Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 19(5): 4458–4480.
DOI: 10.3934/mbe.2022206
Received: 13 October 2021
Revised: 08 January 2022
Accepted: 14 February 2022
Published: 02 March 2022

Research article

Circular Jaccard distance based multi-solution optimization for traveling salesman problems

Hui Li*, Mengyao Zhang and Chenbo Zeng

Department of Computer Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China

* **Correspondence:** Email: ray@mail.buct.edu.cn.

Abstract: Traveling salesman problem is a widely studied NP-hard problem in the field of combinatorial optimization. Many and various heuristics and approximation algorithms have been developed to address the problem. However, few studies were conducted on the multi-solution optimization for traveling salesman problem so far. In this article, we propose a circular Jaccard distance based multi-solution optimization (CJD-MSO) algorithm based on ant colony optimization to find multiple solutions for the traveling salesman problem. The CJD-MSO algorithm incorporates "distancing" niching technique with circular Jaccard distance metric which are both proposed in this paper for the first time. Experimental results verify that the proposed algorithm achieves good performance on both quality and diversity of the optimal solutions.

Keywords: traveling salesman problem; multimodal optimization; multi-solution optimization; Jaccard distance; metaheuristics

1. Introduction

The traveling salesman problem (TSP) is one of the most intensively studied combinatorial optimization problems in the field of graph theory and operations research. It is believed that the TSP was first studied by Hamilton and Kirkman [1]. Since then, many and various algorithms and approximation algorithms have been proposed.

Given a set of nodes (cities) and the pairwise cost (or travel distance), the TSP is to find the best possible cycle of visiting all the cities and returning to the starting point that minimize the total travel cost. The general solution algorithm that guarantees the shortest path has exponential complexity, and no known efficient algorithm exists to date, since TSP is also an NP-hard problem.

The TSP naturally finds many applications in the field of transportation and logistics, such as arranging school bus routes to pick up the children [2], scheduling stacker cranes in warehouses [3], and delivering ordered meals to the customers [4]. Slightly modified, TSP can be applied to some problems which are unlike the path finding problems at first sight. For example, in the DNA sequencing problem [5], the node represents DNA fragments, and distance cost is defined as the similarity measure between DNA fragments.

Various methods have been proposed to solve TSP, such as violence enumeration, dynamic programming (DP) [6], and Hopfield neural network (HNN) [7]. Violence enumeration, as the simplest method, has the time complexity $O(n!)$. DP decomposes the problem into couples of sub-problems which has much smaller time complexity as $O(n^2 2^n)$ [6]. As an intelligent method, Hopfield neural network algorithm(HNN) [7] handles the permutation matrix which is mapped to the legal solution of TSP, and minimizes the energy function intelligently. The drawback of this method is that it may suffer from slow convergence and low accuracy when used to solve the TSP [8].

Using swarm intelligence [9, 10] to address TSP has become a new trend in the past decades. Many metaheuristics has been reported excelling in solving TSP, such as genetic algorithm (GA) [11, 12], particle swarm optimization (PSO) [13, 14], and ant colony optimization (ACO) [15]. Ant colony optimization (ACO) is proposed by Marco Dorigo in 1992, which is inspired by the ant colony behavior of searching food.

ACO is one of the most frequently studied algorithm for solving TSP. Fevrier Valdez et al. [16] implemented the algorithms elitist ant system (EAS) and rank based ant system (ASrank); Gao Shupeng et al. [17] proposed a pheromone initialization strategy; Zhang Yuru et al. [18] developed an improved 2-opt and hybrid algorithm based on ACO; Dewantoro et al. [19] used tabu search algorithm as the local search in ACO.

In recent years, as researchers made slow progress in developing better methods for TSP, some researchers shifted their interests to some promoted problems, such as vehicle routing problem (VRP) and multiple traveling salesman problem (MTSP). MTSP means that multiple salesmen only visit a certain number of cities once and only once, and then return to their original cities with the smallest total cost. MTSP is closely related to some optimization problems such as task assignment [20].

Optimization problems with multiple solutions widely exist in the real world, and TSP is no exception. In the TSP, the salesman prefers couples of optimal solutions at hands in case that some solutions become unavailable for some reasons. Ting Huang et al. [21] proposed a benchmark to study the performance of algorithms for the multi-solution TSP (MSTSP), though few algorithms had been proposed at that time. NMA [22] integrates niche strategies with the memetic algorithm to address the MSTSP. NGA [21] groups related individuals and performs the GA operations including crossover and mutation in each group. NACS [23] is based on the ACO, in which ants are guided to search for different areas through multiple pheromone matrices.

Since ACO is the most successful metaheuristics reported to solve TSP, we also use the ACO in this paper as the base optimization method to address the MSTSP. We use 2-opt strategy to improve the local search ability. The 2-opt strategy provides an effective search operation, while requires less computation than 3-opt or more. We propose a new distance metric named "circular Jaccard distance" instead of other frequently used distance metrics, which brings a considerable improvement in the optimization performance. The circular Jaccard distance is defined as the minimal distances between two paths in various transformation modes such as flip and circular shift. We test the proposed method on the benchmark problems of 25 MSTSP instances, with two evaluation criteria designed for MSTSP. Experimental results indicate that the proposed method substantially improves the performance of multi-solution optimization.

The rest of the article is organized as follows. Section 2 introduces TSP, ACO, and how to solve TSP using ACO. In Section 3, we propose a novel niching strategy, and introduce the 2-opt algorithm and circular Jaccard distance to it, thus proposing the CJD-MSO algorithm. Section 4 describes MSTSP benchmarks, and evaluates the CJD-MSO with experiments. Section 5 concludes the paper finally.

2. Related work

In this section, we first introduce TSP and the ACO algorithm. Then we introduce how to solve the TSP using ACO algorithm. Finally, we introduce the basic multi-solution optimization algorithm for MSTSP.

2.1. Traveling salesman problem

Let $G = (V, E)$ be a graph with set of vertexes V and set of edges $E = \{(x, y)|x, y \in V\}$. Each edge $e \in E$ is assigned a cost c_e. If the graph is directed, then the cost of (x, y) is equal to the cost of (y, x) for all $(x, y) \in E$. The equality does not hold for undirected graph.

Without loss of generality, we suppose graph G is a complete graph. Let the set of vertices be $V = \{v_1, v_2, ..., v_n\}$. The cost matrix is given by $C = (c_{ij})_{n\times n}$, where the cost of the edge from v_i to v_j is denoted c_{ij}.

In the context of the traveling salesman problem, the vertexes correspond to cities and the edges correspond to the path between those cities. Let H be the set of all Hamiltonian cycles which visit each vertex exactly once except for the starting point, in G. The traveling salesman problem (TSP) is to find a Hamiltonian cycle $h \in H$ such that the total costs in the cycle is minimized.

The TSP may be formulated as an integer linear programming (ILP) problem [24]. Let

$$x_{ij} = \begin{cases} 1 & \text{the path goes from city } i \text{ to city } j \\ 0 & \text{otherwise.} \end{cases}$$

For $i = 0, 1, \cdots, n$, define u_i as an artificial variable. Then TSP can be rewritten as the following ILP problem:

$$\begin{array}{lcc} \min & \sum_{i=0}^{n} \sum_{j\neq i, j=0}^{n} c_{ij} x_{ij} & \\ & 0 \le x_{ij} \le 1 & i, j = 0, \cdots, n \\ & u_i \in Z & i = 0, \cdots, n \\ & \sum_{i=0, i\neq j}^{n} x_{ij} = 1 & j = 0, \cdots, n \\ & \sum_{j=0, j\neq i}^{n} x_{ij} = 1 & i = 0, \cdots, n \\ & u_i - u_j + n x_{ij} \le n - 1 & 1 \le i \neq j \le n \end{array} \tag{2.1}$$

The first set of equalities requires each city to be arrived from exact one other city. The second set of equalities requires each city to be a departure to exact one other city. The last constraints with the artificial variable u_i ensure that it is a Hamiltonian cycle. The ILP can be relaxed and solved as a linear programming (LP) using the simplex method. Other exact solution methods include the cutting plane method [25] and branch-and-cut [26].

2.2. *Ant colony optimization for traveling salesman problem*

The exact solution methods suffer from high computational complexity. Therefore, some metaheuristics such as ACO and GA were suggested for addressing the TSP. Ant colony optimization algorithm (ACO) is a swarm intelligence optimization algorithm, which is inspired by the foraging behavior of ants [15].

