A Niching Memetic Algorithm for Multi-Solution Traveling Salesman Problem

Ting Huang, Yue-Jiao Gong, *Senior Member, IEEE*, Sam Kwong, *Fellow, IEEE*, Hua Wang, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Multi-solution problems extensively exist in practice. Particularly, the traveling salesman problem (TSP) may possess multiple shortest tours, from which travelers can choose one according to their specific requirements. However, very few efforts have been devoted to the multi-solution problems in the discrete domain. In order to fill this research gap and to effectively tackle the multi-solution TSP, we propose a niching memetic algorithm in this article. The proposed algorithm is characterized by a niche preservation technique to enable the parallel search of multiple optimal solutions; an adaptive neighborhood strategy to balance the exploration and exploitation; a critical edge-aware method to provide effective guidance to the reproduction; and a selective local search strategy to improve the search efficiency. To evaluate the performance of the proposed algorithm, we conduct comprehensive experiments on a recently published multi-solution optimization test suite. The experimental results show that our algorithm outperforms other compared algorithms. Furthermore, the proposed algorithm is adopted to tackle problems from the well-known TSPLIB library to obtain a set of distinct but good solutions.**

***Index Terms*—Memetic algorithm, multimodal optimization (MMOP), multi-solution traveling salesman problem (MSTSP), neighborhood niching strategy.**

I. Introduction

TRAVELING salesman problem (TSP) [1], [2] is one of the most intensively studied combinational optimization problems. Many engineering and academic issues involve TSPs as subproblems, such as the vehicle routing problems [3]–[5], the coding order optimization problem [6], and the in-port ship routing and scheduling problem [7]. In the TSP, a salesman leaves from a city, travels each city once and only once, and finally returns to the beginning city. The goal of the salesman is to find the shortest traveling tour. In order to meet different needs of practical applications, researchers have proposed a variety of TSP variants, including multi-objective TSP [8], [9], multi-salesman TSP [10], [11], and Dubins TSP [12], [13]. Most TSP applications and TSP variants focus on locating one optimal solution, regardless of the fact that there may be multiple high-quality solutions. However, in practice, it is equally important to provide diverse optimal alternatives to decision makers so that: 1) quick actions can be made under emergency events (e.g., traffic congestion due to road works in the city or the cancelation of flights due to bad weathers) and 2) traffic load can be balanced because of different choices of routes. This requirement gives rise to the research on multi-solution TSP (MSTSP).

Because of the NP-hard nature, it is quite difficult to tackle MSTSPs by deterministic algorithms. The evolutionary algorithms (EAs) are population-based optimizers with meta-heuristic search nature, which avoids the need of exhaustively exploring the entire problem space and allows to find an optimum within a reasonable time [14]. There are several related studies of applying EAs for MSTSPs [15]–[18]. However, we found that the algorithms failed to find any optimum when the city size increases. Therefore, a more powerful and effective algorithm is highly desired to tackle MSTSPs. Besides, dealing with the MSTSP belongs to the multimodal optimization (MMOP) area. In the literature, most efforts paid to the MMOP concentrate on the continuous optimization domain [19]–[29]. In order to deal with the discrete MMOP such as the MSTSP, it is highly desired to develop new optimization algorithms or to conduct domain extension on the available continuous algorithms. Generally, two critical issues need to be considered when designing an effective algorithm for the discrete MMOP: 1) the choice of an appropriate discrete baseline algorithm, so as to offer promising search capability and 2) the technique of population diversity preservation in the discrete domain, so as to maintain diverse candidate solutions during the search.

Paying attention to the baseline algorithm, traditionally, differential evolution (DE) [19]–[22], particle swarm optimization (PSO) [23], [24], and covariance matrix adaptation evolutionary strategies (CMA-ES) [25]–[27] are

Manuscript received October 22, 2018; revised March 14, 2019 and June 11, 2019; accepted August 13, 2019. Date of publication August 20, 2019; date of current version May 29, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61873095, Grant 61873097, and Grant U1701267, in part by the Guangzhou Science and Technology Planning Project under Grant 201904010211, in part by the Guangdong Natural Science Foundation Research Team Project under Grant 2018B030312003, and in part by the Guangdong-Hong Kong Joint Innovation Platform Project under Grant 2018B050502006.

T. Huang and Y.-J. Gong are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: gongyuejiao@gmail.com).

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

H. Wang is with the College of Engineering and Science, Institute for Sustainable Industries and Liveable Cities, Victoria University, Melbourne, VIC 8001, Australia.

J. Zhang is with Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2019.2936440

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

extensively studied for the continuous MMOP, because these algorithms are originally designed for the continuous search space. As for discrete or combinatorial optimization, the other evolutionary computation optimizers, such as ant colony algorithm (ACO) and genetic algorithm (GA), could be more favored [30]–[34].

