3, and a_4, and calculate the distances between city 4 and cities 5, 8, 6, and 7. City 5, with the shortest distance, is selected as the second visited city of O;



Step 3: City 4 is deleted in P_{iv} and U'. Step 2 is repeated with city 5 as the starting city to obtain b_1, b_2, b_3, and b_4. The cities 8, 2, and 6 adjacent to city 5 are found in b_1, b_2, b_3, and b_4, and calculate the distances between city 5 and cities 8, 2, and 6. City 8, with the shortest distance, is regarded as the third visited city of O;



Step 4: The determined city in O is deleted in P_{iv} and U', and Step 3 is repeated until the city size of O is 8;



3. Completely mapped crossover operator

Zahid [49] proposed a completely mapped crossover operator in 2020. Suppose P_{iv} = (1 2 3 4 5 6 7 8) and U' = (6 1 5 3 8 7 2 4) in this part The two individuals are set as an example, and 1–8 represent different city numbers. In P_{iv}, the index of cities 1–8 is regarded as 1–8. In U', the index of city 6 is 1, the index of city 1 is 2, and so on. The main steps of the completely mapped crossover operator are:

Step 1: if $P_{iv} = U'$, $O = P_{iv}$. Otherwise, Step 2 is executed.

Step 2: Compare two cities corresponding to the same index in P_{iv} and U' from the first component to the eighth component until the two cities are different so that W_1 is equal to the city corresponding to the index in U'.

$$W_1 = (6)$$

Step 3: The index k_1 with city 6 is found in P_{iv}, and the city corresponding to k_1 in U' is 7. The k_2 with city 7 is found in P_{iv}, and the city corresponding to k_2 in U' is 2. Then, W_2 is:

$$W_2 = (2)$$

Step 4: The index k_3 with the city g_2 is found in P_{iv}, and the city corresponding to k_3 in U' is 1. The index k_4 with city 6 is found in P_{iv}, and the city corresponding to k_4 in U' is 7. Then, W_1 and W_2 are:

$$W_1 = (6\ 1)$$

$$W_2 = (2\ 7)$$

Step 5: If the first city in P_{iv} appears in W_2, turn to Step 6; otherwise, Step 4 is repeated.

$$W_1 = (6\ 1\ 2\ 7)$$

$$W_2 = (2\ 7\ 6\ 1)$$

Step 6: Let $O_1 = P_{iv}$ and $O_2 = U'$. The cities where O_1 is equal to W_1 and O_2 is equal to W_2 are replaced by *. * indicates that the city is unknown. O_1 and O_2 are:

$$O_1 = (*\ *\ 3\ 4\ 5\ *\ *\ 8)$$

$$O_2 = (6\ 1\ 5\ 3\ 8\ 2\ 7\ 4)$$

Step 7: * in O_1 and O_2 are replaced by cities in W_1 and W_2 in sequence, respectively.

$$O_1 = (2\ 7\ 3\ 4\ 5\ 6\ 1\ 8)$$

$$O_2 = (6\ 1\ 5\ 3\ 8\ 2\ 7\ 4)$$

Step 8: The individual with better fitness values in O_1 and O_2 are selected as newly generated offspring O.

Among the three crossover operators, the partial heuristic crossover operator applies to the case where element 0 is included in the parent; otherwise, the bidirectional heuristic crossover and the completely mapped crossover operator are selected. The former considers the distances between cities, which exhibits fast convergence speed, but easily falls into local optimums. The latter exhibits strong exploration ability and slow convergence speed. To better balance the search ability, the completely mapped crossover operator in the early iteration and the bidirectional heuristic crossover operator in the late iteration are employed with high probability. Thus, an adaptive parameter ρ is designed with Equation (11):

$$\rho = 0.7 - 0.4 * \left(\frac{T}{T_{\max}}\right)^{0.8} \tag{11}$$

where T is the current running time and $T_{\max}$ is the maximum running time.

When $||U'|| \neq 0$, the selection methods of the three crossover operators are: if element 0 is not included in U' in the growth phase and if the random number $k \leq \rho$ ($k \in [0,1]$), the completely mapped crossover operator is selected; otherwise, the bidirectional heuristic crossover operator is selected. If element 0 is included in U', the partial heuristic crossover operator is selected. In the reproduction phase, if element 0 is not included in U', the bidirectional heuristic crossover operator is selected; otherwise, the partial heuristic crossover operator is selected.

4. Symmetry transformation

A zero vector may be generated after the subtraction and multiplication operations, that is, $||U'|| = 0$. Three crossover operators do not apply to the case where a zero vector is included in parent individuals. To realize the addition operation, a symmetry transformation is employed to P_{iv}. The specific steps of the symmetry transformation are as follows:

Step 1: Four unequal random integers, N_1, N_2, N_3, and N_4, are randomly generated between [1, m] and sorted from small to large, where m is the city size;

Step 2: The cities between N_1 and N_2, and N_3 and N_4 are flipped in P_{iv};

Step 3: The city sequences between N_1 and N_2, and N_3 and N_4 are swapped in P_{iv} to generate the offspring O.

Suppose P_{iv} = 5 7 4 6 1 8 3 2, N_1 = 2, N_2 = 3, N_3 = 5, and N_4 = 6. The individual after symmetry transformation is O = 5 8 1 6 4 7 3 2.

The completely mapped crossover operator exhibits inheritance but does not exhibit a heuristic. Compared with the other two crossover operators, it has a strong global search ability. Partial heuristic crossover operators have some heuristics and inheritance. Compared with the other two crossover operators, the global search ability and local search ability are relatively balanced. The bidirectional heuristic crossover operator exhibits heuristic and strong local search abilities compared with the other two crossover operators. In addition, symmetry transformation can maintain population diversity. Therefore, the addition operation takes into account global and local search capabilities, which is beneficial for improving the performance of the algorithm.

4.2. Simple Sorting Grouping

The CPA divides the population into n_1 subgroups, each group is composed of one carnivorous plant and $n_2/\ n_1$ prey, and the details are shown in Table 1. It can be found that: (1) the implementation is complex; (2) the quality of subgroups decreases with the ranking of carnivorous plants, which leads to the imbalance in search ability among the different subgroups; and (3) the quality of individuals and search abilities in the first subgroup is significantly better than that in the other subgroups. After the offspring individuals are grouped, the individuals with the information of the first subgroup may become the local optimum of the other subgroups, which increases the assimilation of individuals and the probability of falling into the local optimum.

Pointing at the above problems, a simple sorting grouping method is proposed, which sorts the individuals according to the fitness value and divides them into two groups for updating with randomly selected individuals. The specific operations are as follows:

Step 1: The individuals are sorted from small to large for the fitness value with the population size n;

Step 2: The population is divided into two groups. The first group is carnivorous plants with the first n_1 individuals, and the second group is prey with the remaining n_2 individuals (n_1 can be arbitrarily determined as long as $n_2 > n_1$, $n_2 + n_1 = n$ is satisfied and n_2 can be divisible by n_1).

Step 3: In the update method of the growth phase and reproduction phase, the carnivorous plants are randomly selected from the first group, and the prey are randomly selected from the second group.

Simple sorting grouping divides the population into two groups according to the fitness value of the individual, which is easier and faster to group. Carnivorous plants and prey are randomly selected from the two groups to update in the simple sorting grouping, which can reduce the probability of becoming stuck in a local optimum and improve the exploration ability.

4.3. Adaptive Attraction Probability

The growth of carnivorous plants or the update of prey in the CPA depends on the attraction probability, γ. The update of prey can enhance the exploration ability, and

the growth of carnivorous plants can enhance the exploitation ability. However, γ in the CPA is a constant 0.8, which does not improve the balance between the exploration and exploitation ability. Therefore, an adaptive attraction probability is designed with Equation (12):

$$\gamma = 0.45 + 0.45 * \left(\frac{T_1}{T_{\max}} \right)^{0.4} \tag{12}$$

where T_1 is the current running time and $T_{\max}$ is the maximum running time.

When γ is greater than the random number, λ ($\lambda \in [0, 1]$), the carnivorous plant grows; otherwise, the prey updates its position. It can be seen from Equation (12) that γ is small in the early stage of iteration, and the prey has a high probability of updating, which exhibits a strong exploration ability. With the increase in time, the probability of the carnivorous plant growing increases, which exhibits a strong exploitation ability.

4.4. Iteration Local Search Algorithm

The idea of ILS [50] is to perturb the local optimal solution obtained by the local search algorithm. Then, the local search algorithm is conducted on the perturbed individual. Three parts are mainly included in ILS: a local search algorithm, a perturbation method, and an acceptance criterion. In ILS, the local search algorithm is first applied to the population. Then, the optimal individual is selected, and the double-bridge perturbation is executed to obtain an intermediate solution. Finally, the intermediate solution is searched locally, and the newly generated solution is used to replace the optimal solution if its fitness value is better than the optimal solution. The details of ILS are shown below.

1. Local search algorithm

The local search process is to search the neighborhood of the current solution to find a better solution and update the solution until it cannot be improved. The 2-Opt algorithm [51] is an efficient method for the TSP, which can quickly eliminate the intersection edges in each path and enhance search precision. Its main idea is that for each route the two nonadjacent edges are exchanged in turn to obtain the set of paths, and the reservation can improve the exchange of paths.

2. Perturbation

Four nonadjacent edges are selected in the double-bridge [52] perturbation to be disconnected and reconnected, aiming to reduce the probability of becoming stuck in a local optimum. X_1= (x_1^1, x_1^2, . . . , x_1^m) is taken as an example to illustrate the process of the double-bridge, where m is the city size and x_1^m indicates m-th city in X_1.

Step 1: Integers J_1, J_2, J_3, and J_4, are randomly generated between [2, $m - 6$], [2 + J_1, $m - 4$], [2 + J_2, $m - 2$], and [2 + J_3, m], respectively;

Step 2: Let $G_1 = J_1 - 1$, $G_2 = J_2 - 1$, $G_3 = J_3 - 1$, and $G_4 = J_4 - 1$;

Step 3: The cities corresponding to indexes J_1, J_2, J_3, J_4, G_1, G_2, G_3, and G_4 in X_1 are regarded as aa_1, aa_2, aa_3, aa_4, bb_1, bb_2, bb_3, and bb_4, respectively.

Step 4: The edges (aa_1, bb_1), (aa_2, bb_2), (aa_3, bb_3), and (aa_4, bb_4) are disconnected, and the edges (aa_1, bb_3), (aa_2, bb_4), (aa_3, bb_1), and (aa_4, bb_2) are reconnected.

An example with X_1 = (3, 2, 1, 6, 5, 4, 7, 10, 9, 11, 8) is provided in Figure 1a, and the initial path after the double-bridge perturbation is provided in Figure 1b.