When the ants search for food, they wander around the nest randomly. If some ant finds food, it carries the food back to the nest and lays down pheromones along the road. Other ants nearby can perceive the pheromone and follow the path, and thus increasing the pheromone density. At the same time, the pheromone always evaporates. The more time it takes for an ant to travel along the path, the more time the pheromone evaporates. Hence the pheromone density becomes higher on the shorter path than the longer one. Other ants tend to take the short path and lay down new pheromones. Pheromones attracts more and more ants move on the shorter paths, and finally find the shortest path.

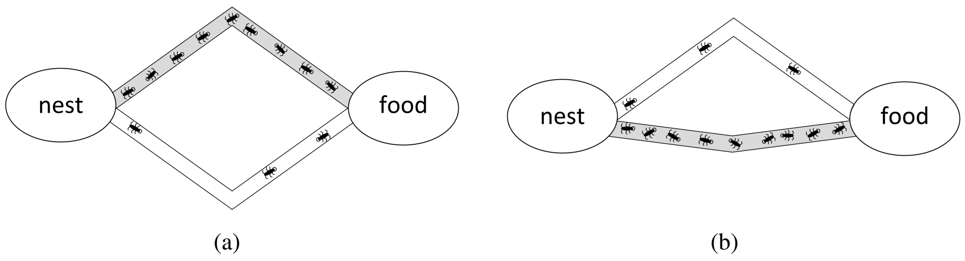


Figure 1. Path choosing behavior of the ant colony.

Figure 1 illustrates two scenarios about how the ants choose path through pheromones. Figure 1(a) contains two paths with the same length, it can be observed that most ants chose the upper path. The main reason is that once the upper path is first taken by some ant, it deposits pheromone on the path, which attracts other ants to choose the upper path. As the number of ants that choose the upper path increase, resulting in a higher pheromone density on the path than the lower one, more other ants tend to choose the upper path.

Figure 1(b) shows that most ants prefer the shorter path. Since the accumulation rate of pheromone on the short path is relatively high, the higher density of pheromone on the short path encourages more ants to choose the short path between the nest and the food.

ACO is a popular metaheuristic for addressing the TSP [27]. Next, we show how to use ACO to solve the TSP. At the beginning, m ants are randomly placed in n cities. The pheromone initial value on all paths between city and city are equal. The ant k ($k = 1, 2, \cdots, m$) calculates the transition probability according to Eq (2.2), before choosing the next city.

$$P_k(i,j) = \begin{cases} \frac{[\tau(i,j)]^{\alpha}[\eta(i,j)]^{\beta}}{\sum\limits_{u\in J_k(i)}[\tau(i,u)]^{\alpha}[\eta(i,u]^{\beta}} & j \in J_k(i) \\ 0 & otherwise, \end{cases} \tag{2.2}$$

where $P_k(i,j)$ is the transition possibility that ant k moving from city i to city j; α is used to control the concentration of pheromone; β is used to control the role of heuristic information; $\tau(i,j)$ is pheromones on edges (i,j); $\eta(i,j)$ is the heuristic factor for moving from city i to city j; $J_k(i)$ is a collection of candidate cities which ant k is allowed to visit next.

The R^k is used to record the vertexes which the ant k has visited. When m ants have completed the tour and go back to the departure vertex, the total cost of the n edges traveled by each ant is calculated. Then, the minimum cost is saved, and the pheromone on each edge is updated. During the iteration, the amount of pheromone remaining on the path after evaporating is calculated as follows:

$$\tau(i,j) = (1-\rho)\cdot\tau(i,j) + \sum_{k=1}^{m}\Delta\tau_k(i,j), \tag{2.3}$$

where ρ, between 0 and 1, is pheromone volatilization coefficient. $\Delta\tau_k(i,j)$ is the pheromones released by the k ant during this tour, which is defined as:

$$\Delta\tau_k(i,j) = \begin{cases}(C_k)^{-1} & (i,j)\in R^k \\ 0 & otherwise\end{cases} \tag{2.4}$$

where C_k represents the path length of the ant k.

Algorithm1 presents the pseudocode.

Algorithm 1 Solving TSP using the ACO algorithm

```
Input: Number of ants $m$, pheromone factor $\alpha$, heuristic function factor $\beta$, pheromone volatilization
    coefficient $\rho$, and maximum number of iterations
Output: An optimal solution
1: Initialize ant colony parameters and pheromone matrix
2: while not meet the stopping criteria do
3:     Randomly place each ant at a different starting point
4:     For each ant $k$ ($k = 1, 2, \cdots, m$), calculate the next vertex to be visited using the transition
       probability formula, until all ants have visited all the vertexes
5:     Calculate the path length $L$ passed by each ant, and save the optimal solution (shortest path)
6:     Update the pheromone concentration on all paths
7: end while
```

2.3. *Multi-Solution optimization*

Most swarm intelligence optimization algorithms are developed to solve the single-solution optimization problems. It is necessary to introduce some strategies to make these algorithms suitable for solving multi-solution optimization problems.

Niching method is one of the most widely used strategies. Niching method is a category of techniques to prevent population convergence to a single optimum by maintaining multiple niches. It should be mentioned that some improved ACO versions and various other metaheuristics may have good mechanism for exploration. Although good exploration ability promotes diversity of the solution, it is preferable to use the niching method for maintaining multiple niches.

The classic niching methods were developed in the early 70s such as crowding method [28]. After that, various niching methods were developed including fitness sharing [29], clearing [30], RTS [31], speciation [32] and clustering [33,34] .

Fitness sharing [29] is inspired by the sharing phenomenon in nature. The resource in a niching is limited, so individuals have to share their resources with other individuals occupying the same niching.

When there are too many individuals in a niche, the fitness of the individual will be reduced, thus leading to reduce the number of individuals. Or, conversely, a small number of individuals in a niche may increase the population size. Fitness is adjusted by the population size in a niche, thus facilitating finding multiple optimal solutions.

Crowding [28] is a technique to maintain diversity during the course of iteration through careful population replacement. First, a number of individuals are selected from the parent generation. The number of individuals are determined by a crowding factor (CF). Then, the new individuals most similar to parent individuals are selected to replace parent individuals. Since the population is initialized with evenly distribution, the evenly distribution can be maintained during the course of iteration. Algorithm 2 depicts the procedure of crowding.

Algorithm 2 Crowding

```
Input: Population $NP$
Output: A set of optimal solutions
1: Randomly generate individuals of $NP$ and a set of $CF$ values
2: while not meet the stopping criteria do
3:     Calculate the fitness of each individual
4:     Generate candidate individuals through the basic operation of the algorithm
5:     Randomly select $CF$ individuals from the parents
6:     Compare the candidate individual with the closest individual among $CF$ individuals, and
       replace the closest individual if the candidate individual has better fitness
7: end while
```

Clearing [30] is a strategy easy to implement. In each iteration, the current optimal individual in the population is selected as the center. Then, all the individuals within the specified clearance radius are removed from the population. The procedure repeats until the original population has no individual, thus obtaining a set of optimal solutions finally.

The principle of speciation is to divide the population into several niches, and the populations in each niche evolve parallelly. Algorithm 3 presents the pseudocode of speciation.

Algorithm 3 Speciation

```
Input: Population and niching radius $\sigma$
Output: A set of optimal solutions
1: Initialize population
2: Calculate the fitness of the population and sort them according to the fitness value
3: while population is not empty do
4:     Find the best individual $seed$ in the population
5:     Find all individuals whose distance to $seed$ is less than $\sigma$, and form a subpopulation with $seed$
6:     Remove the subpopulation from the population
7: end while
```

The goal of clustering [33] is to group individuals into clusters such that individuals in each cluster have greater similarity than inter-clusters. K-means algorithm is a typical clustering algorithm used in niching method, where K represents the number of clusters [34]. Algorithm 4 is the pseudocode of K-means algorithm for niching.

Algorithm 4 K-means algorithm for niching

```
Input: Population $NP$ and clustering number $K$
Output: $K$ species
1: Initialize population
2: Randomly select $K$ individuals from the population as the centers
3: while not meet the stopping criteria do
4:     Calculate the distance between other individuals in the population and these $K$ centers
5:     Assign other individuals to the nearest cluster according to distance
6:     Recalculate and update the centers in the population
7: end while
```

2.4. *Similarity metrics*

The similarity metric plays an essential role in multi-solution optimization. As shown in Algorithm 5, whether the new individual is discarded or not is up to the similarity between the new path and the old one. Next, we introduce couples of similarity metrics which may be used in the multi-solution optimization.