As another critical issue, the population diversity should be considered when tackling MSTSPs. In the population-based algorithms, the population would eventually converge toward one basin of attraction due to the diversity loss caused by genetic drift and selection pressure [35]. The genetic drift causes the disappearance of gene variants in the population, while the selection pressure biases the evolution of individuals to a small part of the search space but ignores the population diversity. To avoid converging toward one basin of attraction and preserve the population diversity, most MMOP solvers utilize niching techniques, which are inspired from biology. In the niching techniques, similar species compete for regional resources. By putting restrictions on the competition of species, the effect of the genetic drift and the selection pressure on the entire population can be eliminated, and the potential candidates of different basins of attraction can survive across generations. Currently, popular niching techniques include crowding [36], [37], speciation [38], clustering [22], [39], [40], and neighborhood strategies [19]. These niching techniques are mainly used for the continuous MMOP, where the Euclidean distance is commonly adopted as a similarity indicator.

In order to deal with MSTSP, in this article, we propose a niching memetic algorithm (NMA) framework. The reason for choosing MA as the baseline algorithm is that MAs have been widely validated as powerful TSP optimizers in [32] and [41]–[52]. (A more detailed MA-related review is presented in Section SI in the supplementary material, where we introduce various MAs specified for TSPs.) Generally, the local search endows MA a promising exploitation ability in the combinatorial search space. However, the previous studies conduct a global optimization for TSP and finally obtain one single optimum. For solving MSTSP, the proposed NMA incorporates a neighborhood niching strategy that enables parallel search for multiple optima. Under the framework of NMA, we specifically design four strategies to improve the algorithm performance. The four auxiliary methods help in achieving exploration–exploitation balance, critical edge awareness, fitness evaluation saving, and decision-maker-friendliness, respectively. Generally speaking, the main contributions of this article are summarized as follows.

A. Filling the Gap of Benchmark Studies of the Discrete Multi-Solution Optimization

Currently, the research progresses on continuous and discrete MMOPs are proceeded at different paces. The former study maintains a momentum of vigorous growth [29], whereas very few attention has been paid to the discrete MMOP area. Many practical problems are discrete in nature and it is appealing to suggest multiple optima for them, such as truss-structure optimization [53], protein structure prediction [54], and job shop scheduling optimization [55], [56]. However, MSTSP, which is derived from the most well-known discrete optimization problem—TSP, has not yet been comprehensively researched. This article targets on promoting the benchmark studies of the discrete MMOP by taking MSTSP as an example. The method and experimental analysis can provide some guidelines for this area and stimulate the further research progress.

B. Designing Novel NMA for MSTSP

NMA takes MA as the baseline algorithm to promise efficient search ability and incorporates a niching strategy to maintain population diversity. Besides, we specifically design four strategies for performance enhancement.

1) Adaptive Neighborhood Strategy: For a neighborhood-based EA, the number of group members indicates the amount of the search resource assigned to the group. The allocation of search resource has an influence on the final solution quality. In NMA, we adaptively determine the neighborhood size according to the search state of each neighborhood, so as to achieve a balance between the exploration and exploitation during the optimization.

2) Critical Edge-Aware Method: For discrete multi-solution optimization, different optimal solutions may share some common elements. We develop a concept of critical edge and design critical edge-aware (CEA) evolution strategies in NMA. The CEA method not only provides auxiliary guidance to the reproduction of population but it also avoids the unnecessary exploitation in the selective local search introduced below.

3) Selective Local Search: Local search facilitates quick convergence, but the side effect is that it usually brings enormous consumption of fitness evaluations by exhaustively exploring the neighborhood region of any given solution. In NMA, to save the inefficient local search on some inferior and redundant solutions, we design a selective local search strategy, which biases promising and distinct individuals, as well as uncritical edges, to maximize the search efficiency.

4) Elite Selection Approach: It is unnecessary to provide all candidate solutions to a decision maker, but a set of representative optima is more appealing. The elite selection approach eliminates redundant and inferior solutions and, subsequently, outputs the distinct and superior ones.

C. Conducting Rich Experiments and Showing Promising Results

A few existing MSTSP optimizers are tested on some toy instances, whereas we have developed a standard benchmark test suite in [18]. The test suite consists of 25 MSTSP instances with different characteristics, which are used to test the performance of NMA and the others in this article. In addition, we also adopt TSPLIB [57] repository to make further investigation and show that many instances in the repository have multi-solution nature. The experimental results show that compared with the rivals, NMA can obtain more high-quality and diverse solutions in a relatively short time, while being less affected by the problem size. Further investigations validate that the proposed auxiliary strategies strengthen the algorithm on their respective advantages. We are the first attempt to performing the algorithms on such abundant MSTSP instances. The results of NMA provides the baselines for future studies in the MSTSP optimization area.

The rest of this article is organized as follows. Section II introduces MSTSP and reviews multi-solution optimization algorithms for MSTSP. Section III describes the proposed algorithm and gives the details of its components. The experimental results and relevant analysis are demonstrated in Section IV. Section V shows the extension work on TSPLIB. Finally, the conclusions are summarized in Section VI.