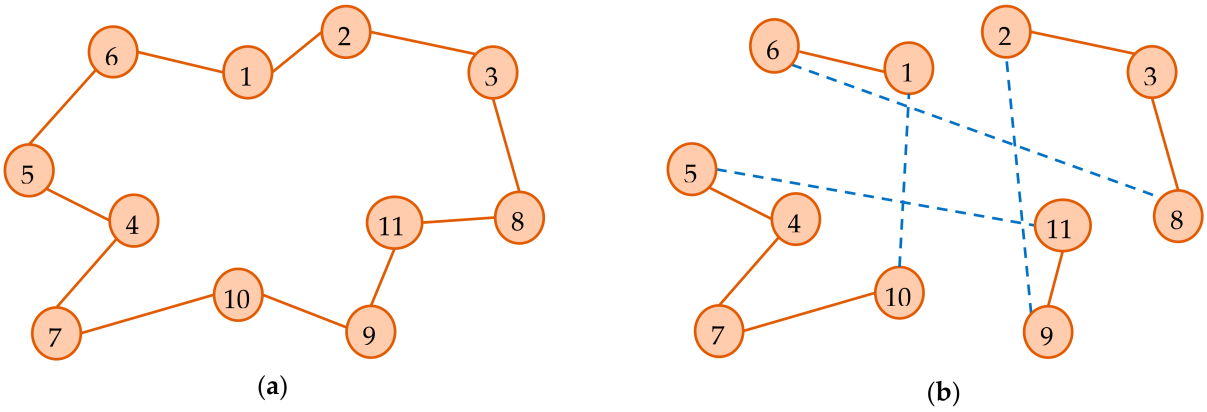


Figure 1. The double-bridge perturbation. (**a**) Initial path; (**b**) The path after double-bridge perturbation. The number in the circle represents the city, and the blue dotted line represents the path after reconnection.

The pseudo-code of ILS is summarized in Algorithm 2. In Algorithm 2, *XX* indicates the individual in the population.

Algorithm 2. ILS

```
Execute the 2-Opt algorithm on XX and calculate the fitness of the population after the 2-Opt algorithm;
Find the best individual and record as X_i;
Execute the double-bridge perturbation and 2-Opt algorithm on X_i and regard the individual after the two operations as X*;
If f(X*) < f(X)
    Replace X with X*;
End
```

4.5. Similarity-Eliminating Operation

For a multi-extremum optimization problem, the individuals in the population continue to move to extreme points and finally gather near these points as the iteration time grows. Therefore, the number of similar and identical individuals increases as the iteration time grows, which leads to poor population diversity and the possibility of excellent individuals being generated being reduced.

To avoid the above phenomenon, the similarity-eliminating operation is added to DCPA with two suboperations: suboperation 1 is based on the same quantity of city sequences, and suboperation 2 is based on the route length.

1. The similarity-eliminating operation based on the same quantity of city sequences:

The solution of the TSP is a sequence of m nonrepeating integers, and m is the city size. The same city sequence in different individuals will increase as the iterations increase. If the quantity of the same city sequence in different individuals is greater than $n^{0.8}$, these individuals are regarded as similar. X_1 = (1 2 3 4 5 6 7 8 9) and X_2 = (7 9 1 2 3 4 8 5 6) are randomly generated and taken as an example to illustrate the similarity-eliminating operation based on the same quantity of city sequences. Each component in X_1 and X_2 represent a city number. X_1 and X_2 are depicted in Figure 2.

Step 1: From Figure 2, it can be found that the same quantity of the city sequences of X_1 and X_2 is 6, which are (1, 2), (2, 3), (3, 4), (5,6), (6, 7), and (9,1).

Step 2: X_1 and X_2 are regarded as similar individuals for the same quantity of city sequence $6 > 9^{0.8}$, where 9 is the city size.

Step 3: The individual X_1 with the shortest route length is preserved, and X_2 is removed. Then, X_2 is randomly generated to maintain the same population size.

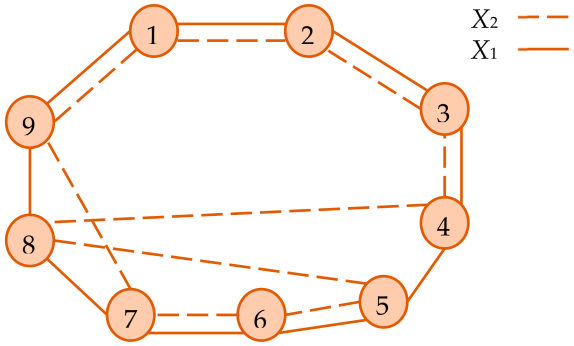


Figure 2. The routes of X_1 and X_2.

2. The similarity-eliminating operation based on the route length:

Individuals with the same route length in the population are considered identical. The steps of the similarity-eliminating operation based on the route length are as follows:

Step 1: Calculate and find individuals with the same route length in the population;

Step 2: One of the same individuals is kept and the rest are removed;

Step 3: The removed individuals are replaced with randomly generated individuals to maintain the same population size.

An example of population size n = 10 and city size m = 6 is given in Table 3, where I is the initial population, f is the route length of individuals, I_1 is the population after removing the identical individuals, Y is the randomly generated individuals, and I' is the population after the similarity-eliminating operation based on the route length. The bold in f means individuals with the same route length.

Table 3. The similarity-eliminating operation based on the route length.

I						f	I_1						f	Y						f	I'						f
6	3	5	1	2	4	33	6	3	5	1	2	4	33	6	3	5	4	1	2	20	6	3	5	1	2	4	33
5	1	2	3	4	6	**46**	5	1	2	3	4	6	46	1	6	4	2	3	5	34	5	1	2	3	4	6	46
4	5	2	3	6	1	**28**	4	5	2	3	6	1	28	2	1	3	5	6	4	43	4	5	2	3	6	1	28
4	3	1	5	6	2	41	4	3	1	5	6	2	41								4	3	1	5	6	2	41
6	4	5	1	3	2	38	6	4	5	1	3	2	38								6	4	5	1	3	2	38
4	5	2	3	6	1	**28**	2	4	5	3	1	6	29								2	4	5	3	1	6	29
2	4	5	3	1	6	29	4	2	6	5	1	3	40								4	2	6	5	1	3	40
1	2	3	4	6	5	**46**															6	3	5	4	1	2	20
4	2	6	5	1	3	40															1	6	4	2	3	5	34
3	6	1	4	5	2	**28**															2	1	3	5	6	4	43

According to the description, identical individuals must be similar individuals, but the converse is not true. For example, 1 2 3 4 and 4 1 2 3 are identical and similar individuals; 1 2 3 4 5 6 7 8 9 and 7 9 1 2 3 4 8 5 6 are similar individuals, but not identical individuals. Suboperation 1 performs the similarity-eliminating operation on the individuals who execute the update method. Therefore, identical individuals may exist in the recombined phase. To maintain population diversity and reduce the possibility of trapping into a local optimum, suboperation 2 is added after the recombination phase.

4.6. The Framework of DCPA

The framework of the DCPA is shown in Figure 3. It can be observed that the performance of the DCPA is mainly concerned with the update method of offspring, the grouping method, the local search strategy, and the similarity-eliminating operation. A new generation method that redefines subtraction, multiplication, and addition operations is designed in the DCPA. The method not only satisfies the legitimacy of the TSP but also enhances the search precision. In addition, a simple sorting grouping method is proposed, which

can promote the grouping speed and exploration ability compared with the method in the CPA. At the same time, an adaptive attraction probability is proposed to better balance the search ability, which can fix the defect that γ is a constant. Then, ILS is introduced to heighten the exploitation ability and reduce the probability of solution stagnation. Finally, the similarity-eliminating operation composed of two suboperations is added, which can not only identify the identical individuals but also the individuals with high similarity in the update operation so that the DCPA can maintain the population diversity during the iteration. Therefore, the DCPA can better balance the exploration and exploitation abilities.

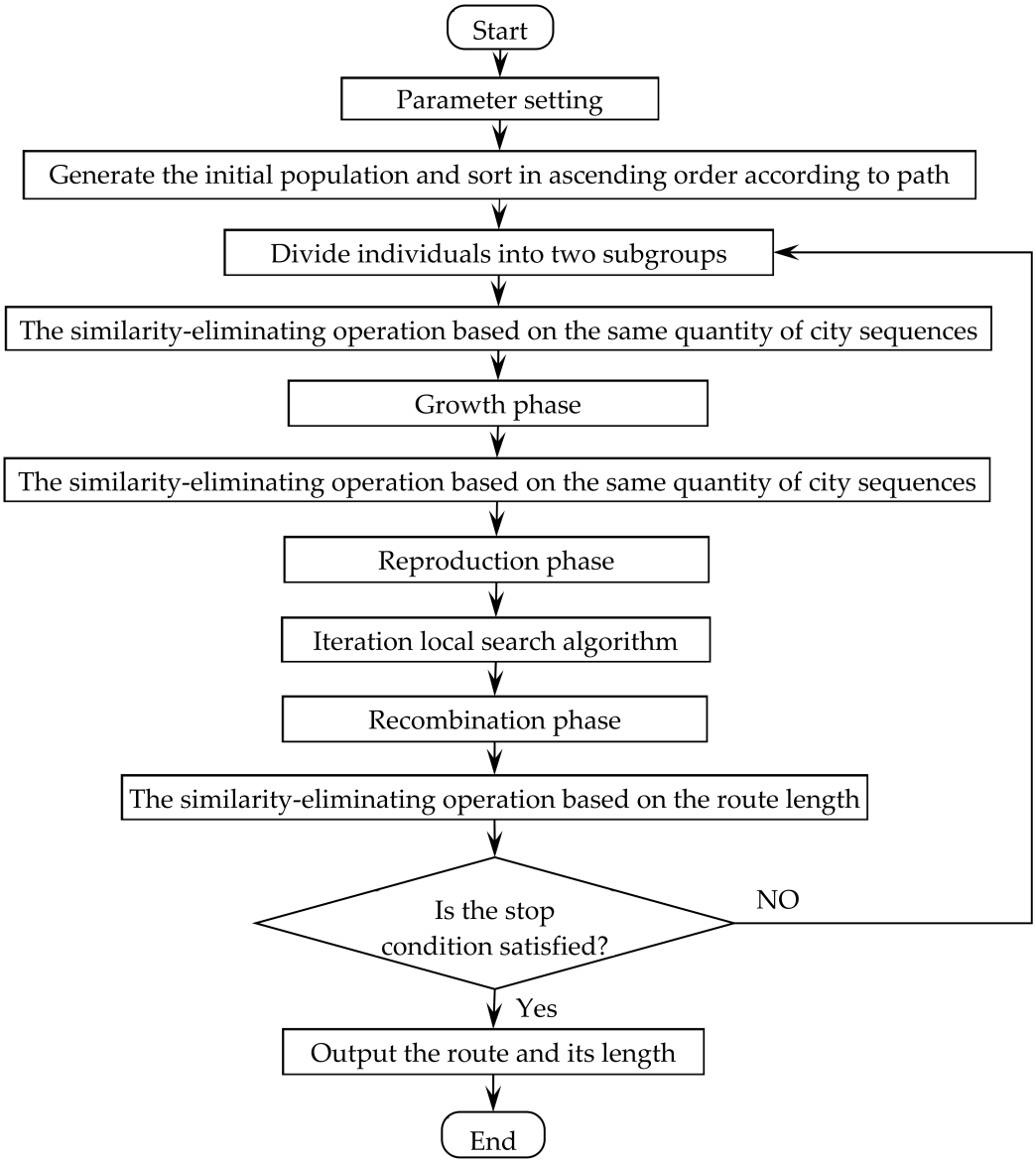


Figure 3. The framework of the DCPA.