2.4.1. Pearson correlation coefficient

Pearson correlation coefficient (PCC) [35], or the bivariate correlation, is a widely used measure of the similarity between vectors. If there are two variables: X and Y, PCC can be formulated as follows:

$$\rho(X,Y) = \frac{E[(X-\mu_X)(Y-\mu_Y)]}{\sqrt{\sum_{i=1}^{n}(X_i-\mu_X)^2}\sqrt{\sum_{i=1}^{n}(Y_i-\mu_Y)^2}} \tag{2.5}$$

where E is the mathematical expectation; μ_X is the expectation of the n-dimensional vector X; μ_Y is the expectation of Y.

The greater the absolute value of the correlation coefficient is, the stronger correlation they have. The correlation with PCC 0 is the weakest correlation.

2.4.2. Distance correlation coefficient

Distance correlation coefficient (DCC) [36] , or distance covariance, is a measure of dependence between two arbitrary vectors, not necessarily equal in dimension. The DCC is zero if and only if the vectors are independent. The formula for DCC is as follows:

$$\rho_{XY} = \frac{E[(X-E(X))(Y-E(Y))]}{\sqrt{D(X)}\sqrt{D(Y)}}, \tag{2.6}$$

where E is the mathematical expectation, D is the variance. The similarity between X and Y is closer if the DCC is greater.

Distance correlation (DC) defined in [36] is the complement set of DCC. It can be formulated as follows:

$$D_{XY} = 1 - \rho_{XY} \tag{2.7}$$

Therefore, the similarity between X and Y is closer if their DC is smaller.

2.4.3. Common edge similarity

Two vectors with more common edges should have stronger similarity. Suppose P_1 and P_2 represent two vectors, the common edge similarity(CES) is defined as follows:

$$similarity(P_1, P_2) = \frac{shared\ edges}{number\ of\ cities} \tag{2.8}$$

The CES similarity is between 0 and 1. Only when the similarity equals to 1, does it mean that the two vectors are identical.

2.4.4. Jaccard distance

The Jaccard index [37], or the Jaccard similarity coefficient, is widely used for comparing the similarity or difference between limited sample sets. Given two sets A and B, the Jaccard coefficient is defined as the ratio of the intersection to the union of A and B, namely,

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| + |A \cap B|}. \tag{2.9}$$

The Jaccard index has value between 0 and 1. Specifically, when both A and B are empty sets, their Jaccard index is defined as 1.

Jaccard distance [38] is also a metric used to measure the difference between two sets. It is the complement index of Jaccard similarity coefficient. The formula of the Jaccard distance is as follows:

$$d_j(A + B) = 1 - J(A, B) = \frac{A \Delta B}{|A \cup B|}, \tag{2.10}$$

where the symmetric difference $A \Delta B = |A \cup B| - |A \cap B|$.

When the Jaccard distance is used to measure the similarity, the result of the completely identical calculation equals to 0, and the result of the completely different calculation equals to 1.

3. Circular Jaccard distance based multi-solution ant colony optimization

In this section, we first propose a novel niching strategy for multi-solution optimization. Next, we propose a circular Jaccard distance as the similarity metric which plays an important role in our algorithm. Then, we introduce the 2-opt strategy which is used as the local search method. Finally, we summarize the whole multi-solution optimization algorithm.

3.1. Distancing

We develop a novel niching method based on the “radius” strategy which is used in the clearing strategy. After initializing a population of path randomly, we select a set of paths from the population with a given ratio, which is regarded as the parent population. In each iteration, a batch of new individual is generated following the rule of the base metaheuristics. For each new individual, if it has a high similarity to the previous one, then the new individual will be discarded. Otherwise, the new individual is saved in the population pool as the offspring candidate. We calculate the fitness of the parent population together with the offspring population, and sort them in ascending order. Then, a number of good individuals are chosen from the queue head to form a new parent population. The strategy requires that the individual in the next generation keeps a certain distance from the current generation. As shown in Figure 2, let 1, 2 and 3 (or 3' or 3") be the possible positions of an individual in the 1st, 2nd and 3rd generation. 2 is a step from 1. And then 3 (or 3' or 3") is a step from 2 $\cdots\cdots$. With high probability, the distance from 3 (or 3' or 3") to 1 is greater than the distance between 2 to 1. Hence it maintains diversity during the course of search.

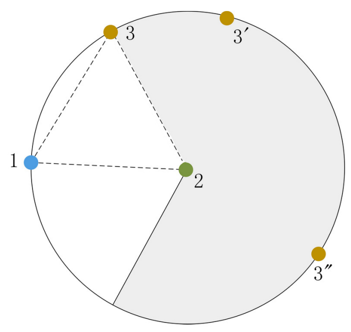


Figure 2. distancing strategy.

The procedure repeats until the stopping criteria are met and the optimal set of individuals is obtained finally. The proposed niching method is different from any other niching method proposed previously, such as crowding, fitness sharing, and clearing. And it is easy to implement compared with clearing. We name it as "distancing".

Algorithm 5 The multi-solution optimization algorithm with distancing

```
Input: The number of ants, iteration bound, $\alpha$, $\beta$, and $\rho$
Output: A set of optimal solutions
1: Initialize population and pheromone matrix
2: Construct a path through pseudo-random selection rules to obtain the first generation parent population
3: while not meet the stopping criteria do
4:     Discard new individual that have a high similarity to the previous one, and form a new offspring population from the remaining individuals
5:     Update the pheromone matrix
6:     Calculate the fitness of the parent population together with the offspring population, and sort them
7:     Select a set of good individuals from the queue head to form a new parent population
8: end while
```

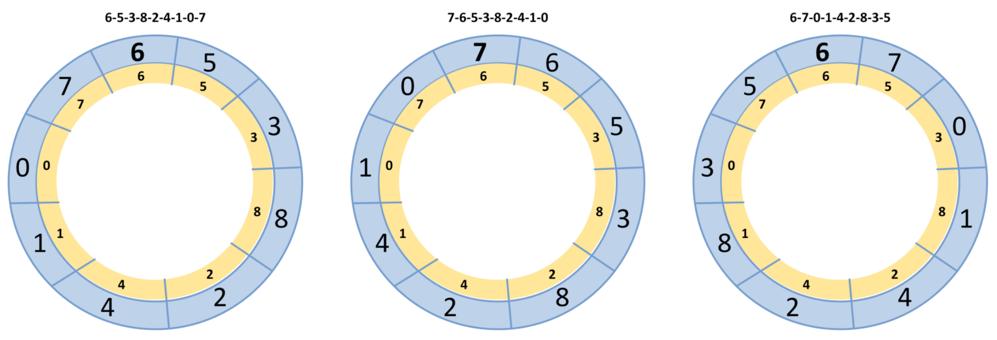
Algorithm 5 gives the pseudocode of the distancing method we proposed. In Algorithm 5, ACO is the base metaheuristics.

3.2. Circular Jaccard distance

The similarity metrics mentioned previously, such as Jaccard distance, are used to promote diversity of the population during the course of iteration, which is also the key measure of quality of the multiple solutions. The valid path in the TSP is always a Hamiltonian cycle. If we use the similarity metrics for vectors in MSTSP, then the property of cycle is not explored and utilized fully.

We take an example to reveal the limitation of Jaccard distance. Suppose there are three paths: “0-6-5-3-7-8-2-4-1” (path 1), “0-1-4-2-8-7-3-5-6” (path 2) and “2-8-7-3-5-6-0-1-4” (path 3). The Jaccard distance between path 1 and path 2 equals to 1. The Jaccard distance between path 2 and path 3 also equals to 1. The Jaccard distance between path 1 and path 3 also equals to 0.889. However, the path 1, 2 and 3 are all the same for the TSP. Therefore, we propose a novel metric named Circular Jaccard Distance (CJD) in this paper to overcome the limitation.

The CJD respects the invariance of the path, which means the paths transformed by cycling or flipping are regarded the same as the original path. When calculating the CJD, first, one of the two paths is transformed by cycling and flipping. Then the Jaccard distance between each pairs are calculated. Finally, the CJD is obtained by minimizing these JDs.