II. Background

A. Multi-Solution Traveling Salesman Problem

In many TSP-related practical applications, multiple approximately good solutions are required. For example, for route scheduling applications, it is always suggested to provide multiple acceptable routes so that the drivers can select the most favorable one according to his own knowledge. In addition, because the road condition is dynamically changed, one route may become invalid due to traffic jam or road maintenance. In these cases, the drivers would like to quickly change to a candidate route with the same quality to accomplish his/her task in time. Moreover, the traffic load can be balanced with different options of routes. These application backgrounds motivate us to research into the MSTSP area.

This section introduces and formulates MSTSP, which has exactly the same mathematical form as TSP. Given a list of cities, a salesman travels each city once and only once, constructing a Hamilton path. The aim of the MSTSP solver is to find all the shortest Hamilton paths. Mathematically, consider a graph $G = (V, E)$, where $V = \{1, 2, 3, \ldots, N\}$ is a set of cities (denoted by the city indices), and $E = \{(i, j)|i, j \in V, i \neq j\}$ is a set of edges, indicating the road/edge between the cities i and j. Each edge (i, j) has a weight value, denoted as d_{ij} that measures the distance between the two cities i and j. The distance is rounded to the nearest integer, as the edge weight type EUC_2D described in TSPLIB [57]. The Hamilton path can be formulated as the permutation π. Thereafter, MSTSP is to find a shortest length in all possible permutations. More precisely, we consider

$$\min f(\pi) = \sum_{k=1}^{N-1} d_{\pi(k)\pi(k+1)} + d_{\pi(N)\pi(1)} \tag{1}$$

where N is the number of cities and $\pi(k)$ is the kth element of the permutation π. The target of MSTSP is to find a set of solutions with the exactly same minimum tour length.

Recently, we released a comprehensive benchmark of 25 MSTSP instances of different characteristics [18]. According to the design methods, the MSTSPs are classified into three categories, i.e., simple MSTSPs, geometry MSTSPs, and composite MSTSPs.

1) Simple MSTSPs are randomly generated and the number of optima is uncontrollable, but we apply an accurate algorithm to obtain the optimal solution set.
2) Geometry MSTSPs are designed using different symmetrical geometries, such as the rectangle, the regular pentagon, and the regular hexagon. The optima can be adjusted by controlling the combination of specific geometries, the location of geometries, and the rotation of the geometries. Fig. 1 illustrates MSTSP9, one of the geometry MSTSPs, with ten cities and four optimal solutions. In the subfigures, the black circles represent city locations and the red polyline connecting all cities indicates an optimal route. Besides, the numbers at the top of the subfigures represent the lengths of the routes.
3) Composite MSTSPs are more complicated, which apply several small-scale MSTSPs to construct a larger instance. Each small-scale MSTSP is a city cluster, and the city clusters are distributed at different geometric locations. The multi-solution characteristics are derived from two relationships: a) the intra-cluster and b) the inter-cluster relationships. The number of optima is under control by assembling different types of small-scale MSTSPs. In general, for the 25 MSTSP instances, the number of cities ranges from 9 to 66, and the number of optima scales from 2 to 196.

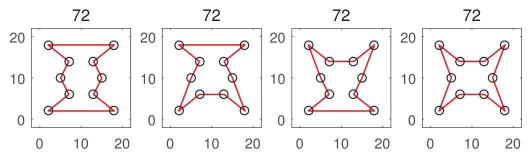

Fig. 1. Optimal solutions of MSTSP9.

Note that some accurate algorithms can deal with small-scale MSTSPs based on the current computation condition. For example, we utilize the branch-and-bound (BnB) algorithm to solve the problems and find that for the MSTSP3/6/12 with 10/12/15 cities, the BnB algorithm consumes 0.07 s, 2.87 s, and 2255.74 s, respectively, to obtain the complete optima set. The time cost increases dramatically by the city size. When dealing with MSTSPs with even larger sizes, such as the composite MSTSPs ($N \geq 28$), the time overhead of the accurate algorithm becomes intolerable. Note that since it requires to find multiple optimal routes, the MSTSPs are much harder than the traditional TSPs, so that the perception of "large scale" is different between MSTSPs and TSPs. In this article, the MSTSPs with $N \geq 28$ are regarded as large-scale cases, for which the accurate algorithm cannot stop execution in an acceptable time.

B. Multi-Solution Optimization Algorithms for MSTSP

TSP is an NP-hard problem that the accurate algorithms endure the "curse of dimensionality." Requiring to obtain multiple optima, MSTSP is even harder than TSP. The optimizers for MSTSP should not only possess a high search efficiency for satisfactory solutions but also maintain a good population diversity for diverse solutions. With limited search resources, these two targets are somewhat contradictory, which are hard to be accomplished simultaneously.