The pseudo-code of the DCPA is summarized in Algorithm 3, where n is the population size, T is the iteration time of the proposed algorithm, and $T_{\max}$ is the maximum iteration time.

Algorithm 3. The DCPA

Let $T = 0$;
Randomly generate n initial individuals, record the population as A, and calculate its fitness;
Divide individuals into subgroups according to Section 4.2;
While $T < T_{\max}$,
  Perform the eliminate operation based on the same city sequence on the involved individuals and the offspring generated in the growth
phase by Equation (1) and Equation (4) with the redesigned subtraction, multiplication, and addition operators;
  Perform the eliminate operation based on the same city sequence on the involved individuals and the offspring generated by Equation (5)
with the redesigned subtraction, multiplication, and addition operators and record the population as B;
  Perform Algorithm 2 on B and record the population as C;
  A, B, and C combine to form a new population, recorded as I;
  Perform the eliminate operation based on route length on I, which is discussed in Section 4.5;
  Select n best chromosomes according to the objective function from I;
  Record the running time and update runtime;
End While
Output the best solution and the optimal value;

5. Experimental Results and Discussions

The algorithms involved in this work were developed in Matlab R2019b. To ensure the fairness of the experiment, all experiments were performed on the same computer. The computer used Windows 10 and operated at 3.09 GHz with 32GB of RAM. Several comparisons were conducted for evaluating the performance of the proposed algorithm in various instances from TSPLIB as follows: (1) self-comparison, where the algorithm was compared with other versions of the DCPA to evaluate the significance of improvements, (2) DCPA without a local search algorithm was compared with the other algorithms without local search, and finally, (3) a DCPA comparison with other algorithms with local search.

5.1. Experiment Termination Conditions

Due to the different complexities of the algorithms participating in the comparison, the traditional method of setting the maximum number of iterations as the termination condition could not fairly measure the performance. Thus, the maximum iteration time with different city sizes was set as the termination condition in this work, which is shown in Table 4. In addition, if the length of the current optimal route was less than or equal to the best known optimum before the maximum iteration time, the iteration was ended. Each experiment was performed with 20 independent runs, and the best solution of each dataset for each run was recorded.

Table 4. The maximum iteration times with different city sizes.

City Number	Run Time (s)
$m < 50$	10
$50 \leq m < 100$	20
$100 \leq m < 200$	50
$200 \leq m < 300$	100
$300 \leq m < 600$	160
$600 \leq m < 1000$	250
$m \geq 1000$	600

5.2. Comparisons and Analysis

To evaluate the performance of the proposed algorithm, the minimum (Best), maximum (Worst), average (Avg), standard deviation (SD), and deviation percentage of the Avg (PDA) values were adopted as a measurement. The Friedman test [53,54] and Holm's procedure [55] were employed to verify the significant differences among the participating algorithms.

5.2.1. Validation of the Improved Component

Several versions of the proposed algorithm, starting from the CPA with swap sequence and with ILS to the final version of the DCPA, which incorporates a new individual-generated method, similarity-eliminating operation, simple sorting group method, adaptive attraction rate, and local search algorithm, are described in Table 5. For the seven algorithms in Table 5, the population size was n = 100, the quantity of the carnivorous plant was 25, and the quantity of the prey was 75. For each algorithm, each instance was run 20 times independently, and the maximum running time is summarized in Table 4.

Table 5. Description of different versions of the CPA.

Algorithm	Reference	Description
Version 1	[26]	CPA with swap sequence and swap operator
Version 2	[30]	CPA with the order-based decoding
Version 3	-	CPA with a new type of individual generation method
Version 4	-	Version 3 + simple sorting group
Version 5	-	Version 4 + similarity-eliminating operation
Version 6	-	Version 5 + adaptive attraction rate
DCPA	-	Version 6 + iterated local search

1. Comparison with the other individual-generated methods

Three algorithms, Version 1–Version 3, were adopted to illustrate the effectiveness of our proposed individual generation method, and the results are shown in Table 6. The description of the participating algorithms is shown in Table 5.

It can be observed from Table 6 that the PDA of Version 3 was less than 3.35% on 11 instances and more than 3.35% on 5 instances, such as rat99, eil101, bier127, ch130, and d198. However, the PDA of the other two methods was more than 3.35% on 16 instances, and compared with the other two methods, our method could obtain better solutions on 15 instances, except on bay29. In addition, the mean values of Version 3 were superior to the other two methods on 15 instances, except on d198. The above analyses verified that our individual-generated method is suitable for the TSP, and it can direct the produced discrete solutions towards optimality compared with the other two methods.

2. Self-comparisons

Several self-comparisons between different versions of the DCPA, to illustrate the effect of each improvement component (simple sorting group, similarity-eliminating operation, and adaptive attraction probability) on the overall performance of the proposed DCPA, were conducted, and the results are summarized in Table 7. The description of the participating algorithms is shown in Table 5.

Compared with Version 3 in Table 7, it can be noticed that the PDA obtained by Version 4 only lost in pr107 and improved in 21 instances. The PDA obtained by Version 5 won in 22 instances compared with Version 4. For Version 6, it lost in 6 instances compared to Version 5 with regard to PDA and performed better in 16 instances. The analyses above indicates that adding the new components can increase the performance of the proposed algorithm.

The mean rank and final rank of the Friedman rank test are depicted in Figure 4, where the mean rank gradually decreased and the final rank gradually increased among Version 3, Version 4, Version 5, and Version 6, which verified that each added component enhanced the algorithm's performance.

Table 6. Comparison of the proposed individual-generated method with the other two individual-generated methods.

Name	BKS	Evaluation Indicators	Best	Mean	SD	PDA	Name	BKS	Evaluation Indicators	Best	Mean	SD	PDA
bayg29	9073	Version 1	**9073**	9219.25	177.28	3.36	eil101	629	Version 1	736	748.35	8.05	21.30
		Version 2	9077	10,127.5	1091.11	11.62			Version 2	1261	1701.35	274.05	170.48
		Version 3	**9073**	**9117**	**42.78**	**0.48**			Version 3	**638**	**653**	**6.92**	**3.79**
att48	33,522	Version 1	35,391	36,165.95	**439.73**	10.12	lin105	14,379	Version 1	15,187	15,944.65	356.42	14.97
		Version 2	39,577	49,339.7	7817.5	47.19			Version 2	41,812	53,678.9	7589.82	273.31
		Version 3	33,522	34,206	530.07	2.04			Version 3	**144,58**	**146,41**	**102.91**	**1.82**
eil51	426	Version 1	478	481.95	**1**	13.38	pr107	44,303	Version 1	45,174	45,852.6	**337.53**	4.47
		Version 2	525	705.05	112.07	65.50			Version 2	10,8219	171,205.5	42,135.77	286.44
		Version 3	**429**	**436**	4.86	**2.41**			Version 3	**44,715**	**45,743**	1370.78	**3.25**
berlin52	7542	Version 1	7924	8064.25	**112.47**	8.47	pr124	59,030	Version 1	61,879	64,735	1537.78	12.91
		Version 2	9746	11,709.05	1395.88	55.25			Version 2	201,876	329,922.4	49,473.11	458.91
		Version 3	**7542**	**7716**	154.19	**2.30**			Version 3	**59,246**	**60,513.85**	**775.80**	**2.51**
st70	675	Version 1	771	794.85	**7.13**	18.96	bier127	118,282	Version 1	127,565	133,797.4	**2118.56**	15.34
		Version 2	1017	1554.55	281.25	130.3			Version 2	315,939	368,448.5	25,343.56	211.5
		Version 3	**677**	**693**	13.51	**2.62**			Version 3	**120,557**	**123,923.4**	2214.12	**4.77**
eil76	538	Version 1	599	608.5	**2.91**	14.13	ch130	6110	Version 1	7025	7111.55	**38.27**	17.71
		Version 2	914	1189.5	124.17	121.1			Version 2	20,274	24,971.05	2122.1	308.69
		Version 3	**542**	**554**	10.43	**3.00**			Version 3	**6217**	**6386.15**	109.19	**4.52**
rat99	1211	Version 1	1423	1441.45	**9.12**	19.98	ch150	6528	Version 1	7053	7116.95	**15.05**	9.38
		Version 2	2935	3660.3	471.74	202.3			Version 2	19,449	30,320.05	3536.21	364.46
		Version 3	**1224**	**1264**	26.26	**4.37**			Version 3	**6589**	**6714.6**	96.17	**2.86**
kroA 100	21,282	Version 1	23,224	24,032.4	417.15	16.30	d198	15,780	Version 1	17,620	**17,780.25**	**128.94**	13.92
		Version 2	51,822	67,556.65	12,309.2	217.4			Version 2	67,595	93,606.95	13,727.66	493.2
		Version 3	**21,282**	**21,650**	**285.89**	**1.73**			Version 3	**17,355**	18,823	564.70	19.29

Note: the best is set in bold.

Table 7. Experimental results of the DCPA and its different versions.

Name	BKS	Evaluation Indicators	Best	Worst	Mean	SD	PDA (%)	Name	BKS	Evaluation Indicators	Best	Worst	Mean	SD	PDA (%)
bayg29	9073	Version 3	**9073**	9237	9116.7	42.78	0.48	lin105	14,379	Version 3	14,458	14,826	14,641.3	102.91	1.82
		Version 4	**9073**	9213	9107.30	42.91	0.38			Version 4	14,490	15,291	14,640.8	184.95	1.82
		Version 5	**9073**	9094	9074.85	4.7	0.02			Version 5	**14,379**	14,715	14,570.1	**84.74**	1.33
		Version 6	**9073**	**9077**	**9073.40**	**1.20**	**0.00**			Version 6	**14,379**	**14,625**	**14,500.75**	85.96	**0.85**
att48	33,522	Version 3	**33,522**	34,929	34,205.65	530.07	2.04	pr107	44,303	Version 3	44,715	49,126	45,743	1370.78	3.25
		Version 4	**33,522**	35,107	34,093.50	394.83	1.70			Version 4	44,358	48,064	46,058.5	1131.31	3.96
		Version 5	**33,522**	**34,465**	33,944.05	329.51	1.26			Version 5	**44,354**	45,116	**44,632.4**	222.29	**0.74**
		Version 6	**33,522**	34,828	**33,837.40**	**318**	**0.94**			Version 6	44,486	**45,072**	44,773.65	**167.88**	1.06
eil51	426	Version 3	429	448	436.25	4.86	2.41	pr124	59,030	Version 3	59,246	61,671	60,513.85	775.8	2.51
		Version 4	428	444	436.00	3.60	2.35			Version 4	59,210	61,011	59,847.8	623.21	1.39
		Version 5	**426**	**436**	**429.55**	**3.29**	0.83			Version 5	59,087	60,639	**59,597.75**	522.62	**0.96**
		Version 6	**426**	**436**	**429**	3.35	**0.79**			Version 6	**59,076**	**60,413**	59,786.4	**442.27**	1.28
berlin52	7542	Version 3	**7542**	8203	7715.65	154.19	2.30	bier127	118,282	Version 3	120,557	128,935	123,923.4	2214.12	4.77
		Version 4	**7542**	8159	7703.85	187.20	2.15			Version 4	120,035	129,044	123,524.1	2533.8	4.43
		Version 5	**7542**	**7944**	7689.45	164.4	1.96			Version 5	**119,315**	130,820	122,036.5	2267.63	3.17
		Version 6	**7542**	7994	**7662.45**	**152.95**	**1.60**			Version 6	119,534	**124,323**	**121,398.2**	**1131.27**	**2.63**
st70	675	Version 3	677	724	692.7	13.51	2.62	ch130	6110	Version 3	6217	6614	6386.15	109.19	4.52
		Version 4	675	725	692.50	11.58	2.59			Version 4	6143	6527	6297.25	88.84	3.06
		Version 5	676	**703**	687.3	7.85	1.82			Version 5	6174	6432	6272.05	59.48	2.65
		Version 6	**675**	708	**683.80**	**6.80**	**1.30**			Version 6	**6159**	**6351**	**6258.2**	**56.35**	**2.43**
eil76	538	Version 3	542	570	554.15	10.43	3.00	pr144	58,537	Version 3	59,136	63,287	60,308.3	1286.94	3.03
		Version 4	**540**	570	553.85	7.93	2.95			Version 4	58,658	60,882	59,550.8	749.64	1.73
		Version 5	541	**555**	**550.2**	**3.91**	**2.27**			Version 5	58,673	59,448	59,009.6	282.1	0.81
		Version 6	541	561	550.75	6	2.37			Version 6	**58,620**	**59,395**	**58,846.6**	**228.59**	**0.53**
pr76	108,159	Version 3	109,809	115,790	112,139.7	1532.74	3.68	kroB150	26,130	Version 3	26,561	27,461	27,011.05	297.03	3.37
		Version 4	108,856	116,962	111,818.5	2119.3	3.38			Version 4	26,484	27,550	26,957.35	451.15	3.17
		Version 5	109,118	113,585	111,190	**1257.32**	2.80			Version 5	26,380	27,549	26,688.65	311.21	2.14
		Version 6	**108,644**	**113,296**	**111,014.5**	1322.67	**2.64**			Version 6	**26,180**	**27,409**	**26,687.8**	**303.45**	**2.13**