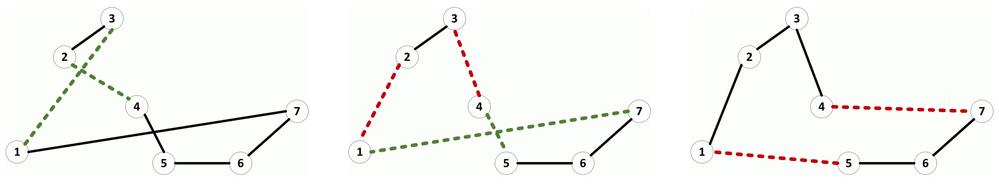
(a) original path (b) rotated path (c) flipped path

Figure 3. Transformations before calculating the circular Jaccard distance.

For example, suppose the original path is “0-1-4-2-8-7-3-5-6”. The new generated path is the path “6-5-3-8-2-4-1-0-7”. By transforming the second path, one can obtain “6-5-3-8-2-4-1-0-7”, “5-3-8-2-4-1-0-7-6”, “3-8-2-4-1-0-7-6-5”, “8-2-4-1-0-7-6-5-3”, “2-4-1-0-7-6-5-3-8”,“4-1-0-7-6-5-3-8-2”, “1-0-7-6-5-3-8-2-4”, “0-7-6-5-3-8-2-4-1”, “7-6-5-3-8-2-4-1-0”, “6-7-0-1-4-2-8-3-5”, “5-6-7-0-1-4-2-8-3”, “3-5-6-7-0-1-4-2-8”, “8-3-5-6-7-0-1-4-2”, ”2-8-3-5-6-7-0-1-4”, “4-2-8-3-5-6-7-0-1”, “1-4-2-8-3-5-6-7-0”, “0-1-4-2-8-3-5-6-7” and “7-0-1-4-2-8-3-5-6”. As shown in Figure 3, Figure 3(a) is the original path. Figure 3(b) is the path obtained by clockwise or counter-clockwise rotation. Figure 3(c) is the path obtained by flipping. The CJDs between the converted paths and the original path are 1, 0.78, 1, 0.67, 1, 0.78, 1, 1, 0.89, 1, 1, 1, 1, 0.89, 1, 1, 0.5 and 0.67 respectively.

3.3. 2-opt strategy

In optimization, n-opt is a simple local search method for solving combinatorial optimization problems [39]. The n-opt treatment involves deleting n edges in a tour, reconstructing the path in some other possible ways.



(a) initial solution (b) after one 2-opt operation (c) after two 2-opt operations

Figure 4. The 2-opt strategy. The green dotted edges are replaced by the red dash-dotted edges.

Take the 2-opt as example, as shown in Figure 4, Figure 4(a) is the original path. In every 2-opt strategy, two edges of the solution are deleted and two new edges are added to obtain a new path. If the new path is better, the original path is replaced. Specifically, since $d(V_1, V_3) + d(V_2, V_4) > d(V_1, V_2) + d(V_3, V_4)$, the edges (V_1, V_3) and (V_2, V_4) are removed, and the edges (V_1, V_2) and (V_3, V_4) are added instead, thus forming a new path, see Figure4(a). Here d represents the distance or similarity. Again, in Figure 4(c), since $d(V_1, V_7)+d(V_4, V_5) > d(V_1, V_5)+d(V_4, V_7)$, the edges (V_1, V_7) and (V_4, V_5) are removed, and the edges (V_1, V_5) and (V_4, V_7) are added instead, see Figure 4(b).

The 3-opt strategy is able to create more choices of new paths. However, the computational complexity becomes $O(n^3)$ which is not affordable for most applications. Therefore, we use 2-opt as the main local search method in this paper.

3.4. Circular Jaccard distance based multi-solution optimization algorithm based on ant colony optimization

In this paper, we propose a circular Jaccard distance based multi-solution optimization (CJD-MSO) algorithm to improve the performance for solving the traveling salesman problem.

First, we initialize pheromone matrix, randomly place the ants to the initial cities, calculate the next city according to the transition probabilities, and thus obtain the first generation parent population. In the iterative process, the 2-opt algorithm is used to improve the quality of the solution and the distancing strategy is used to maintain diversity. The circular Jaccard distance between each new individual with the last individual path is calculated. If the similarity value is higher than a preset value, which means that two paths are significantly different, then it is retained. Otherwise, the new individual is discarded. The parent population and offspring population are mixed together, and a set of good individuals with higher fitness is chosen to be the next generation of parent population. Algo.6 shows the entire process of CJD-MSO.

Algorithm 6 The CJD-MSO algorithm

```
Input: The number of ants $m$, iteration bound, and the parameters of CJD-MSO ($\alpha, \beta$ and $\rho$)
Output: A set of optimal solutions
1: Initialize population and pheromone matrix
2: Randomly place each ant at a different starting point
3: For each ant $k(k = 1, 2, \ldots, m)$, calculate the next city to be visited according to the transition
   formula obtained by the roulette method, until all ants have visited all the cities
4: Obtain the first generation parent population through the 2-opt strategy
5: while not meet the stopping criteria do
6:     Randomly place each ant at a different starting point
7:     For each ant $k(k = 1, 2, \ldots, m)$, calculate the next city to be visited according to the transition
   probability formula obtained by the roulette method, until all ants have visited all the cities
8:     For each new population, 2-opt algorithm is used to improve the quality of solutions
9:     Calculating the circular Jaccard distance between the new individual path and latest individual
   path
10:    if the similarity is higher than a preset value then
11:        Save the new individual path
12:    else
13:        Discard the new individual path
14:    end if
15:    Obtain the offspring population, and update the pheromone matrix
16:    Mix the parent population with the offspring population
17:    Sort them and select some good individuals from the queue to form the next parent population
18:    Save the optimal paths obtained so far
19: end while
```

4. Experiments

In this section, first we describe the benchmark TSPs on which we shall test our proposed algorithm. Next, we introduce the evaluation criteria we shall use. Then, we exhibit our experimental results including the performance of CJD-MSO and some comparison details.

4.1. Benchmark TSPs

Ting Huang [21] proposed a benchmark set for MSTSPs in 2018, which contains 25 MSTSP instances, as shown in Table 1. In the benchmark set, MSTSPs are classified into three categories: simple MSTSPs (from MSTSP1 to MSTSP6), geometry MSTSPs (from MSTSP7 to MSTSP12), and composite MSTSPs (from MSTSP13 to MSTSP25). The number of cities in the MSTSPs ranges from 9 to 66. And the number of optimal scales ranges from 2 to 196. We test our algorithm on the benchmark set.

Table 1. The benchmark MSTSPs.

Name	City nums	Optimal solutions nums	Shortest path length	Name	City nums	Optimal solutions nums	Shortest path length
MSTSP1	9	3	680	MSTSP14	34	16	3575
MSTSP2	10	4	1265	MSTSP15	22	72	9455
MSTSP3	10	13	832	MSTSP16	33	64	8761
MSTSP4	11	4	803	MSTSP17	35	10	9061
MSTSP5	12	2	754	MSTSP18	39	20	23763
MSTSP6	12	4	845	MSTSP19	42	20	14408
MSTSP7	10	56	130	MSTSP20	45	20	10973
MSTSP8	12	110	1344	MSTSP21	48	4	6767
MSTSP9	10	4	72	MSTSP22	55	9	10442
MSTSP10	10	4	72	MSTSP23	59	10	24451
MSTSP11	10	14	78	MSTSP24	60	36	9614
MSTSP12	15	196	130	MSTSP25	66	26	9521
MSTSP13	28	70	3055				

4.2. Evaluation criteria

We use two evaluation criteria to test our work: F_β and the diversity indicator (DI) [21].

F_β is an indicator calculated from the precision value P and the recall value R. The formula is as follows:

$$F_\beta = \frac{(1+\beta)^2 \cdot P \cdot R}{\beta^2 \cdot P + R}, \tag{4.1}$$

where P represents the proportion of optimal solutions, and R represents the proportion of the found optimal solutions to known optimal solutions. When β is set to 1, P and R have the same importance. We set β as 0.3 to attach more importance to the precision.

The formula of P and R are as follows:

$$P = \frac{TP}{TP + FP}, \tag{4.2}$$

$$R = \frac{TP}{TP + FN}. \tag{4.3}$$

The true positive, TP, represents the number of optimal solutions obtained by the algorithm; The false positive, FP, represents the number of non-optimal solutions obtained by the algorithm; The false negative, FN, represents the optimal solutions that the algorithm did not find.