There are several multi-solution optimization algorithms available for tackling MSTSP, including the multi-chromosome cramping-based GA (MCC-GA) [15], niche-based ant colony system (NACS) [17], and the neighborhood-based GA (NGA) [18]. MCC-GA encodes a set of s solutions into one chromosome. The problem space grows exponentially with the solution size s. From this point of view, it is difficult to locate all the optima without the prior knowledge of the number of optima. Different from the previous algorithms, NACS uses multiple pheromone matrices to locate diverse solutions. When ants find out a superior and distinct solution, a new pheromone matrix is created according to the solution. Ants on the different pheromone matrices tend to locate distinct optima. The problem encountered by NACS is about the niche parameters, which are difficult to appropriately set without the prior knowledge of problems. NGA gathers similar neighbors as a group and performs the basic genetic operations of GA within the group. The genetic drift and the selection pressure are limited in the corresponding groups, so as to locate diverse optima. However, NGA suffers from slow convergence and difficulties in setting an appropriate neighborhood size. To summarize, the development of discrete MMOP algorithms predominantly considers the diversity preservation, the convergence speed, and the appropriate parameter settings.

III. Niching Memetic Algorithm for MSTSP

This section introduces the proposed algorithm NMA. The algorithm is outlined first and described in part subsequently.

A. Framework

NMA takes MA as the baseline algorithm and incorporates a neighborhood niching strategy. The neighborhood niching strategy partitions the whole population into several neighborhood groups and every group tends to locate a different optimum. MA is applied to search effectively within the given search space. Under this framework, we design several enhancement strategies to improve the performance of the proposed algorithm.

The chromosomes adopt the permutation-based representation defined in Section II-A. Then, the procedure of NMA is described as follows. First, we initialize *NP* chromosomes using a partially greedy strategy. For one chromosome with N genes (N denotes the number of cities), the first $\lfloor N/2 \rfloor$ genes are constructed with randomly selected but distinct cities, and the last $\lceil N/2 \rceil$ genes choose the city one by one using the greedy strategy—it picks the closest unchosen city to the last determined city in the chromosome. Then, the order of the chromosomes is rearranged using a neighborhood niching strategy, where the topologically close chromosomes are grouped to compose a mating pool. Note that the permutation-based presentation contains both the node and edge information of a TSP route. When measuring the distance between two solutions, the edge set of the solution is adopted rather than the absolute node position set, which is going to be expounded in Section III-B. Meanwhile, a diversity enhancement approach is adopted to avert being trapped in local optima. The neighborhood leaders are utilized to construct a critical edge set (CES) for the problem. The CES is then adopted by the reproduction and local search strategies to provide additional guidance. According to the routine of GA, we perform the crossover operation with the probability P_c and the mutation operation with the probability P_m. Then, a selective local search is applied to strengthen the exploitation ability. Now, we obtain offspring and subsequently determine the next generation using a replacement operation. The above process is repeated until the terminal condition is met. When it comes to the end of the algorithm, we obtain a set of final solutions. A post-processing procedure is adopted to select elites from the final solution set. The overall process of NMA is outlined in **Algorithm 1**. The subcomponents are described in details one by one in the following.

Algorithm 1 Niching Memetic Algorithm

Input: An MSTSP instance T, the population size NP, the crossover rate P_c, the mutation rate P_m, a termination criteria, and the minimum (maximum) neighborhood size $M_{\min}$ ($M_{\max}$)
Output: A representative solution set $\mathbb{S}$

```
1:  $Parent$ ← Initialize($NP$)
2:  Evaluate($Parent$)
3:  $CES$ ← {}
4:  while the termination criterion not satisfied do
5:      $MatingPool$ ← NeighborhoodStrategy($Parent$, $M_{\min}$, $M_{\max}$) /* Algorithm 2 */
6:      $eMatingPool$ ← DiversityEnhancement($MatingPool$, $CES$) /* Algorithm 3 */
7:      UpdateCriticalEdgeSet($eMatingPool$, $CES$) /* Algorithm 4 */
8:      $cOffspring$ ← Crossover($eMatingPool$, $CES$, $P_c$)
9:      $mOffspring$ ← Mutation($cOffspring$, $CES$, $P_m$)
10:     $lsOffspring$ ← LocalSearch($mOffspring$, $CES$)
11:     Evaluate($lsOffspring$)
12:     $Parent$ ← Replacement($lsOffspring$, $MatingPool$, $CES$)
13: end while
14: $\mathbb{S}$ ← Preserve ($Parent$) /* Algorithm 5 */
```

B. Adaptive Neighborhood Strategy

With regards to the MMOP, it is essential to maintain population diversity, so as to hold multiple potential solutions during the search. To preserve the population diversity, we utilize the neighborhood niching strategy, whose essence is to limit the reproduction or selection pressure of a niche (or group) within a subspace. Through this method, members of the same niche can produce spatially close offspring or survive competitive and geographically close offspring. Eventually, diverse niches locate distinct solutions. One important part of the neighborhood niching strategy is how to set an appropriate neighborhood size. The group with too many members would lead to excessive search when the group tends to converge, while the group with too few members would result in insufficient search for the group requires exploration. To address this issue, we adaptively determine the neighborhood size according to the niche state. Here, the neighborhood size m is an integer number ranging from $M_{\min}$ to $M_{\max}$, where $M_{\min}$ and $M_{\max}$ denote the allowable minimum and maximum neighborhood size, respectively. In the rest of this section, we first introduce the neighborhood partition method and then elaborate on the adaptation mechanism of the neighborhood size.