Table 7. *Cont.*

Name	BKS	Evaluation Indicators	Best	Worst	Mean	SD	PDA (%)	Name	BKS	Evaluation Indicators	Best	Worst	Mean	SD	PDA (%)
rat99	1211	Version 3	1224	1304	1263.95	26.26	4.37	ch150	6528	Version 3	6589	6963	6714.6	96.17	2.86
		Version 4	1228	1296	1256.65	19.99	3.77			Version 4	6581	6944	6713.7	107.03	2.84
		Version 5	1214	**1269**	1240.9	**15.9**	2.47			Version 5	6564	**6727**	**6640.95**	**47.38**	**1.73**
		Version 6	**1213**	1284	**1240.35**	21.28	**2.42**			Version 6	**6555**	6758	6642	53.52	1.75
kroA100	21,282	Version 3	**21,282**	22,173	21,650.15	285.89	1.73	d198	15,780	Version 3	17,355	20,243	18,823.25	564.7	19.29
		Version 4	**21,282**	22,000	21,541.4	190.34	1.22			Version 4	17,702	18,916	18,292.2	361.01	15.92
		Version 5	**21,282**	21,901	**21,478.35**	183.97	**0.92**			Version 5	**16,384**	17,557	16,884.75	283.94	7.00
		Version 6	**21,282**	**21,758**	21,480.3	**130.23**	0.93			Version 6	16,463	**17,454**	**16,847.5**	**269.84**	**6.76**
kroB100	22,141	Version 3	22,404	23,346	22,820.65	259.05	3.07	kroA 200	29,368	Version 3	29,718	31,835	30,708.55	647.38	4.56
		Version 4	22,270	23,161	22,584.35	221.26	2.00			Version 4	29,791	31,380	30,439.35	435.4	3.65
		Version 5	22,249	22,930	22,482.5	210.13	1.54			Version 5	29,533	30,609	30,058.8	**263.78**	2.35
		Version 6	**22,220**	**22,670**	**22,434.75**	**198.77**	**1.33**			Version 6	**29,455**	**30,595**	**29,997.2**	299.18	**2.14**
eil101	629	Version 3	638	667	652.85	6.92	3.79	pr226	80,369	Version 3	82,506	93,964	86,059.55	2469.64	7.08
		Version 4	640	671	652.7	7.16	3.77			Version 4	81,681	87,224	84,539.7	1533.33	5.19
		Version 5	**630**	664	645.3	7.94	2.59			Version 5	**80,882**	84,034	**81,824.95**	662.4	**1.81**
		Version 6	632	**660**	**644.4**	**6.69**	**2.45**			Version 6	81,031	**83,800**	81,936.4	**645.2**	1.95

Note: the best is set in bold.

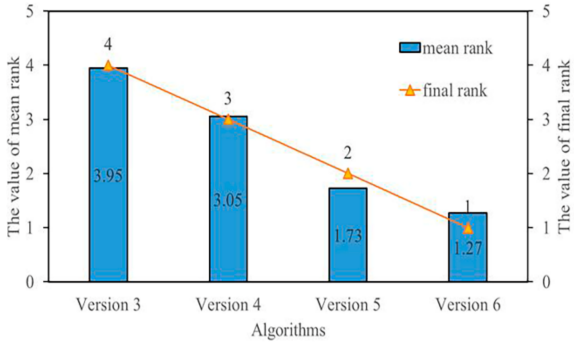


Figure 4. The ranks of the five participating algorithms.

The Friedman statistic, χ^2, with the significance level of 0.05 is summarized in Table 8, where χ^2 is 59.62. The critical value of $\chi^2_{0.05[3]}$ is 7.82 with $s - 1 = 3$ degrees of freedom. Thus, the null hypothesis is rejected at $\alpha = 0.05$.

Table 8. The results of the Friedman statistic when the algorithms number $s = 4$ and standard instances number $n = 22$.

α	χ^2	$\chi^2_{\alpha(s-1)}$	Null Hypothesis	Alternative Hypothesis
0.05	59.62	7.82	Reject	Accept

Holm's procedure was used to further verify the significant impact of a single component on the algorithm performance. The results of all pairwise comparisons are summarized in Table 9, where v is the rank value corresponding to the unadjusted p-value, sorted from small to large.

Table 9. The unadjusted and adjusted p-values for multiple comparisons among the five algorithms.

v	Comparison	Unadjusted p-Value	Adjusted p-Value	v	Comparison	Unadjusted p-Value	Adjusted p-Value
1	Version 6 vs. Version 3	5.59×10^{-12}	3.34×10^{-11}	4	Version 5 vs. Version 4	7.08×10^{-4}	2.12×10^{-3}
2	Version 5 vs. Version 3	11.05×10^{-8}	5.25×10^{-8}	5	Version 4 vs. Version 3	1.95×10^{-12}	3.90×10^{-2}
3	Version 6 vs. Version 4	5.00×10^{-6}	2.00×10^{-5}	6	Version 6 vs. Version 5	0.24	0.24

As presented in Table 9, it is noted that the statistical test does not show a significant difference between Version 6 and Version 5 for the adjusted p-value and unadjusted p-value, which are more than 0.05 at a 95% confidence level, and this is because the adaptive attraction rate focuses on enhancing the exploration rather than the exploitation ability of the algorithm. However, compared to Version 5, Version 6 enhanced the mean PDA, which verified that the adaptive attraction rate plays a positive role in the algorithm's performance. The rest of the adjusted p-values are all lower than 0.05, which confirms that the similarity-eliminating operation and the simple sorting group can significantly enhance the algorithm's performance. The analyses above confirmed the efficiency of the introduced components in improving the performance of the proposed algorithm.

5.2.2. Validation of the Proposed Algorithm

1. Compared with the algorithms without the local search algorithm:

In this paper, three algorithms without local search were compared with DCPA without iterated local search (Version 6), which include DJAYA [27], the agglomerative greedy brain storm optimization algorithm (AGBSO3) [56], and DSMO [25]. The parameters of the participating algorithms are summarized in Table 10, and the comparison results on 10 instances from TSPLIB are shown in Table 11.

Table 10. Parameter settings of the participating algorithms.

Algorithm	Year	Reference	Parameters
Version 6	2022	-	population size: 100, carnivorous plant: 25, prey: 75
DJAYA	2021	[27]	N = 20; $ST1$ = 0.5; $ST2$ = 0.5
AGBSO3	2020	[56]	cluster-num: 5, p-replace: 0.3, p-one: 0.6, p-two: 0.4, p-one-center:0.45, p-two-center: 0.5, *MM*: 100
DSMO	2019	[25]	*MG*: 5, *LLL*: 50–100, *GLL*: 50–100, *pr*: 0.1, population size: 200

Table 11. Comparison results of Version 6 and the other algorithms without a local search algorithm.

Name	BKS	Evaluation Indicators	Best	Worst	Mean	SD	PDA (%)	Name	BKS	Evaluation Indicators	Best	Worst	Mean	SD	PDA (%)
bayg29	9073	Version 6	**9073**	**9077**	**9073.4**	**1.2**	**0.00**	kroA100	21,282	Version 6	**21,282**	**21,758**	**21,480.3**	**130.23**	**0.93**
		DSMO	**9073**	**9077**	9073.6	1.43	**0.00**			DSMO	21,483	21,892	21,676.6	148	1.85
		AGBSO3	**9073**	9120	9083.1	16.16	0.11			AGBSO3	**21,282**	22,092	21,495.55	238.7	1.00
		DJAYA	**9073**	9332	9208.5	154.86	1.49			DJAYA	21,549	22,926	22,237.1	370.19	4.49
eil51	426	Version 6	**426**	436	**429.35**	3.35	**0.79**	kroB100	22,141	Version 6	**22,220**	**22,670**	**22,434.75**	198.77	**1.33**
		DSMO	427	**432**	**429.35**	**1.96**	**0.79**			DSMO	22,232	22,722	22,512.75	**166.7**	1.68
		AGBSO3	**426**	438	430.15	3.78	0.97			AGBSO3	22,284	22,905	22,525.45	280.53	1.74
		DJAYA	429	446	438.35	8.34	2.90			DJAYA	22,551	23,529	22,960.2	370.19	3.70
st70	675	Version 6	**675**	708	**683.8**	6.8	**1.30**	eil101	629	Version 6	**632**	660	**644.4**	6.69	**2.45**
		DSMO	678	**696**	685.45	**3.75**	1.55			DSMO	648	664	658.15	4.99	4.63
		AGBSO3	**675**	711	685.65	8.26	1.58			AGBSO3	637	**652**	645.15	**4.24**	2.57
		DJAYA	687	749	716.25	19.89	6.11			DJAYA	651	681	664.7	**4.24**	5.68
eil76	538	Version 6	**541**	561	550.75	5.66	2.37	lin105	14,379	Version 6	**14,379**	**14,625**	**14,500.75**	**85.96**	**0.85**
		DSMO	551	561	557.05	**3.18**	3.54			DSMO	14,471	14,812	14,639.15	103.59	1.81
		AGBSO3	542	**559**	**550.05**	4.65	**2.24**			AGBSO3	**14,379**	14,992	14,535.5	166.09	1.09
		DJAYA	554	585	564.4	7.19	4.91			DJAYA	14,683	14,653	15,081	307.67	4.88
rat99	1211	Version 6	**1213**	1284	**1240.35**	21.28	**2.42**	ch150	6528	Version 6	**6555**	**6758**	**6642**	**53.52**	**1.75**
		DSMO	1220	1298	1276.65	15.93	5.42			DSMO	6664	6918	6773.8	56.86	3.77
		AGBSO3	1220	**1276**	1240.6	**15.67**	2.44			AGBSO3	6578	6835	6667.6	73	2.14
		DJAYA	1270	1334	1299.95	19.9	7.35			DJAYA	6564	6963	6762.7	123.32	3.60

Note: the best is set in bold.