The P, R, and F_β are real values between 0 and 1. If the solutions obtained by the algorithm are all optimal solutions, then P equals to 1. If the algorithm finds all known optimal solutions, then R equals to 1. F_β is 1, if the values of P and R both are 1. Larger F_β means better performance.

The DI helps to assess the diversity performance of the multiple solutions. DI is defined by the average maximum similarity between the obtained solutions and the ground-truth solutions. The formula of DI is as follows:

$$DI(P, S) = \frac{\sum_{i=1}^{|P|} max_{j=1,\dots,|S|} S(P_i, S_j)}{|P|}, \tag{4.4}$$

where P represents the known optimal solution set; S represents the solution set obtained by the algorithm; P_i represents the i-th solution in the set P; $S_j$represents the j-th solution in the set S; and $S(P_i, S_j)$ represents the similarity. Larger DI means that more solutions have been found, given the known optimal solutions.

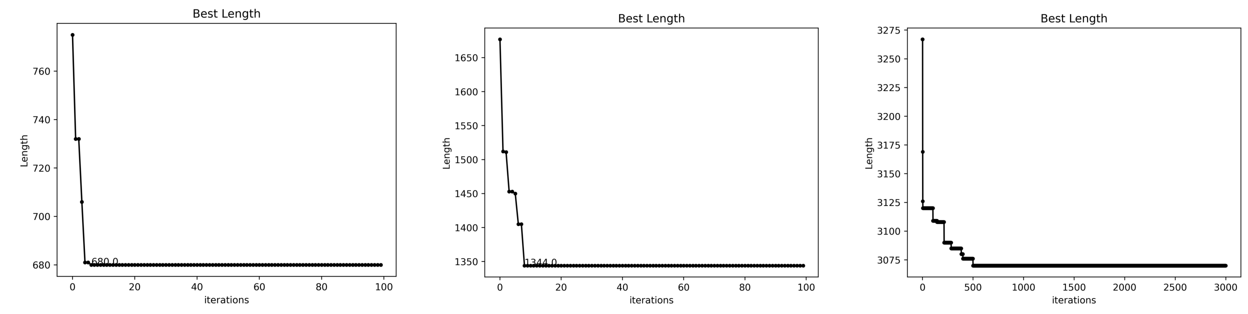
4.3. Experimental results

In this section, we conduct a series of numerical experiments to evaluate the proposed algorithm. Firstly, we test the performance of ACO in finding single optimal solution for the MSTSP. Secondly, we compare the algorithmic performance with various similarity metrics including PCC, DC, CES and JD. Thirdly, we compare the algorithmic performance with circular Jaccard distance to that with JD. Finally, we compare CJD-MSO with NACS [23], NGA [21], NMA [22], and give the values of F_β and DI.

The parameter settings are as follows: $\alpha = 1.0, \beta = 2.0, \rho = 0.5$. The threshold value of the CJD is set to 0.4. All algorithms are implemented in Python language and run on the Mac OS Catalina system.

We use the MSTSP instances to test the ACO algorithm. Although the MSTSP instances are designed for testing multi-solution optimization algorithms, it is also eligible for testing the single solution optimization algorithms.

Figure 5 presents the convergence curves of ACO. For simple and geometry MSTSPs, the ACO converges to the optimal solution rapidly. The ACO without niching method can find only one optimum in a single run. For some composite MSTSPs such as Figure 5(c), the ACO cannot find any optimum within 3000 iterations, which indicates that composite MSTSPs are really hard problems.



(a) MSTSP1 with the optimum 680 (b) MSTSP8 with the optimum 1344 (c) MSTSP13 with the optimum 3055

Figure 5. The convergence curves.

Next, we improve the original ACO by adding "distancing" strategy. This time, we find couples of different solutions for MSTSPs including the simple MSTSPs, the geometry MSTSPs, and the composite MSTSPs, as shown in Figure 6.

We compare the performance of F_β and DI on MSTSP among various similarity metrics including PCC, DC, CES and JD.

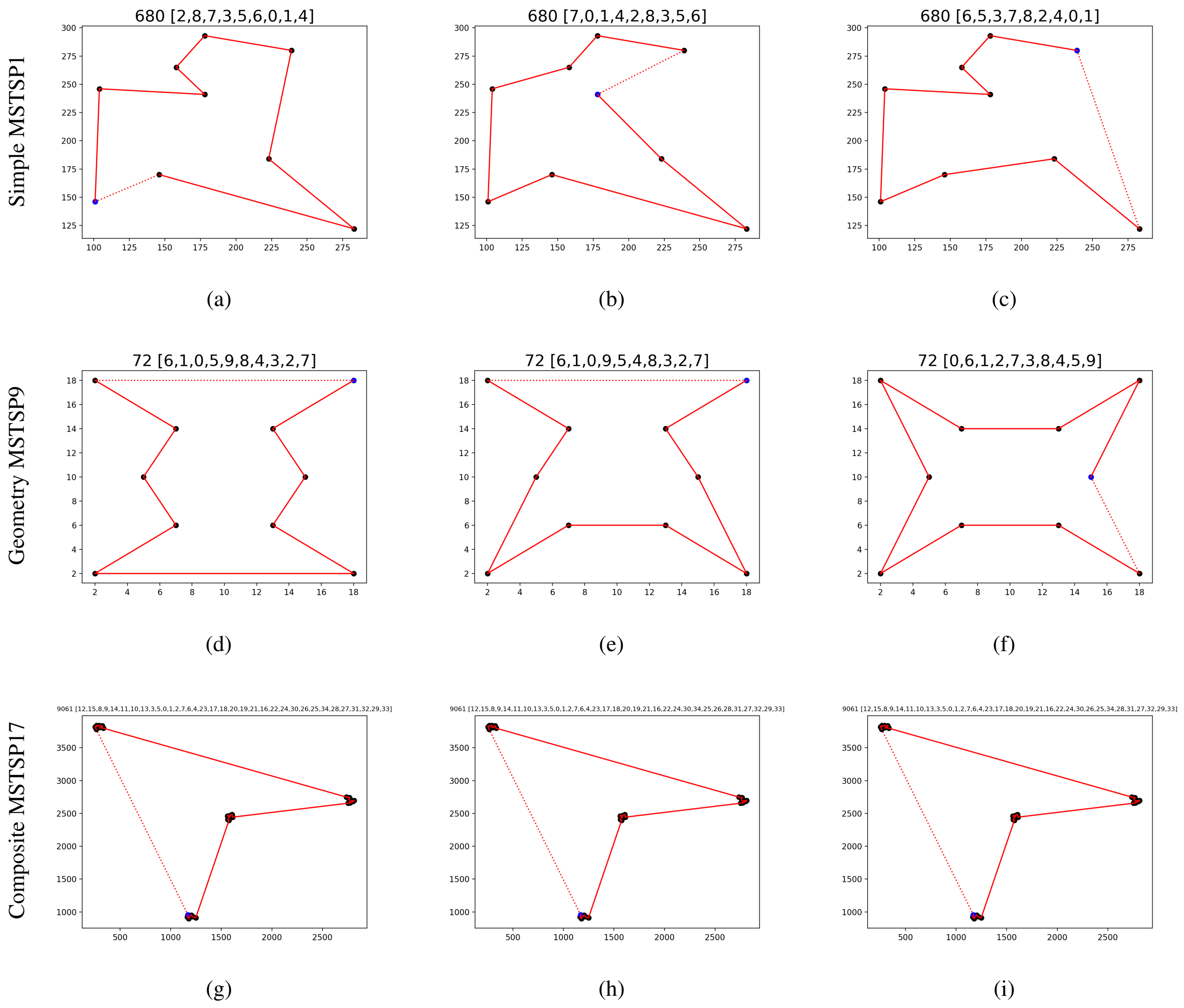


Figure 6. The real solutions found for various MSTSPs.

As shown in Figure 7 and Table 2, the four similarity metrics work well in the simple MSTSPs. For geometry MSTSPs and composite MSTSPs, JD considerably outperforms the other three methods.

For MSTSPs with more optimal solutions or more cities, such as MSTSP18 and MSTSP22, since the F_βs almost drop to 0 for all similarity metrics. We compare their DIs in Figure 8.

From Figure 8, one can observe that JD considerably outperforms PCC and DC, and is slightly better than CES. So, we believe that JD is a good similarity metric for solving the MSTSPs. Next, we compare the performance of CJD with that of JD. Table 3 gives F_β value of the algorithms with CJD and CD.