Illustrated in **Algorithm 2**, the algorithm determines neighborhood groups iteratively. First, the unprocessed chromosome with the shortest length is identified as the group leader

Algorithm 2 *MatingPool* $\leftarrow$ NeighborhoodStrategy(*Parent*, $M_{\min}$, $M_{\max}$)

```
1:  $tmpParent \leftarrow Parent$
2:  $MatingPool \leftarrow \{\}$
3:  $L_{\min} \leftarrow$ MinLength($Parent$)
4:  $L_\beta \leftarrow$ betaLength($Parent$, $NP/M_{\min}$)
5:  while $tmpParent \neq \varnothing$ do
6:      $leader \leftarrow$ FindBest($tmpParent$) /*Find out the best individual in $tmpParent$.*/
7:      if $|tmpParent| \geq M_{\max}$ then
8:          $\gamma \leftarrow \min((leader.\text{Len} - L_{\min})/(L_\beta - L_{\min}), 1.0)$
9:          $m \leftarrow floor(\gamma \times (M_{\max} - M_{\min}) + M_{\min})$
10:         if mod$(m, 2) \neq 0$ then
11:             $m = m + 1$
12:         end if
13:     else
14:         $m \leftarrow |tmpParent|$
15:     end if
16:     CalculateShareDist($tmpParent$, $leader$) /*Calculate the sharing distance between each individual in $tmpParent$ and $leader$.*/
17:     $sortParent \leftarrow$ Sort($tmpParent$.shareDist, "desc") /*Sort the individuals by the sharing distance in an descending order.*/
18:     $NeighborGroup \leftarrow sortParent[1, \ldots, m]$ /*Assign the first $m$ closest sorted individuals to $NeighborGroup$.*/
19:     Shuffle($NeighborGroup$) /*Shuffle the order of $NeighborGroup$.*/
20:     $MatingPool \leftarrow MatingPool + NeighborGroup$
21:     $tmpParent \leftarrow tmpParent - NeighborGroup$
22: end while
```

(*leader*). If the number of the unprocessed chromosomes is larger than or equal to $M_{\max}$, the neighborhood size m is adaptively determined according to the length of the *leader* (which will be detailed afterward). Then, the unprocessed members are sorted by the sharing distance defined in Eq. (2), in a descending order. The first m sorted members (including the *leader*) form a new neighborhood group. The newly formed group is shuffled and then added to the mating pool. In the meantime, the added members are labeled processed and eliminated from the current population. The above process is repeated until all the members are processed and all the neighborhood groups are settled.

In the above neighborhood partition process, we need to calculate the pairwise similarity between the current *leader* and each unprocessed members. In this article, the similarity of two permutations π_i and π_j is measured by sharing distance as follows:

$$S(\pi_i, \pi_j) = \frac{|\Phi(\pi_i) \cap \Phi(\pi_j)|}{N} \tag{2}$$

where $\Phi(\pi_i)$ and $\Phi(\pi_j)$ denote the edge set of π_i and π_j, respectively; $\Phi(\pi_i) \cap \Phi(\pi_j)$ means the intersection set of $\Phi(\pi_i)$ and $\Phi(\pi_j)$; and $|\cdot|$ denotes the number of edges in $\Phi(\pi_i) \cap \Phi(\pi_j)$. The sharing distance measure operates on the edge set of a route rather than the absolute node positions. So that the solutions having different absolute node orders but the same visiting edge set are identified as the same solution.

The neighborhood size m is adaptively adjusted to improve the generalization performance of the proposed algorithm. In niching methods, the neighborhood size can be regarded as the allocated search resources for each solution. The motivation of our adaptive neighborhood strategy is to allocate the search resources according to the current search state of the niche. In the proposed algorithm, a niche is covered by a neighborhood group, and the neighborhood members are dominated by the neighborhood leader. First, we calculate a niche state measure $\gamma \in [0, 1]$ according to the fitness value (i.e., length) of the neighborhood leader *leader*:

$$\gamma = \min\left(\frac{L_{leader} - L_{\min}}{L_\beta - L_{\min}}, 1.0\right) \tag{3}$$

where L_{leader} is the length of *leader*; $L_{\min}$ and L_β are the shortest length and the βth shortest length of the population, respectively; and $\beta = NP/M_{\min}$ indicates the maximum number of possible neighborhood leaders. The fitness value L_β can be considered as a cutoff point to identify whether the neighborhood leader is "superior" or "inferior."

Then, the neighborhood size m is adjusted based on γ using

$$m = \lfloor \gamma \times (M_{\max} - M_{\min}) + M_{\min} \rfloor. \tag{4}$$

Based on Eqs. (3) and (4), there are three situations as follows.

1) If L_{leader} is shorter than L_β and equal to $L_{\min}$, indicating that the *leader* is superior. The calculated γ value is 0 and, hence, the neighborhood size m is $M_{\min}$. The superior *leader* is allocated with the least search resource since this solution requires minor improvements than the other solutions.
2) If L_{leader} is larger than L_β, meaning that the *leader* is inferior. According to Eqs. (3) and (4), γ equals to 1 and m equals to $M_{\max}$. The algorithm assigns the most search efforts to the inferior neighborhood groups to help them quickly identify good solutions.
3) If L_{leader} is shorter than L_β but larger than $L_{\min}$, implying that the *leader* is neither the best nor the worst but between the two extremes. The niche state measure γ is a real value between 0 and 1 for quantifying the quality of this point between the two extremes. Next, based on Eq. (4), the neighborhood size is between $M_{\min}$ and $M_{\max}$, depending on γ.