According to the results in Table 11, it can be observed that Version 6 achieved a smaller PDA (%) for 8 instances out of 10 instances. For Version 6, the PDA (%) was no more than 2.45% for all 10 instances. However, there were only six instances of DSMO, nine instances of AGBSO3, and one instance of DJAYA that were no more than 2.45%. The experimental results illustrate that Version 6 has a strong optimization capability and is stable to compute a satisfactory solution in each experiment. Thus, our proposed method without the local search algorithm is robust and effective in solving the TSP compared with the other three algorithms without the local search algorithm.

The mean rank and final rank of the Friedman rank test are depicted in Figure 5, where the mean rank and final rank of Version 6 are better than the other three algorithms.

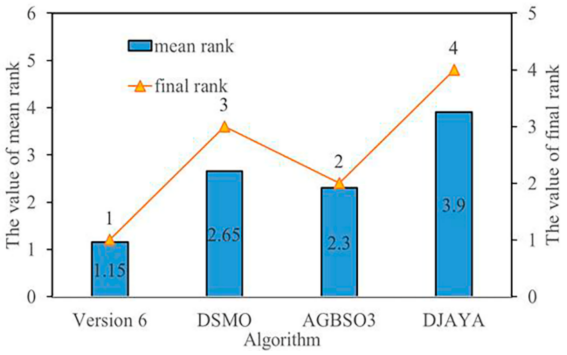


Figure 5. The ranks of the four participating algorithms.

The Friedman statistic, χ^2, with the significance level of 0.05 is summarized in Table 12, where χ^2 is 23.3. The critical value of $\chi^2_{0.05[3]}$ is 7.82 with $s - 1 = 3$ degrees of freedom. Thus, the null hypothesis is rejected at $\alpha = 0.05$.

Table 12. The results of the Friedman statistic when the algorithms number $s = 4$ and standard instances number $n = 10$.

α	χ^2	$\chi^2_{\alpha(s-1)}$	Null Hypothesis	Alternative Hypothesis
0.05	23.3	7.82	Reject	Accept

Holm's procedure was used to find the concrete pairwise comparisons that produced significant differences among the four participating algorithms. The results of Holm's procedure are summarized in Table 13.

Table 13. P-values by Holm post hoc test (Version 6 is the control method).

Algorithm	Unadjusted p-Value	Adjusted p-Value
DJAYA	2.00×10^{-6}	6.00×10^{-6}
DSMO	9.38×10^{-3}	1.88×10^{-2}
AGBSO3	4.64×10^{-2}	4.64×10^{-2}

As shown in Table 13, the unadjusted p-values and adjusted p-values are all lower than 0.05, which illustrates that there are significant differences between Version 6 and DJAYA, Version 6 and DSMO, and Version 6 and AGBSO3. From the analysis in Figure 5 and Tables 12 and 13, it can be confirmed that Version 6 performed significantly better than DJAYA, DSMO, and AGBSO3.

2. Compared with the algorithms with the local search algorithm:

For demonstrating the competitiveness of DCPA, 27 instances with cities from 29 up to 1577 were selected and compared with six algorithms in the literature, namely, DSSA [30], DSFLA [57], DBAL [40], D-GWO [29], ABC [26], and PACO-3Opt [58]. The parameters of the participating algorithms are shown in Table 14. Among them, the DSSA, DSFLA, and D-GWO use the 2-Opt local search algorithm, the ABC and PACO-3Opt use the 3-Opt local search algorithm, and the DBAL uses 2-Opt, 2.5-Opt, and 3-Opt local search algorithms.

Table 14. Parameter settings of algorithms.

Algorithm	Year	Reference	Parameters
DCPA	2022	This study	population size: 100, carnivorous plant: 25, prey: 75
DSSA	2022	[30]	population size: 50, PD: 0.4, alarm value: 0.8, PV: 0.2
DSFLA	2021	[57]	α = [1,5], β = [1,5], $Q = 1000$, $\rho = 0.1$, $N = 20$, $m = 10$
DBAL	2021	[40]	population size: 50, r_i: 0.5, A_i:0.5
D-GWO	2021	[29]	population size: 50
ABC	2018	[26]	$E_b = 20$, $O_b = 0.5E_b + 1$, $Mait3 = 10$, $lim = 5$
PACO-3Opt	2018	[58]	$Q = 30$, $\rho = 0.1$, $I = 5$, $\alpha = 3$, $\beta = 2$, $m = 10$

The comparison results in Table 15 show that our proposed method outperformed the most competitive methods in solving TSPs, as it can obtain better Best values and MPDA (%) among the seven participating algorithms. The DCPA could obtain 15 theoretical optimal solutions of 27 instances, and 4 of the remaining 12 instances were close to the optimal solutions (as their percentage deviation of the best found solution to the theoretical optimal solution was no more than 0.26%). In general, the seven algorithms are effective for TSP, but the performances of the comparison algorithms were challenged by the increasing problem scales. In some relatively large instances, such as d657, u724, pr1002, rl1323, and fl1577, the percentage deviation of the best found solution to the theoretical optimal

solution of our proposed algorithm was no more than 2.31%. However, the values of the comparison algorithm were all more than 2.37%. In addition, the MPDA (%) achieved by the DCPA were 0.69, 1.32, 0.83, 2.39, 2.18, and 0.9, which were superior to DSSA, DSFLA, DBAL, DGWO, ABC, and PACO-3Opt, respectively.

Table 15. Experiment results of the DCPA and the six other participating algorithms.

Name	BKS	Evaluation Indicators	DCPA	DSSA	DSFLA	DBAL	D-GWO	ABC	PACO-3Opt
att48	33,522	Best	**33,522**	**33,522**	**33,522**	**33,522**	**33,522**	**33,522**	**33,522**
		Worst	**33,522**	**33,522**	**33,522**	**33,522**	33,700	33,966	33,606
		Mean	**33,522**	**33,522**	**33,522**	**33,522**	33,560.5	33,632.35	33,541
		SD	**0.00**	**0.00**	**0.00**	**0.00**	47.28	126.78	33.32
		PDA (%)	**0.00**	**0.00**	**0.00**	**0.00**	0.11	0.33	0.06
eil51	426	Best	**426**	**426**	**426**	**426**	**426**	**426**	**426**
		Worst	428	**427**	428	**427**	428	436	428
		Mean	426.5	426.2	427.35	**426.1**	426.8	428.8	426.95
		SD	0.60	0.40	0.96	**0.30**	0.77	2.37	0.77
		PDA (%)	0.12	0.05	0.32	**0.02**	0.19	0.66	0.22
berlin52	7542	Best	**7542**	**7542**	**7542**	**7542**	**7542**	7542	**7542**
		Worst	**7542**	**7542**	**7542**	**7542**	**7542**	7759	**7542**
		Mean	**7542**	**7542**	**7542**	**7542**	**7542**	7572.7	**7542**
		SD	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	66.48	**0.00**
		PDA (%)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.41	**0.00**
eil76	538	Best	**538**	**538**	**538**	**538**	539	**538**	**538**
		Worst	540	543	543	**538**	550	571	544
		Mean	538.35	541.05	540.65	**538**	545.35	545.65	540.15
		SD	0.66	1.34	1.19	**0.00**	2.92	7.95	1.61
		PDA (%)	0.07	0.57	0.49	**0.00**	1.37	1.42	0.40
pr76	108,159	Best	**108,159**	**108,159**	**108,159**	**108,159**	108,159	**108,159**	**108,159**
		Worst	**108,159**	**108,159**	108,961	109,186	109,043	114,574	109,043
		Mean	**108,159**	**108,159**	108,564.9	108,571.90	108,462.9	109,053.7	108,327.7
		SD	**0.00**	**0.00**	250.35	341.54	348.81	1507.54	303.48
		PDA (%)	**0.00**	**0.00**	0.38	0.38	0.28	0.83	0.16
rat99	1211	Best	**1211**	**1211**	**1211**	**1211**	1213	**1211**	**1211**
		Worst	1220	**1214**	1215	**1214**	1246	1292	1232
		Mean	1212	1211.45	1212.95	**1211.4**	1229.3	1233.55	1215.65
		SD	2.69	0.87	1.05	0.8	7.78	18.91	5.72
		PDA (%)	0.08	0.04	0.16	0.03	1.51	1.86	0.38
kroC100	20,749	Best	**20,749**	**20,749**	**20,749**	**20,749**	20,749	**20,749**	**20,749**
		Worst	**20,749**	**20,749**	208,75	20,769	20,985	21,471	20,983
		Mean	**20,749**	**20,749**	208,19.55	20,750	20,802.75	20,948.05	20,792.25
		SD	**0.00**	**0.00**	36.2	4.36	74.95	190.15	63.39
		PDA (%)	**0.00**	**0.00**	0.34	**0.00**	0.26	0.96	0.21
pr107	44,303	Best	**44,303**	**44,303**	**44,303**	**44,303**	44,342	44,387	44,387
		Worst	**44,387**	44,358	44,381	**44,387**	44,621	48,553	44,577
		Mean	**44,309.4**	44,316.45	44,317.85	44,311.4	44,453.6	45,117.5	44,492
		SD	26.64	**19.93**	22.56	30.34	87.11	995.36	67.09
		PDA (%)	**0.01**	0.03	0.03	0.02	0.34	1.84	0.43