Table 2. The F_β values of PCC, DC, CES and JD. (BRPS is the total of bold items.)

NAME	PCC	DC	CES	JD	NAME	PCC	DC	CES	JD
MSTSP1	**1.000**	**1.000**	**1.000**	**1.000**	MSTSP14	0.813	**0.905**	0.766	**0.905**
MSTSP2	**1.000**	**1.000**	**1.000**	**1.000**	MSTSP15	0.684	**0.804**	0.679	0.795
MSTSP3	0.835	0.835	**0.993**	0.835	MSTSP16	0.629	0.748	0.586	**0.760**
MSTSP4	**1.000**	**1.000**	**1.000**	**1.000**	MSTSP17	0.325	**0.650**	0.382	**0.650**
MSTSP5	**1.000**	**1.000**	0.907	**1.000**	MSTSP18	0.000	0.000	0.224	0.000
MSTSP6	**1.000**	**1.000**	**1.000**	**1.000**	MSTSP19	0.000	**0.700**	0.237	**0.700**
MSTSP7	0.751	0.690	**0.879**	0.690	MSTSP20	0.160	0.130	0.172	**0.520**
MSTSP8	0.619	0.520	**0.801**	0.520	MSTSP21	**0.813**	**0.813**	0.120	**0.813**
MSTSP9	**1.000**	**1.000**	**1.000**	**1.000**	MSTSP22	0.000	0.000	0.000	0.000
MSTSP10	0.929	**1.000**	**1.000**	0.591	MSTSP23	0.000	**0.333**	0.000	0.000
MSTSP11	0.941	0.982	0.967	**0.983**	MSTSP24	0.188	0.207	0.118	**0.313**
MSTSP12	0.291	0.292	**0.671**	0.315	MSTSP25	0.000	**0.069**	0.000	0.000
MSTSP13	0.743	0.775	0.743	**0.950**	**BRPS**	**7**	**14**	**9**	**15**

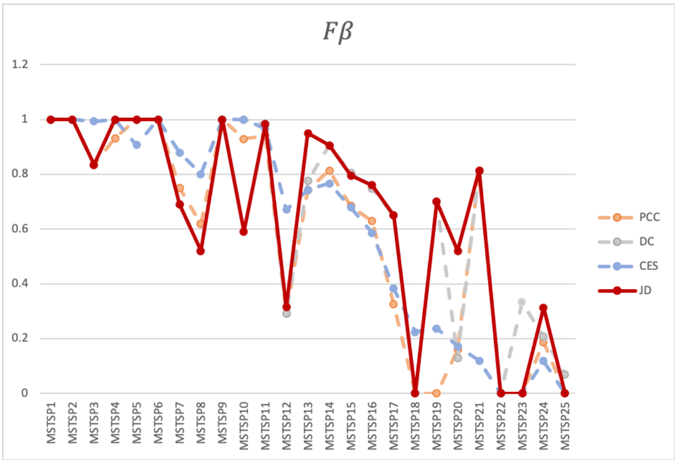


Figure 7. Comparison of F_β for PCC, DC, CES, and JD.

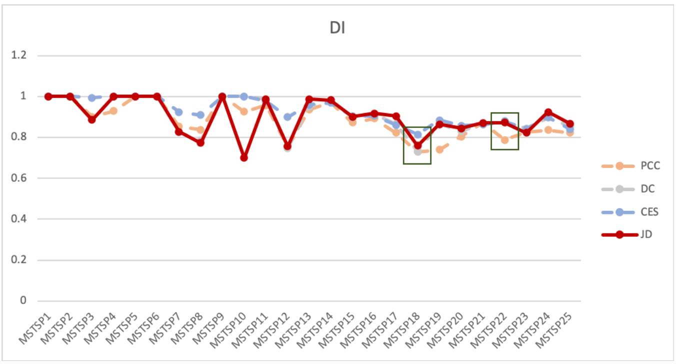


Figure 8. Comparison of *DI* for PCC, DC, CES, and JD.

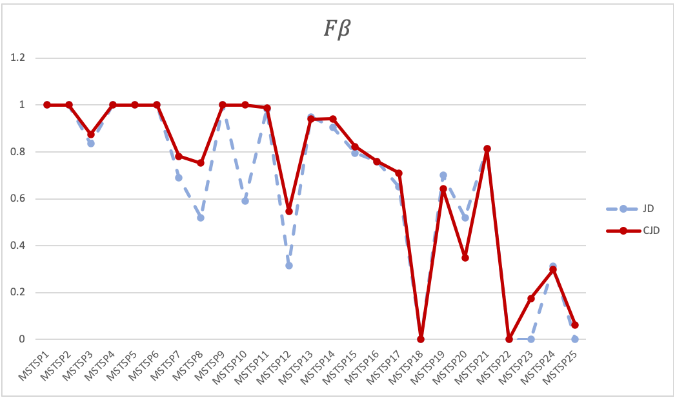


Figure 9. Comparison of F_β (CJD vs JD).

Table 3. The F_β values of (CJD and JD). The BRPS represents the total of bold items.

NAME	JD	CJD	NAME	JD	CJD
MSTSP1	**1.000**	**1.000**	MSTSP14	0.905	**0.942**
MSTSP2	**1.000**	**1.000**	MSTSP15	0.795	**0.823**
MSTSP3	0.835	**0.875**	MSTSP16	**0.760**	**0.760**
MSTSP4	**1.000**	**1.000**	MSTSP17	0.650	**0.710**
MSTSP5	**1.000**	**1.000**	MSTSP18	0.000	0.000
MSTSP6	**1.000**	**1.000**	MSTSP19	**0.700**	0.642
MSTSP7	0.690	**0.782**	MSTSP20	**0.520**	0.349
MSTSP8	0.520	**0.752**	MSTSP21	**0.813**	**0.813**
MSTSP9	**1.000**	**1.000**	MSTSP22	0.000	0.000
MSTSP10	0.591	**1.000**	MSTSP23	0.000	**0.175**
MSTSP11	0.983	**0.988**	MSTSP24	**0.313**	0.299
MSTSP12	0.315	**0.547**	MSTSP25	0.000	**0.062**
MSTSP13	0.950	**0.940**	**BRPS**	**12**	**19**

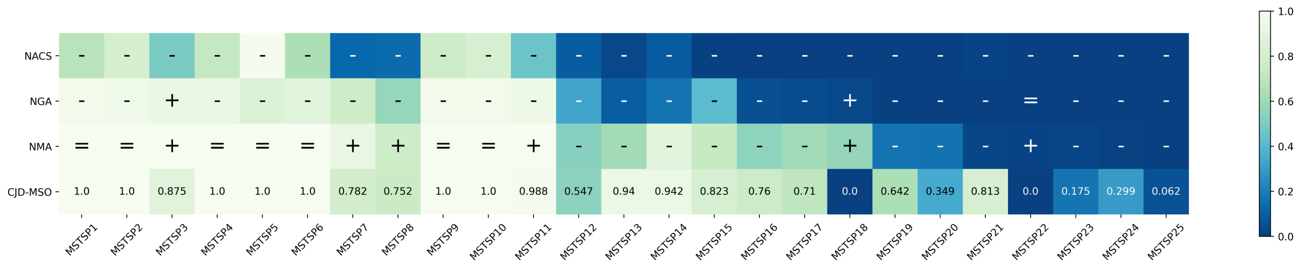


Figure 10. Pseudo-color plot of 25 MSTSPs in terms of F_β with three compared algorithms and the proposed algorithm. Taking F_β of CJD-MSO as the base algorithm, the algorithm worse than CJD-MSO is marked as "-", the algorithm better than CJD-MSO is marked as "+", and the algorithm has the same performance is marked as "=".

From Figure 9 and Table 3, we can observe that CJD significantly outperforms JD, indicating that CJD facilitates maintaining the diversity of the population.

Finally, we perform experiment to compare CJD-MSO with NGA [21], NMA [22] and NACS [23]. Table 4 compares the F_β value of these algorithms. The best F_β for each MSTSP instance is marked in bold. BRPS represents the total of bold items.

Table 4. The F_β values of the algorithm in 25 instances.