Note that $M_{\min}$ and $M_{\max}$ are utilized to limit the lower and upper bounds of the neighborhood size. These boundary settings are commonly necessary for the parameter adaptation methods [58]–[60]. Although they are additional parameters, the boundary parameters are not problem-dependent, which will not reduce the generalization performance of the algorithms. As the GA requires pairs of chromosomes to conduct the crossover, we recommend to set $M_{\min}$ and $M_{\max}$ value as even integers larger than 2.

Explanatory Diagram: Suppose the population has six chromosomes (i.e., NP = 6), which are illustrated in Fig. 2. In addition, assume that the minimum neighborhood size $M_{\min}$ = 2 and the maximum neighborhood size $M_{\max}$ = 4. Thus, we have $\beta = 6/2 = 3$. The tour lengths of the chromosomes are listed in dashed boxes. It can be calculated from Fig. 2 that the minimum length $L_{\min}$ and the βth length L_β are 150 and 250, respectively. After that we calculate the pairwise sharing distances of chromosomes, which are shown in the lower triangular matrix of Fig. 3, where the values in the diagonal represent the sharing distance to itself, i.e., $S(\pi_i, \pi_i) = 1.00$. Specifically, the first best chromosome, P1, is chosen as the

P1	P2	P3	P4	P5	P6
150	250	250	300	400	500

Fig. 2. Sorted parental chromosomes with their lengths shown below.

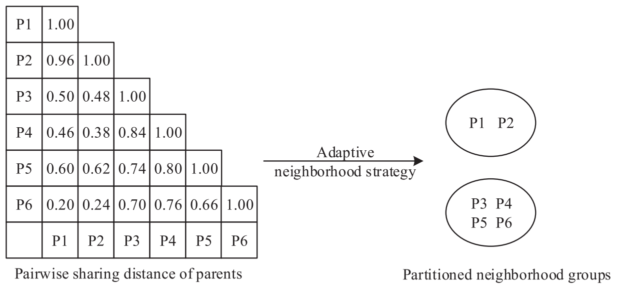
P1 1.00
P2 0.96 1.00
P3 0.50 0.48 1.00
P4 0.46 0.38 0.84 1.00
P5 0.60 0.62 0.74 0.80 1.00
P6 0.20 0.24 0.70 0.76 0.66 1.00
P1 P2 P3 P4 P5 P6
Pairwise sharing distance of parents
Adaptive neighborhood strategy
P1 P2
P3 P4 P5 P6
Partitioned neighborhood groups

Fig. 3. Diagram of the adaptive neighborhood strategy. The lower triangular matrix on the left lists the pairwise sharing distance of the parental chromosomes. The two ellipses on the right represent two partitioned neighborhood groups.

first neighborhood leader. Recalling Eqs. (3) and (4), we obtain the first neighborhood size being 2. Thereafter, we choose the first two nearest neighbors, i.e., P1 and P2, to comprise the first neighborhood group. When it turns to the second neighborhood group, another similar process goes over. The neighborhood leader in the remaining members is chosen, that is, P3. The neighborhood size is 4 according to Eqs. (3) and (4). Thus, the first four nearest neighbors to P3 are chosen to comprise the second neighborhood group. The two resultant neighborhood groups are drawn within the ellipses on the right of Fig. 3.

C. Diversity Enhancement Approach

The adaptive neighborhood strategy rearranges the order of chromosomes and forms an ordered mating pool. However, with the population evolving, group members gather around, being trapped in a local optimum, which leads to diversity loss to some extent. The evolution with converged neighborhoods makes the group members similar or even the same. The further search within this kind of converged group can neither bring in more diversity nor locate any better solution. Out of this concern, we design a diversity enhancement approach to help the converged group jumping out of local attraction and moving toward other spatially near better locations. To be specific, for each neighborhood group in the mating pool, we calculate the sharing distances between the first member and each member in the group to check whether they possess the same permutations (i.e., the sharing distance of the same permutations equals to 1). If all members are the same, we consider this group is converged and subsequently adopt the mutation operation (will be introduced in Section III-E) to perturb all the members except the *leader* in the converged group. After the perturbation, the converged members are redistributed to the near region. The procedure is described in **Algorithm 3**.