Table 15. *Cont.*

Name	BKS	Evaluation Indicators	DCPA	DSSA	DSFLA	DBAL	D-GWO	ABC	PACO-3Opt
pr124	59,030	Best	**59,030**	**59,030**	**59,030**	**59,030**	59,030	**59,030**	**59,030**
		Worst	**59,030**	**59,030**	59,164	59,293	59,463	63,072	59,087
		Mean	**59,030**	**59,030**	59,045.9	59,047.75	59,175.5	60,477.3	59,048.95
		SD	**0.00**	**0.00**	32.88	58.06	132.94	1209.66	23.7
		PDA (%)	**0.00**	**0.00**	0.03	0.03	0.25	2.45	0.03
ch130	6110	Best	**6110**	**6110**	6128	**6110**	6142	**6110**	6121
		Worst	6172	**6138**	6183	6147	6245	6302	6193
		Mean	6134.6	**6120.8**	6170.05	6125.45	6199.5	6197.35	6157.75
		SD	20.35	**8.04**	14.94	11.54	28.97	48.83	19.91
		PDA (%)	0.40	**0.18**	0.98	0.25	1.46	1.43	0.78
pr144	58,537	Best	**58,537**	**58,537**	58,570	**58,537**	58,537	**58,537**	**58,537**
		Worst	**58,537**	**58,537**	58,636	58,554	58,797	62,192	58,590
		Mean	**58,537**	**58,537**	58,611.25	58,537.85	58,617.4	59,376.3	58,539.65
		SD	**0.00**	**0.00**	23.11	3.71	89.07	992.53	11.57
		PDA (%)	**0.00**	**0.00**	0.13	**0.00**	0.14	1.43	0.00
kroB150	26,130	Best	**26,130**	**26,130**	26,224	**26,130**	26,304	26,132	**26,130**
		Worst	**26,229**	26,269	26,492	26,261	26,576	28,433	26,438
		Mean	**26,149.45**	26,200.3	26,413.95	26,197.55	26,432	26,781.7	26,251.55
		SD	**28.85**	31.49	59.14	44.86	103.62	609.95	97.99
		PDA (%)	**0.07**	0.27	1.09	0.26	1.16	2.49	0.47
pr152	73,682	Best	**73,682**	**73,682**	73,818	**73,682**	73,780	73,780	**73,682**
		Worst	73,818	**73,686**	74,227	74,105	74,424	77,122	74,215
		Mean	73,756.8	**73,682.2**	73,961.4	73,866.75	74,080.5	74,877.85	73,889.25
		SD	69.03	**0.87**	131.35	144.91	246.86	873.65	131.26
		PDA (%)	0.10	**0.00**	0.38	0.25	0.54	1.62	0.28
rat195	2323	Best	**2329**	2340	2344	2336	2448	2362	2333
		Worst	**2347**	2371	2359	2379	2526	2466	2366
		Mean	**2339.4**	2360.25	2350.15	2362.7	2398.25	2408.55	2349.4
		SD	7.26	7.16	**4.05**	8.94	17.17	29.09	8.89
		PDA (%)	**0.71**	1.60	1.17	1.71	3.24	3.68	1.14
kroA200	29,368	Best	**29,368**	29,479	29,502	29,401	29,605	29,451	29,411
		Worst	**29,459**	29,627	29,653	29,558	30,134	30,835	29,802
		Mean	**29,399.1**	29,545.55	29,567.7	29,443.75	29,867.8	30,134.3	29,572.95
		SD	**26.06**	39.34	46.59	40.39	161.07	336.5	111.47
		PDA (%)	**0.11**	0.60	0.68	0.26	1.70	2.61	0.70
tsp225	3916	Best	**3916**	3916	3964	3930	3961	3971	3933
		Worst	**3963**	3987	4015	3995	4066	4175	3981
		Mean	**3936.8**	3970.2	3999.2	3966.6	4020.4	4035.7	3956.95
		SD	17.1	17.16	**11.21**	17.7	26.93	50.53	13
		PDA (%)	**0.53**	1.38	2.12	1.29	2.67	3.06	1.05
pr299	48,191	Best	**48,229**	48,648	49,416	48,670	50,606	48,909	48,887
		Worst	**48,620**	49,149	50,082	49,063	51,876	52,626	49,665
		Mean	**48,373.5**	48,917.25	49,716.2	48,865.45	51,468	49,830.45	49,191.25
		SD	**94.62**	133.29	170.26	127.63	330.01	884.1	218.58
		PDA (%)	**0.38**	1.51	3.16	1.40	6.80	3.40	2.08
lin318	420,29	Best	**42,071**	42,402	43,028	42,312	43,980	42,920	42,475
		Worst	**42,554**	42,986	43,451	42,801	45,180	44,113	43,099
		Mean	**42,351.25**	42,726.8	43,272.9	42,563.45	44,738.25	43,606.95	42,810.9
		SD	153.61	135.28	122.61	**116.89**	304.42	358.82	193.01
		PDA (%)	**0.77**	1.66	2.96	1.27	6.45	3.75	1.86

Table 15. *Cont.*

Name	BKS	Evaluation Indicators	DCPA	DSSA	DSFLA	DBAL	D-GWO	ABC	PACO-3Opt
fl417	11,861	Best	**11,877**	11,992	11,987	12,009	12,166	11,929	11,906
		Worst	**11,906**	12,051	12,075	12,177	12,440	13,628	12,050
		Mean	**11,891.75**	12,021.85	12,032.6	12,069.6	12,273.65	12,528.5	11,960.75
		SD	**10.01**	17.11	22.4	45.15	72.54	593.48	35.16
		PDA (%)	**0.26**	1.36	1.45	1.76	3.48	5.63	0.84
pcb442	50,778	Best	**50,998**	51,955	53,030	51,828	52,309	52,340	52,234
		Worst	**51,875**	52,387	53,796	52,536	53,485	54,071	53,153
		Mean	**51,376.05**	52,187.4	53,460.25	52,113.85	52,786.75	52,818.55	52,664.65
		SD	193.49	**121.12**	216.41	165.17	339.6	446	306.49
		PDA (%)	**1.18**	2.78	5.28	2.63	3.96	4.02	3.72
d493	35,002	Best	**35,327**	35,691	36,500	35,803	35,768	35,691	36,148
		Worst	**36,045**	36,105	37,038	36,249	36,291	37,062	36,651
		Mean	**35,685.85**	35,904.35	36,806.95	35,971.5	36,124.5	36,361.65	36,365.8
		SD	164.81	**95.92**	132.71	113.38	132.3	348.15	141.33
		PDA (%)	**1.95**	2.58	5.16	2.77	3.21	3.88	3.90
d657	48,912	Best	**49,483**	50,493	51,461	50,481	52,854	50,469	50,558
		Worst	**50,268**	50,960	52,334	51,244	54,095	54,431	51,432
		Mean	**49,954.8**	50,736.5	51,865.1	50,935.45	53,482.8	51,451.45	50,981.35
		SD	259.18	**133.79**	224.87	157.9	325.46	867.34	246.45
		PDA (%)	**2.13**	3.73	6.04	4.14	9.34	5.19	4.23
u724	41,910	Best	**42,514**	43,498	43,776	43,718	43,242	43,191	43,586
		Worst	**43,340**	44,036	44,520	44,280	43,929	44,733	44,160
		Mean	**42,859.65**	43,822.55	44,192	44,008.8	43,610.85	43,885.95	43,849.25
		SD	223.26	161.46	182.92	**151.69**	199.12	357.65	163
		PDA (%)	**2.27**	4.56	5.45	5.01	4.06	4.71	4.63
rat783	8806	Best	**9102**	9220	9262	9194	9190	9232	9158
		Worst	**9240**	9343	9384	9406	9379	9603	9271
		Mean	**9171.15**	9291.15	9324.6	9337.35	9246.45	9335.85	9241.3
		SD	48.58	32.58	32.72	54.59	53.42	100.19	**31.26**
		PDA (%)	**4.15**	5.51	5.89	6.03	5.00	6.02	4.94
pr1002	259,045	Best	**265,037**	271,315	274,076	271,780	284,870	270,993	269,286
		Worst	**268,393**	274,092	277,273	274,884	289,243	284,976	273,567
		Mean	**266,654.7**	272,733	275,562.5	273,650.6	287,131.7	274,699.1	271,607.6
		SD	861.72	**697.16**	853.03	838.31	1332.26	3986.59	1195.93
		PDA (%)	**2.94**	5.28	6.38	5.64	10.84	6.04	4.85
rl1323	270,199	Best	**273,796**	279,779	281,554	280,997	297,682	279,315	280,227
		Worst	**278,664**	284,327	286,376	286,261	304,697	295,711	286,598
		Mean	**276,211.5**	282,183	283,997.3	284,064.8	301,421.3	285,740.6	283,804.3
		SD	**1272.4**	1125.22	1508.6	1421.58	2085.74	3683.21	1682.25
		PDA (%)	**2.23**	4.44	5.11	5.13	11.56	5.75	5.04
fl1577	22,249	Best	**225,86**	22,776	22,914	23,141	23,811	23,242	22,967
		Worst	**23,000**	23,170	23,194	23,575	24,110	24,520	23,723
		Mean	**22,780.5**	23,036.25	23,008.05	23,370.9	23,926.9	23,649.2	23,309.5
		SD	**84.63**	106.12	97.8	126.36	100.06	323.09	244.68
		PDA (%)	**2.39**	3.54	3.41	5.04	7.54	6.29	4.77
MPDB / MPDA (%)			**0.48/0.85**	1.14/1.54	1.67/2.17	1.23/1.68	2.57/3.24	1.35/3.03	1.21/1.75

Note: the best is set in bold, the MPDB means the mean percentage deviation of the best found solution to the best known solution, and the MPDA (%) means the mean of PDA.

Then, the Friedman rank test was used to order the participating algorithms based on their performance in solving TSPs. The order of the seven algorithms is shown in Figure 6.

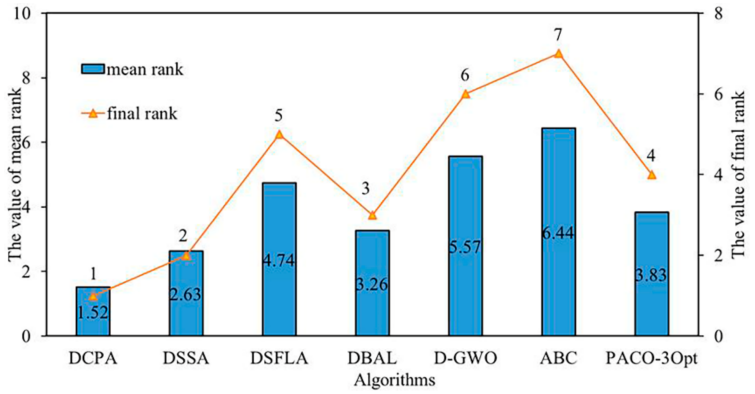


Figure 6. The ranks of the seven participating algorithms.

The Friedman statistic χ^2 with the significance level of 0.05 is summarized in Table 16, where χ^2 is 105.32. The critical value of $\chi^2_{0.05[6]}$ is 12.59 with $s - 1 = 6$ degrees of freedom. Thus, the null hypothesis is rejected at $\alpha = 0.05$.

Table 16. The results of the Friedman statistic when the algorithms number $s = 7$ and standard instances number $n = 27$.

α	χ^2	$\chi^2_{\alpha(s-1)}$	Null Hypothesis	Alternative Hypothesis
0.05	105.32	12.59	Reject	Accept

Lastly, Holm's procedure was used to find the concrete pairwise comparisons that produced significant differences among the seven participating algorithms. The results of Holm's procedure are summarized in Table 17.

Table 17. *p*-values by Holm post hoc test (DCPA is the control method).

Algorithm	Unadjusted *p*-Value	Adjusted *p*-Value
ABC	0	0
DGWO	5.28×10^{-12}	1.70×10^{-11}
DSFLA	4.24×10^{-8}	8.24×10^{-8}
PACO-3Opt	8.20×10^{-5}	2.46×10^{-4}
DBAL	3.07×10^{-3}	6.14×10^{-3}
DSSA	5.88×10^{-2}	5.88×10^{-2}

The results obtained from Holm's procedure show that the DCPA did not show a significant difference from the DSSA for the adjusted *p*-value and unadjusted *p*-value, which are more than 0.05 at a 95% confidence level. However, compared to the DSSA, the DCPA significantly enhanced the MPDA and MPDB, which verified the algorithm's superior performance. The statistical test shows that the DCPA was significantly different from the other five comparison algorithms, with all *p*-values less than 0.05 at a 95% confidence level. The analyses above demonstrate the outstanding performance of the DCPA.