NAME	NACS	NGA	NMA	CJD-MSO	NAME	NACS	NGA	NMA	CJD-MSO
MSTSP1	0.684	0.973	**1.000**	**1.000**	MSTSP14	0.087	0.172	0.883	**0.942**
MSTSP2	0.804	0.959	**1.000**	**1.000**	MSTSP15	0.004	0.416	0.732	**0.823**
MSTSP3	0.497	0.936	**1.000**	0.875	MSTSP16	0.000	0.054	0.554	**0.760**
MSTSP4	0.724	0.932	**1.000**	**1.000**	MSTSP17	0.000	0.044	0.605	**0.710**
MSTSP5	0.989	0.846	**1.000**	**1.000**	MSTSP18	0.000	0.031	**0.571**	0.000
MSTSP6	0.643	0.877	**1.000**	**1.000**	MSTSP19	0.000	0.007	0.168	**0.642**
MSTSP7	0.125	0.769	**0.923**	0.782	MSTSP20	0.000	0.000	0.165	**0.349**
MSTSP8	0.137	0.578	**0.772**	0.752	MSTSP21	0.012	0.000	0.023	**0.813**
MSTSP9	0.768	0.974	**1.000**	**1.000**	MSTSP22	0.000	0.000	**0.013**	0.000
MSTSP10	0.813	0.969	**1.000**	**1.000**	MSTSP23	0.000	0.000	0.016	**0.175**
MSTSP11	0.459	0.949	**1.000**	0.988	MSTSP24	0.000	0.000	0.010	**0.299**
MSTSP12	0.090	0.331	0.535	**0.547**	MSTSP25	0.000	0.000	0.002	**0.062**
MSTSP13	0.025	0.096	0.611	**0.940**	**BRPS**	0	0	**13**	**19**

We use the pseudo-color plot (Figure 10) to exhibit the comparison results. Only eight "+" exists, which indicates that CJD-MSO perform very well compared with other multi-solution optimization algorithms.

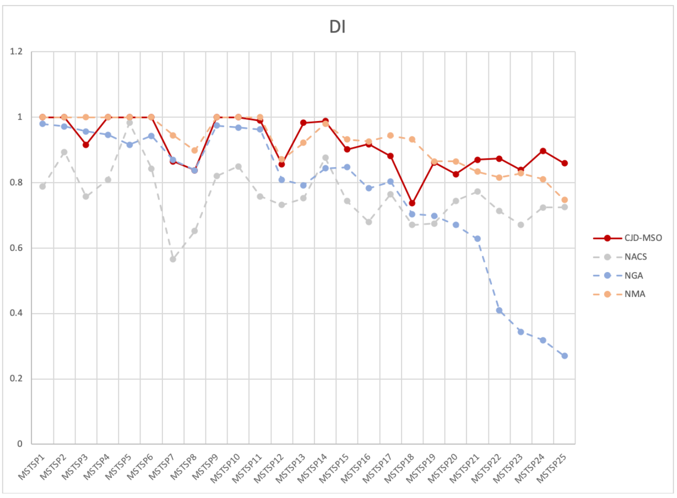


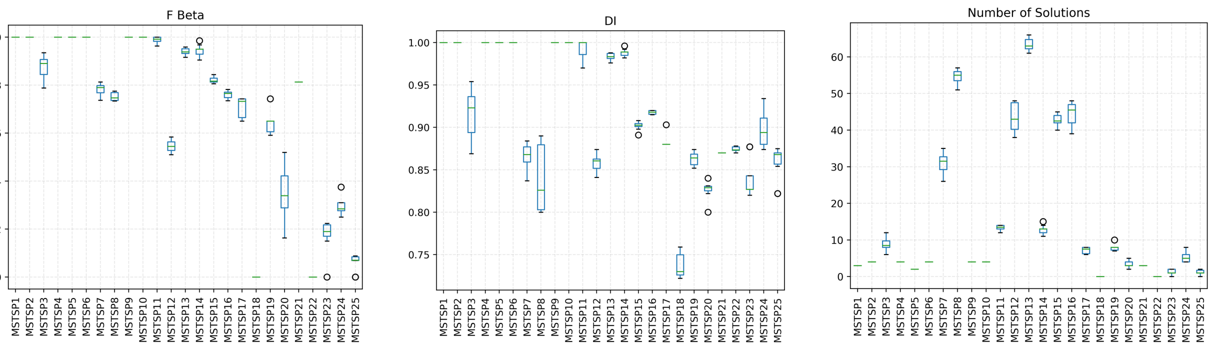
Figure 11. Comparison of *DI* among the algorithms in 25 instances.

Table 5. The *DI* values of the algorithms in 25 instances.

NAME	NACS	NGA	NMA	CJD-MSO	NAME	NACS	NGA	NMA	CJD-MSO
MSTSP1	0.788	0.980	**1.000**	**1.000**	MSTSP14	0.876	0.844	0.981	**0.988**
MSTSP2	0.894	0.972	**1.000**	**1.000**	MSTSP15	0.744	0.847	**0.932**	0.902
MSTSP3	0.757	0.957	**1.000**	0.916	MSTSP16	0.680	0.783	**0.926**	0.918
MSTSP4	0.809	0.947	**1.000**	**1.000**	MSTSP17	0.765	0.803	**0.944**	0.881
MSTSP5	0.983	0.916	**1.000**	**1.000**	MSTSP18	0.671	0.704	**0.932**	0.737
MSTSP6	0.843	0.943	**1.000**	**1.000**	MSTSP19	0.675	0.699	**0.865**	0.862
MSTSP7	0.566	0.869	**0.945**	0.865	MSTSP20	0.745	0.671	**0.865**	0.826
MSTSP8	0.652	0.838	**0.898**	0.838	MSTSP21	0.773	0.628	0.834	**0.870**
MSTSP9	0.820	0.975	**1.000**	**1.000**	MSTSP22	0.713	0.409	0.816	**0.874**
MSTSP10	0.850	0.969	**1.000**	**1.000**	MSTSP23	0.671	0.344	0.829	**0.839**
MSTSP11	0.758	0.963	**1.000**	0.990	MSTSP24	0.724	0.319	0.811	**0.897**
MSTSP12	0.732	0.809	**0.871**	0.857	MSTSP25	0.725	0.270	0.747	**0.859**
MSTSP13	0.752	0.792	0.922	**0.983**					

We calculate the *DI* of CJD-MSO, and compare with other algorithms in Table 5 and Figure 11. For simple MSTSPs and geometry MSTSPs, the DIs of CJD-MSO are stably good. Seven MSTSPs have a DI value of 1, and the left MSTSPs are approximately 1. From MSTSP19 to MSTSP25, CJD-MSO significantly outperforms other algorithms too.

The results of the proposed method is based on the repeated experiments. Figure 12 gives the box plots of F_β, DI, and number of found solutions on 25 MSTSPs.



(a) F_β (b) DI (c) Number of Solutions

Figure 12. The box plots of F_β, DI, and number of multiple solutions on 25 MSTSPs.

5. Conclusions

In this article, we proposed a new multi-solution optimization algorithm named "CJD-MSO" to address the MSTSPs. In the algorithm, we proposed a novel niching technique named "distancing" method, together with a novel similarity metric named "circular Jaccard distance", to maintain population diversity during the course of searching multiple optimal solutions. The "distancing" strategy, different from all niching method emerge so far, were easy to implement. And " circular Jaccard distance" played an essential role in improving the performance of multi-solution optimization.

We also employed basic ACO as the base metaheuristic and employed the 2-opt strategy as the local search method in CJD-MSO. We laid greater emphasis on the strategy of niching and metric definition than compare many and various classic metaheuristics or their improved versions. Experimental results in 25 MSTSP instances showed that CJD-MSO significantly outperforms other multi-solution optimization algorithms, in terms of F_β and DI.

We believe that the " circular Jaccard distance" metric and the "distancing" strategy can be applied to other base metaheuristics for addressing some other combinatorial optimization problems.

Acknowledgements

This work was supported by Ministry of Education in China (MOE) under project grants of Humanities and Social Sciences (No.21YJAZH040).