Algorithm 3 *eMatingPool* ← DiversityEnhancement (*MatingPool*, *CES*)

```
1: $eMatingPool$ = {}
2: for each $NeighborGroup \in MatingPool$ do
3:     $leader \leftarrow NeighborGroup.\mathrm{Leader}$
4:     $flag \leftarrow$ true
5:     for each $p \in NeighborGroup$ do
6:         if $S(p, leader) \neq 1$ then
7:             $flag \leftarrow$ false
8:             break
9:         end if
10:    end for
11:    if $flag$ == true then
12:        $newNeighborGroup \leftarrow \mathrm{Mutation}(NeighborGroup - leader, CES, 1.0)$
13:        $eMatingPool = eMatingPool + newNeighborGroup$
14:    else
15:        $eMatingPool = eMatingPool + NeighborGroup$
16:    end if
17: end for
```

D. Update of Critical Edge Set

The MSTSP has some features, including that it has a discrete (finite) problem space and it requires to find multiple optimal solutions. Therefore, the different optimal solutions of an MSTSP may share some common elements. This conforms to the practical situations such as in the route scheduling applications that the multiple optimal routes may share the same subset of visiting edges. The MSTSP benchmark test suite also well characterizes this situation. These common edges are critical for constructing an optimal solution. Thus, if the critical edges can be identified, it would be helpful to enhance the operators of NMA, so as to further improve the search efficiency, especially for the large scale MSTSPs. Therefore, we bring in the concept of critical edge and develop CEA evolution strategies in NMA.

The CES is incrementally updated during the evolution. The insertion and deletion mechanisms are described as follows.

1) *Insertion*: Each chromosome in the population is attached with a stagnation generation value *sg* that indicates how long the fitness of the chromosome has not been updated. The neighborhood leaders whose *sg* values are larger than *N* are identified as persistent leaders and added to the set *pLeader*. The algorithm inserts each edge belongs to the intersection of solutions' edges in the *pLeader* into CES.
2) *Deletion*: The insertion of an edge may cause the following invalid situations: a) the edge is already existed in the CES and b) a node is connected with more than two other nodes in CES. The first in first out (FIFO) method is adopted to deal with these invalid situations. Namely, for the conflicted edges in CES, the earliest inserted one is deleted. Besides, to keep the CES up-to-date, in each generation, the edges inserted before *N* earlier generations are also deleted from the CES.

The overall procedure of CES update is summarized in **Algorithm 4**. Afterward, the CES is adopted to improve the following reproduction and local search operations of NMA.

E. Critical Edge-Aware Reproduction

Taking GA as the baseline, the reproduction of the proposed algorithm consists of a crossover and a mutation operation. The crossover operation mates parental chromosomes to breed

Algorithm 4 UpdateCriticalEdgeSet(*eMatingPool*, *CES*)

```
1: $pLeader = \{\}$
2: for each $NeighborGroup \in eMatingPool$ do
3:     $leader \leftarrow NeighborGroup$.Leader
4:     if $leader.sg \geq N$ then
5:         $pLeader = pLeader + leader$
6:     end if
7: end for
8: $candidates = \bigcap_{leader \in pLeader} \Phi(leader)$ /*Calculate the intersection of solutions' edges in $pLeader$.*/
9: for each $edge \in candidates$ do
10:    $CES = CES + edge$
11:    conflictedFIFO($CES$, $edge$) /*Once conflicted, use FIFO.*/
12: end for
13: deleteOldest($CES$) /*Delete the edges inserted before $N$ earlier generations.*/
```

children, expecting to generate new competitive individuals. The crossover operation working on the entire population would cause genetic drift that makes all the candidate solutions tend to be same. However, it should be avoided for multi-solution problems. Derived from the idea that similar parents would generate similar offspring, we limit the scope of the crossover operation in the neighborhood group that comprises with similar chromosomes. In the adaptive neighborhood strategy (described in Section III-B), we gather the similar chromosomes in the same neighborhood group, and meanwhile construct the mating pool in groups. Thereafter, the crossover operation can conveniently pair two parents in the sequence of the resultant mating pool.

This article improves the partially mapped crossover (PMX) [61] by incorporating a CEA method. The resultant crossover is named CEA-PMX. First, two exchange points are randomly chosen from the chromosome. The segmental genes between the two exchange points of one parent are picked out and passed to a child directly. Second, the edges in the intersection of the parent and the CES are also inherited by the child. Third, the remaining part of this child is inherited from another parent. During the inheritance process, if the candidate gene has already been included in the child, the corresponding mapping gene (the allele of the other parent) is copied instead. The other child is generated in a similar way.

Explanatory Diagram: The operation of CEA-PMX is illustrated in Fig. 4. Suppose that we have CES = {(6, 7)}. First, two exchange points are randomly generated at the positions of three and six, respectively. In the following, the segmental genes between the exchange points of P1 are directly passed into O1 (the passed genes are marked blue in O1). Then, the critical edge (6, 7) in P1 is also inherited by O1. For the vacant locus of O1, they inherit the genes from P2 in the corresponding vacant positions (the inherited genes are marked red in O1). However, three genes to be inherited have already been included in O1, i.e., genes 7, 6, and 5, and thus their mapping genes, 1 for 7, 10 for 6, and 3 for 5, take the places instead (the alternate genes are marked black in O1). The child O2 is generated by the similar method.