3. Comparison of convergence speed

For demonstrating the convergence performance of the DCPA, the convergence analysis of the participating algorithms was analyzed with average computation time and convergence curves. The average computation times of 16 instances in 20 runs are summarized in Table 18, and the graphical representation of the convergence analysis is depicted in Figure 7. The T_{avg} in Table 18 denotes the average time of 16 instances. The *x*-axis in Figure 7 is taken as the running time, and the *y*-axis is taken as the length of the route.

Table 18. The computation times of the participating algorithms with city sizes less than 250.

Instance	Algorithms						
	DCPA (s)	DSSA (s)	DSFLA (s)	DBAL (s)	D-GWO (s)	ABC (s)	PACO-3Opt (s)
att48	**0.73**	1.04	0.98	8.43	7.18	7.48	3.75
eil51	12.31	**1.88**	17.94	6.50	16.21	18.40	14.89
berlin52	0.42	**0.23**	0.46	1.33	2.19	10.14	0.82
eil76	9.67	**4.56**	20.11	7.90	21.08	19.10	17.89
pr76	**2.33**	2.73	19.98	19.74	14.88	16.53	11.23
rat99	**8.71**	11.06	19.32	13.38	20.52	19.82	19.39
kroC100	**4.75**	4.87	47.73	19.66	37.76	44.83	32.23
pr107	**13.52**	31.66	29.92	16.40	51.13	50.08	50.49
pr124	**2.22**	2.56	28.97	21.04	47.08	47.17	32.00
ch130	**42.52**	46.99	50.21	46.69	52.90	49.38	50.46
pr144	4.23	**1.81**	50.22	18.49	40.30	45.11	23.28
kroB150	**41.82**	48.21	50.32	49.40	54.87	50.21	48.79
pr152	23.60	**17.38**	50.34	44.32	53.47	50.16	47.78
rat195	**50.04**	50.09	50.49	50.51	52.37	50.32	51.46
kroA200	**83.20**	100.08	100.59	100.43	100.92	100.44	101.60
tsp225	**89.90**	100.10	100.52	100.66	103.01	100.36	100.04
T_{avg}	**24.37**	26.58	39.88	32.80	42.24	42.47	37.88

Note: the best is set in bold.

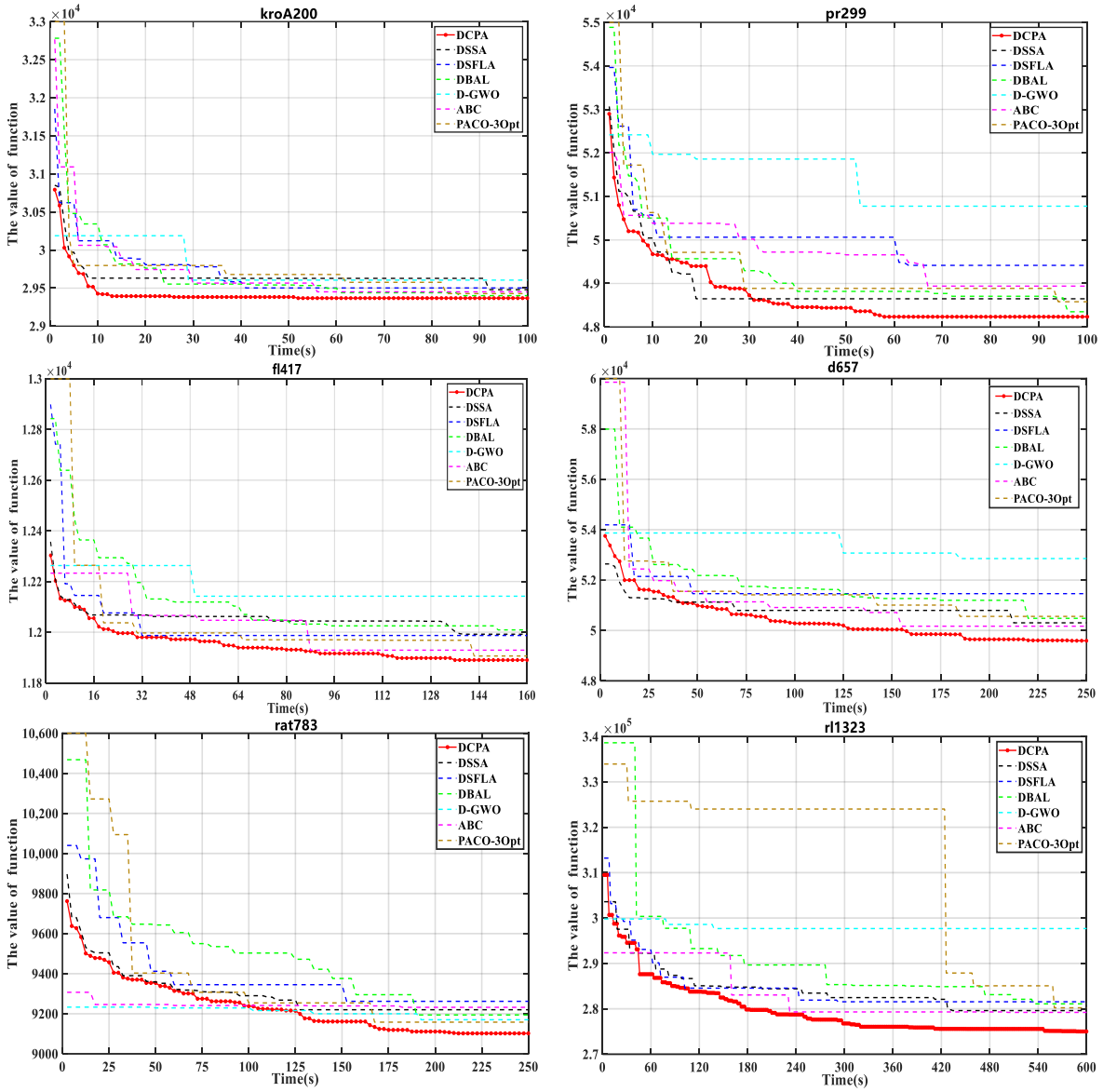


Figure 7. The convergence cure of the participating algorithms.

The analyses above show that the DCPA can achieve a higher quality than the comparison algorithms, and the results in Table 18 confirmed that the DCPA also performs faster in computational time than the others, as the DCPA won on 11 of 16 instances. The T_{avg} of the DCPA was the shortest among the seven algorithms, which means that the proposed algorithm can exhibit superior quality solutions in a much faster computational time than the other participating algorithms.

As depicted in Figure 7, the convergence speed of the DCPA was slower than the DSSA on pr299 and d657 and slower than ABC and D-GWO on fl417, rat783, and rl1323 in the early stage of iteration. ABC and D-GWO call the local optimization algorithm various times in each iteration. Therefore, these two algorithms converge faster in the early stage of the iteration, but the global search ability decreases with the iteration time, which easily falls into the local optimum. The DSSA employed order-based decoding, which exhibits strong global search ability in the early stage and converges faster. However, with the iteration time increase, the local search ability of the DSSA becomes weak and it converges prematurely. The DCPA takes into account both the global and local search ability during the iteration, and it maintains a high-level convergence speed in all participating algorithms.

6. Conclusions and Future Work

This work proposes a new discrete carnivorous plant algorithm with similarity elimination for the TSP, which considers intelligence and heuristics in both the intensification and diversification stages and finally increases the overall performance of the algorithm. The DCPA presents a new individual generation method that redefines the addition, subtraction, and multiplication operators. The newly generated individual can not only satisfy the discrete properties of the TSP but also maintains the good characteristics of the better individual in the parents, which helps to enhance the search precision. After that, a simple sorting grouping method is proposed, which randomly selects carnivorous plants and prey for updating, and reduces the computational complexity and the assimilation speed. Then, an adaptive variable attraction probability is proposed, and the high probability of prey updating enhances the exploration capability in the early stage of iteration. At the late stage, a high probability of carnivorous plant growth enhances the exploitation capability. Finally, the similarity-eliminating operation based on the same quantity of city sequences and route length is added to the DCPA, which effectively reduces the number of similar and identical individuals, helps to maintain population diversity, and reduces the probability of stagnation.

To verify the effectiveness of the improvements in the DCPA, two sets of experiments were designed. The first experiment compared the new generation method with the two generation methods in the literature. Then, the self-comparison of different versions of the DCPA was carried out in the second experiment. From the results, the following conclusions can be drawn: (1) the new generation method can solve the TSP effectively; and (2) simple sorting grouping, adaptive attraction probability, and the similarity-eliminating operation play positive roles in the enhancement of the algorithm's performance.

To assess the performance of the proposed algorithm, the algorithm was compared with nine algorithms. The parametric and nonparametric statistical tests proved that the proposed algorithm has superior performance in solution quality. In addition, the convergence curves show that the DCPA converges more quickly than the other comparison algorithms. The above analyses demonstrate that the DCPA is an effective and competitive choice to solve the TSP.

The DCPA was only verified on the instances of the TSP and the parameters were determined with trial and error. It is not suitable for other optimization problems with discrete domains. In future work, the proposed DCPA will be tested for more complex discrete problems, such as multi-objective traveling salesman problems and multi-traveling salesman problems. In addition, the optimal parameter combination, the research work

on the running time, the accuracy of the algorithm, and the applied research can be carried out.