Conflict of interest

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. M. Dorigo, Traveling salesman problem, in *IEEE International Conference Evolutionary Computation*, (2013).
2. R. M. Al-Khatib, K. M. O. Nahar, SRT-GA: smart real-time system using a powerful genetic algorithm for school bus routing problem, in *2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS)*, (2017), 1–8. https://doi.org/10.1109/IT-DREPS.2017.8277816
3. S. Ma, Y. Hao, Research on the order picking optimization problem of the automated warehouse, in *2009 Chinese Control and Decision Conference*, (2009), 990–993. https://doi.org/10.1109/CCDC.2009.5192816
4. T. Ghorpade, C. G. Corlu, Selective pick-up and delivery problem: a simheuristic approach, in *2020 Winter Simulation Conference (WSC)*, (2020), 1468–1479. https://doi.org/10.1109/WSC48552.2020.9384060
5. P. Kalita, B. Kalita, SCS and TSP in DNA sequencing, *Int. J. Management It Eng.*, **3** (2013), 263–277.
6. M. Cygan, L. Kowalik, A. Socala, Improving TSP tours using dynamic programming over tree decomposition, *ACM Trans. Algorithms*, **15** (2019), 1–19.
7. J. L. An, J. Gao, J. H. Lei, G. H. Gao, An improved algorithm for TSP problem solving with Hopfield neural networks, *Adv. Mater. Res.*, **143** (2011), 538–542. https://doi.org/10.4028/www.scientific.net/AMR.143-144.538

8. T. Liu, H. Zhang, Y. Gao, Solving TSP via fuzzy dynamic PSO and HNN algorithm, in *2012 7th International Conference on Computer Science Education (ICCSE)*, (2012), 105–109. https://doi.org/10.1109/ICCSE.2012.6295036

9. H. Li, X. Liu, Z. Huang, C. Zeng, P. Zou, Z. Chu, et al., Newly emerging nature-inspired optimization - algorithm review, unified framework, evaluation, and behavioral parameter optimization, *IEEE Access*, **8** (2020), 72620–72649.

10. H. Li, Z. Huang, X. Liu, C. Zeng, P. Zou, Multi-fidelity Meta-optimization for nature inspired optimization algorithms, *Appl. Soft Comput.*, **96** (2020), 106619. https://doi.org/10.1016/j.asoc.2020.106619

11. H. Li, J. Zhang, Fast source term estimation using the PGA-NM hybrid method, *Eng. Appl. Artif. Intel.*, **62** (2017), 68–79. https://doi.org/10.1016/j.engappai.2017.03.010

12. L. Wang, J. Zhang, H. Li, An improved genetic algorithm for TSP, in *2007 International Conference on Machine Learning and Cybernetics*, (2007), 925–928. https://doi.org/10.1109/ICMLC.2007.4370274

13. G. C. Chen, Y. U. Jin-Shou, Particle swarm optimization algorithm, *Inf. Control*, (2005), 454–458.

14. J. Zhang, W. Si, Improved enhanced self-tentative PSO algorithm for TSP, in *2010 Sixth International Conference on Natural Computation*, (2010), 2638–2641. https://doi.org/10.1109/ICNC.2010.5583011

15. M. Dorigo, G. D. Caro, L. M. Gambardella, Ant algorithms for discrete optimization, *Artif. Life*, **5** (1999), 137–172. https://doi.org/10.1162/106454699568728

16. F. Valdez, I. Chaparro, Ant colony optimization for solving the TSP symetric with parallel processing, in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, (2013), 1192–1196. https://doi.org/10.1109/IFSA-NAFIPS.2013.6608570

17. S. Gao, J. Zhong, Y. Cui, C. Gao, X. Li, A novel pheromone initialization strategy of ACO algorithms for solving TSP, in *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, (2017), 243–248. https://doi.org/10.1109/FSKD.2017.8393155

18. Y. Zhang, C. Wang, H. Li, X. Su, M. Zhao, N. Zhang, An improved 2-Opt and ACO hybrid algorithm for TSP, in *2018 Eighth International Conference on Instrumentation Measurement, Computer, Communication and Control (IMCCC)*, (2018), 547–552. https://doi.org/10.1109/IMCCC.2018.00121

19. R. W. Dewantoro, P. Sihombing, Sutarman, The combination of ant colony optimization (ACO) and Tabu search (TS) algorithm to solve the traveling salesman problem (TSP), in *2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*, (2019), 160–164. https://doi.org/10.1109/ELTICOM47379.2019.8943832

20. T. Öncan, A survey of the generalized assignment problem and its applications, *INFOR Inform. Syst. Oper. Res.*, **45** (2008), 123–141. https://doi.org/10.3138/infor.45.3.123

21. T. Huang, Y. J. Gong, J. Zhang, Seeking multiple solutions of combinatorial optimization problems: a proof of principle study, in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, (2018), 18–21. https://doi.org/10.1109/SSCI.2018.8628856

22. T. Huang, Y. J. Gong, S. Kwong, H. Wang, J. Zhang, A niching memetic algorithm for multi-solution traveling salesman problem, *IEEE Trans. Evolut. Comput.*, **24** (2020), 508–522. https://doi.org/10.1109/TEVC.2019.2936440

23. X. C. Han, H. W. Ke, Y. J. Gong, Y. Lin, W. L. Liu, J. Zhang, Multimodal optimization of traveling salesman problem: a niching ant colony system, *Proc. Genet. Evol. Comput. Conf. Companion*, (2018), 87–88. https://doi.org/10.1145/3205651.3205731

24. L. Gilbert, The traveling salesman problem: an overview of exact and approximate algorithms, *Eur. J. Oper. Res.*, **59** (1992), 231–247. https://doi.org/10.1016/0377-2217(92)90138-Y

25. B. Fleischmann, A cutting plane procedure for the travelling salesman problem on road networks, *Eur. J. Oper. Res.*, **21** (1985), 307–317. https://doi.org/10.1016/0377-2217(85)90151-1

26. H. P. Hipólito, S. G. Juan, A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery, *Discrete Appl. Math.*, **145** (2004), 126–139. https://doi.org/10.1016/j.dam.2003.09.013

27. M. Dorigo, L. M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evol. Comput.*, **1** (1997), 53–66. https://doi.org/10.1109/4235.585892

28. K. A. De Jong, *An Analysis of the Behavior of A Class of Genetic Adaptive Systems*, ProQuest Dissertations Publishing, 1975.

29. D. E. Goldberg, J. Richardson, Genetic algorithms with sharing for multimodal function optimization, in *Genetic Algorithms and Their Applications, Proceedings of the Second International Conference on Genetic Algorithms*, Proc Icga, (1987), 41—49.

30. A. Petrowski, A clearing procedure as a niching method for genetic algorithms, in *Proceedings of IEEE International Conference on Evolutionary Computation*, (1996), 798–803. https://doi.org/10.1109/ICEC.1996.542703

31. G. R. Harik, Finding multimodal solutions using restricted tournament selection, in *Proceedings of the 6th International Conference on Genetic Algorithms*, (1995), 24–31.

32. J. P. Li, M. E. Balazs, G. T. Parks, P. J. Clarkson, A species conserving genetic algorithm for multimodal function optimization, *Evol. Comput.*, **10** (2002), 207–234. https://doi.org/10.1162/106365602760234081

33. X. Li, M. G. Epitropakis, K. Deb, A. Engelbrecht, Seeking multiple solutions : an updated survey on niching methods and their applications, *IEEE Trans. Evol. Comput.*, **21** (2017), 518–538. https://doi.org/10.1109/TEVC.2016.2638437

34. H. Li, P. Zou, Z. Huang, C. Zeng, X. Liu, Multimodal optimization using whale optimization algorithm enhanced with local search and niching technique, *Math. Bio. Eng.*, **17** (2020), 1–27. https://doi.org/10.3934/mbe.2020001

35. T. Bektas, L. Gouveia, Requiem for the Miller-Tucker-Zemlin subtour elimination constraints?, *Eur. J. Oper. Res.*, **236** (2014), 820–832. https://doi.org/10.1016/j.ejor.2013.07.038

36. G. Szekely, M. L. Rizzo, N. K. Bakirov, Measuring and testing dependence by correlation of distances, *Ann. Statist.*, **35** (2007), 2769–2794. https://doi.org/10.1214/009053607000000505

37. J. Bank, B. Cole, Calculating the Jaccard similarity coefficient with map reduce for entity pairs in wikipedia, *Wikipedia Similarity Team*, **1** (2008), 94.

38. M. Shameem, R. Ferdous, An efficient k-means algorithm integrated with Jaccard distance measure for document clustering, in *Asian Himalayas International Conference on Internet*, (2009), 1–6. https://doi.org/10.1109/AHICI.2009.5340335

39. G. A. Croes, A method for solving travelling salesman problems, *Oper. Res.*, **6** (1958), 791–812. https://doi.org/10.1287/opre.6.6.791