Author Contributions: Conceptualization, P.-L.Z.; X.-B.S. and J.-Q.W.; methodology, P.-L.Z. and X.-B.S.; software, P.-L.Z. and X.-B.S.; validation, P.-L.Z.; X.-B.S. and J.-Q.W.; formal analysis, H.-H.S. and J.-L.B.; investigation, P.-L.Z. and X.-B.S.; resources, P.-L.Z.; X.-B.S. and J.-Q.W.; data curation, H.-H.S., J.-L.B. and H.-Y.Z.; writing—original draft preparation, P.-L.Z. and X.-B.S.; writing—review and editing, P.-L.Z.; X.-B.S. and J.-Q.W.; visualization, P.-L.Z.; X.-B.S. and J.-Q.W.; supervision, J.-Q.W.; project administration, P.-L.Z. and X.-B.S.; funding acquisition, J.-Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Social Science Fund of China, grant number 21BGL17.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Papadimitriou, C.H.; Steiglitz, K. *Combinatorial Optimization: Algorithms and Complexity*; Courier Corporation: Chelmsford, MA, USA, 1998.
2. Hartmanis, J. Computers and intractability: A guide to the theory of np-completeness (Michael R. Garey and David S. Johnson). *Siam Rev.* **1982**, *24*, 90. [CrossRef]
3. Eldos, T.; Kanan, A.; Nazih, W.; Khatatbih, A. Adapting the Ant Colony Optimization Algorithm to the Printed Circuit Board Drilling Problem. *World Comput. Sci. Inf. Technol. J.* **2013**, *3*, 100–104.
4. An, H.; Li, W. Synthetically improved genetic algorithm on the traveling salesman problem in material transportation. In Proceedings of the 2011 International Conference on Electronic & Mechanical Engineering and Information Technology, Harbin, China, 12–14 August 2011; pp. 3368–3371.
5. Savla, K.; Frazzoli, E.; Bullo, F. Traveling Salesperson Problems for the Dubins Vehicle. *IEEE Trans. Autom. Control* **2008**, *53*, 1378–1391. [CrossRef]
6. Cheng, H.; Yang, S. Genetic algorithms with elitism-based immigrants for dynamic shortest path problem in mobile ad hoc networks. In Proceedings of the 2009 IEEE Congress on Evolutionary Computation, Trondheim, Norway, 18–21 May 2009; pp. 3135–3140.
7. Gharehchopogh, F.S. Advances in tree seed algorithm: A comprehensive survey. *Arch. Comput. Methods Eng.* **2022**, *29*, 3281–3304. [CrossRef]
8. Ghafori, S.; Gharehchopogh, F.S. Advances in spotted hyena optimizer: A comprehensive survey. *Arch. Comput. Methods Eng.* **2021**, *29*, 1569–1590. [CrossRef]
9. Dantzig, G.; Johnson, R.F. Solution of a Large-Scale Traveling-Salesman Problem. *J. Oper. Res. Soc. Am.* **1954**, *2*, 393–410. [CrossRef]
10. Bellman, R.E.; Dreyfus, S.E. *Applied Dynamic Programming*; Princeton University Press: Princeton, NJ, USA, 2015.
11. Padberg, M.; Rinaldi, G. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Oper. Res. Lett.* **1987**, *6*, 1–7. [CrossRef]
12. Kizilateş, G.; Nuriyeva, F. On the nearest neighbor algorithms for the traveling salesman problem. In *Advances in Computational Science, Engineering and Information Technology*; Springer: Cham, Switzerland, 2013; pp. 111–118.
13. Kanellakis, P.-C.; Papadimitriou, C.H. Local search for the asymmetric traveling salesman problem. *Oper. Res.* **1980**, *28*, 1086–1099. [CrossRef]
14. Gu, J.; Huang, X. Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP). *IEEE Trans. Syst. Man Cybern.* **1994**, *24*, 728–735.
15. Wang, Y.; Han, Z. Ant colony optimization for traveling salesman problem based on parameters optimization. *Appl. Soft Comput.* **2021**, *107*, 107439. [CrossRef]
16. Shahadat, A.S.B.; Akhand, M.A.H.; Kamal, M.A.S. Visibility Adaptation in Ant Colony Optimization for Solving Traveling Salesman Problem. *Mathematics* **2022**, *10*, 2448. [CrossRef]
17. Liu, Q.; Du, S.; Wyk, B.; Sun, Y. Niching particle swarm optimization based on Euclidean distance and hierarchical clustering for multimodal optimization. *Nonlinear Dyn.* **2020**, *99*, 2459–2477. [CrossRef]

18. Deng, Y.; Xiong, J.; Wang, Q. A Hybrid Cellular Genetic Algorithm for the Traveling Salesman Problem. *Math. Probl. Eng.* **2021**, *2021*, 6697598. [CrossRef]
19. Wang, J.; Ersoy, O.K.; He, M.; Wang, F. Multi-offspring genetic algorithm and its application to the traveling salesman problem. *Appl. Soft Comput.* **2016**, *43*, 415–423. [CrossRef]
20. Nagata, Y.; Soler, D. A new genetic algorithm for the asymmetric traveling salesman problem. *Expert Syst. Appl.* **2012**, *39*, 8947–8953. [CrossRef]
21. Mi, M.; Xue, H.; Ming, Z.; Yu, G. An Improved Differential Evolution Algorithm for TSP Problem. In Proceedings of the Intelligent Computation Technology and Automation, International Conference, Changsha, China, 11–12 May 2010.
22. Gharehchopogh, F.S.; Namazi, M.; Ebrahimi, L.; Abdollahzadeh, B. Advances in Sparrow Search Algorithm: A Comprehensive Survey. *Arch. Comput. Methods Eng.* **2022**, 1–29. [CrossRef]
23. Zhong, Y.; Lin, J.; Wang, L.; Zhang, H. Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem. *Inf. Sci.* **2017**, *421*, 70–84. [CrossRef]
24. Gharehchopogh, F.S. An Improved Tunicate Swarm Algorithm with Best-random Mutation Strategy for Global Optimization Problems. *J. Bionic Eng.* **2022**, *19*, 1177–1202. [CrossRef]
25. Akhand, M.; Ayon, S.I.; Shahriyar, S.A.; Siddique, N.; Adeli, H. Discrete Spider Monkey Optimization for Traveling Salesman Problem. *Appl. Soft Comput.* **2019**, *86*, 105887. [CrossRef]
26. Khan, I.; Maiti, M.K. A swap sequence based Artificial Bee Colony algorithm for Traveling Salesman Problem. *Swarm Evol. Comput.* **2019**, *44*, 428–438. [CrossRef]
27. Gunduz, M.; Aslan, M. DJAYA: A discrete Jaya algorithm for solving traveling salesman problem. *Appl. Soft Comput.* **2021**, *105*, 107275. [CrossRef]
28. Osaba, E.; Del Ser, J.; Sadollah, A.; Bilbao, M.N.; Camacho, D. A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. *Appl. Soft Comput.* **2018**, *71*, S1568494618303818.
29. Panwar, K.; Deep, K. Discrete Grey Wolf Optimizer for symmetric travelling salesman problem. *Appl. Soft Comput.* **2021**, *105*, 107298. [CrossRef]
30. Zhang, Z.; Han, Y. Discrete sparrow search algorithm for symmetric traveling salesman problem. *Appl. Soft Comput.* **2022**, *118*, 108469. [CrossRef]
31. Ezugwu, A.E.-S.; Adewumi, A.O.; Frîncu, M.E. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. *Expert Syst. Appl.* **2017**, *77*, 189–210. [CrossRef]
32. Sörensen, K. Metaheuristics—The metaphor exposed. *Int. Trans. Oper. Res.* **2015**, *22*, 3–18. [CrossRef]
33. Ong, K.M.; Ong, P.; Sia, C.K. A carnivorous plant algorithm for solving global optimization problems. *Appl. Soft Comput.* **2021**, *98*, 106833. [CrossRef]
34. Stodola, P.; Otřísal, P.; Hasilová, K. Adaptive Ant Colony Optimization with node clustering applied to the Travelling Salesman Problem. *Swarm Evol. Comput.* **2022**, *70*, 101056. [CrossRef]
35. Yong, W. The hybrid genetic algorithm with two local optimization strategies for traveling salesman problem. *Comput. Ind. Eng.* **2014**, *70*, 124–133.
36. Ha, Q.M.; Deville, Y.; Pham, Q.D.; Hà, M. A hybrid genetic algorithm for the traveling salesman problem with drone. *J. Heuristics* **2020**, *26*, 219–247. [CrossRef]
37. Wang, Y.; Wu, Y.W.; Xu, N. Discrete symbiotic organism search with excellence coefficients and self-escape for traveling salesman problem. *Comput. Ind. Eng.* **2019**, *131*, 269–281. [CrossRef]
38. Kóczy, L.T.; Földesi, P.; Tüű-Szabó, B. Enhanced discrete bacterial memetic evolutionary algorithm—An efficacious metaheuristic for the traveling salesman optimization. *Inf. Sci.* **2018**, *460–461*, 389–400. [CrossRef]
39. Zhong, Y.; Wang, L.; Lin, M.; Zhang, H. Discrete pigeon-inspired optimization algorithm with Metropolis acceptance criterion for large-scale traveling salesman problem. *Swarm Evol. Comput.* **2019**, *48*, 134–144. [CrossRef]
40. Saji, Y.; Barkatou, M. A discrete bat algorithm based on Lévy flights for Euclidean Traveling Salesman Problem. *Expert Syst. Appl.* **2021**, *172*, 114639. [CrossRef]
41. Benyamin, A.; Farhad, S.G.; Saeid, B. Discrete farmland fertility optimization algorithm with metropolis acceptance criterion for traveling salesman problems. *Int. J. Intell. Syst.* **2021**, *36*, 1270–1303. [CrossRef]
42. Al-Gaphari, G.H.; Al-Amry, R.; Al-Nuzaili, A.S. Discrete crow-inspired algorithms for traveling salesman problem. *Eng. Appl. Artif. Intell.* **2021**, *97*, 104006. [CrossRef]
43. Zhang, Z.; Yang, J. A discrete cuckoo search algorithm for traveling salesman problem and its application in cutting path optimization. *Comput. Ind. Eng.* **2022**, *169*, 108157. [CrossRef]
44. Samanlioglu, F.; Jr, W.G.F.; Kurz, M.E. A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem. *Comput. Ind. Eng.* **2008**, *55*, 439–449. [CrossRef]
45. Ezugwu, A.E.-S.; Adewumi, A.O. Discrete symbiotic organisms search algorithm for travelling salesman problem. *Expert Syst. Appl.* **2017**, *87*, 70–78. [CrossRef]
46. Ali, I.M.; Essam, D.; Kasmarik, K. A novel design of differential evolution for solving discrete traveling salesman problems. *Swarm Evol. Comput.* **2020**, *52*, 100607. [CrossRef]
47. Gharehchopogh, F.S.; Abdollahzadeh, B. An efficient harris hawk optimization algorithm for solving the travelling salesman problem. *Clust. Comput.* **2021**, *25*, 1981–2005. [CrossRef]

48. Zhang, P.; Wang, J.; Tian, Z.; Sun, S.; Li, J.; Yang, J. A genetic algorithm with jumping gene and heuristic operators for traveling salesman problem. *Appl. Soft Comput.* **2022**, *127*, 109339. [CrossRef]
49. Iqbal, Z.; Bashir, N.; Hussain, A.; Cheema, S.A. A novel completely mapped crossover operator for genetic algorithm to facilitate the traveling salesman problem. *Comput. Math. Methods* **2020**, *2*, e1122. [CrossRef]
50. Loureno, H.R.; Martin, O.; Stützle, T. Iterated Local Search. In *Handbook of Metaheuristics*; Springer: Boston, MA, USA, 2003; Volume 57.
51. Croes, G.A. A Method for Solving Traveling-Salesman Problems. *Oper. Res.* **1958**, *6*, 791–812. [CrossRef]
52. Lin, S. An effective heuristic algorithm for the traveling salesman problem. *Ann. Ops. Res.* **1973**, *21*, 498–516. [CrossRef]
53. Friedman, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Publ. Am. Stat. Assoc.* **1939**, *32*, 675–701. [CrossRef]
54. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **1940**, *11*, 86–92. [CrossRef]
55. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
56. Wu, C.; Fu, X. An Agglomerative Greedy Brain Storm Optimization Algorithm for Solving the TSP. *IEEE Access* **2020**, *8*, 201606–201621. [CrossRef]
57. Huang, Y.; Shen, X.-N.; You, X. A discrete shuffled frog-leaping algorithm based on heuristic information for traveling salesman problem. *Appl. Soft Comput.* **2021**, *102*, 107085. [CrossRef]
58. Gülcü, A.; Mahi, M.; Baykan, M.K.; Kodaz, H. A parallel cooperative hybrid method based on ant colony optimization and 3-Opt algorithm for solving traveling salesman problem. *Soft Comput.* **2018**, *22*, 1669–1685. [CrossRef]