The mutation operation perturbs chromosomes to bring in diversity. We amend the exchange mutation (EM) proposed

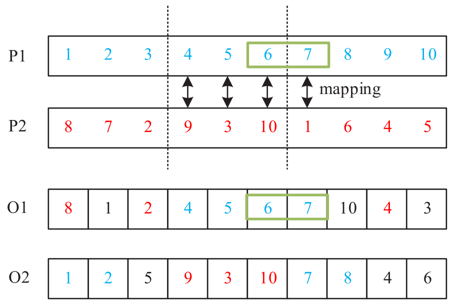

Fig. 4. Sorted parental chromosomes with their lengths shown below.

by Banzhaf [62] to make it fit the proposed algorithm. For a candidate chromosome, EM randomly exchanges two genes. The number of exchanged edges is two when the selected genes are adjacent; otherwise, the number is four. However, considering that the former case is similar to the 2-opt used in our local search, such operation is inefficient, since the mutated part could be reverted back after the following local search. To avoid such inefficient situations, we force to select nonadjacent genes in the mutation. Besides, the mutation preserves critical edges on the chromosomes, if there are any. The mutation is named CEA nonadjacent EM (CEA-NEM).

F. Selective Local Search

Local search expects to exploit within a small vicinity of a given solution, which is a promising technique for enhancing the exploitation. Particularly, when the local search is performed to the same neighborhood group, it is regarded to search around the covering subspace of this group. We adopt 2-opt as our local search approach, as 2-opt consumes the least time among the common local search operations. Each search step of 2-opt exchanges two edges of a permutation. Owing to the particularity of permutation, it is unnecessary to re-evaluate the length of the tour after each move (one re-evaluation consumes one fitness evaluation). In the view of operational complexity, each search step subtracts two edges and adds two edges, that is to say the number of operations is 4. One complete evaluation takes N operations (N denotes the number of elements in the permutation), and therefore, we count one fitness evaluation when the 2-opt exchanging operations achieve $N/4$ times.

Typically, local search is performed on each individual in each generation. However, some search efforts could be less effective because of the following reasons. First, the individuals in the small neighborhood are very similar to each other. The local search could be repetitive for these similar individuals. Second, the improvement that local search brings to some inferior individuals would disappear, since new competitive children are generated by evolutionary operations and take the place of the inferior parental individuals. Third, the edges in CES are the currently identified critical edges, which can be protected owing to its high possibility of optimality. Therefore, to save unnecessary search steps, the proposed selective strategy biases the local search on the uncritical edges of the promising and diverse candidate solutions. To be specific, by the adaptive neighborhood strategy described in Section III-B,

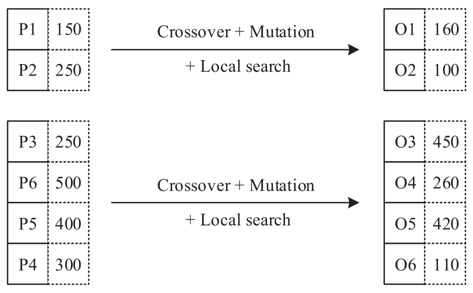

Fig. 5. Diagram of neighborhood groups. The parental neighborhood groups are listed on the left; the corresponding offspring groups, generated with crossover, mutation, and local search, are listed on the right.

we roughly classify the neighborhood groups into two cases as follows.

1) The neighborhood group whose size is exactly $M_{\min}$ tends to be converged. Converged group comprises close individuals. Therefore, it is redundant to perform local search on all the group members. Consequently, for such groups, one of the best members performs local search.
2) For the other neighborhood groups, they are relatively diverse and require explorative search. Hence, the first half of the members with the shortest tours are selected to adopt local search. Meanwhile, in the above procedure, the critical edges in CES are neglected, which do not undergo the 2-opt.

G. Replacement Operation

After a series of operations are performed, we obtain final offspring. Replacement operation brings in selection pressure to determine which offspring survive to next generation. Selection pressure promotes the evolution process, but it forces the population converging toward one optimum. In order to effectively maintain multiple solutions, we limit the selection pressure within respective neighborhood groups, rather than on the global space, so as to simultaneously search in different potential subspaces. The specific operation is that every child competes with the spatially nearest member/members of the corresponding parental neighborhood group. Meanwhile, if more than one member has the same largest sharing distance with the child, the member with the longest tour is selected as the competitor. After the competition, the winner survives and enters the next generation.

Explanatory Diagram: Fig. 5 shows the diagrammatic representation of offspring generated by the preceding crossover, mutation, and local search operations. Afterward, the replacement operation is visually illustrated in Fig. 6. O1 is compared with the members in the corresponding parental group, i.e., P1 and P2. P1 is the nearest chromosome to O1, and thus O1 is compared with P1. As P1 has a shorter tour than O1, O1 is discarded. In a similar way, the competitor of O2 is P2. O2 with the tour length 100 surpasses P2 with the tour length 250, and thus O2 takes the place of P2. In the second neighborhood group, O3 has two nearest neighbors P6 and P4 with the largest sharing distance 0.90. The tour length of P6 is worse than that of P4. Therefore, P6 becomes the rival to O3. The outcome is that P6 fails and O3 displaces P6. In a similar

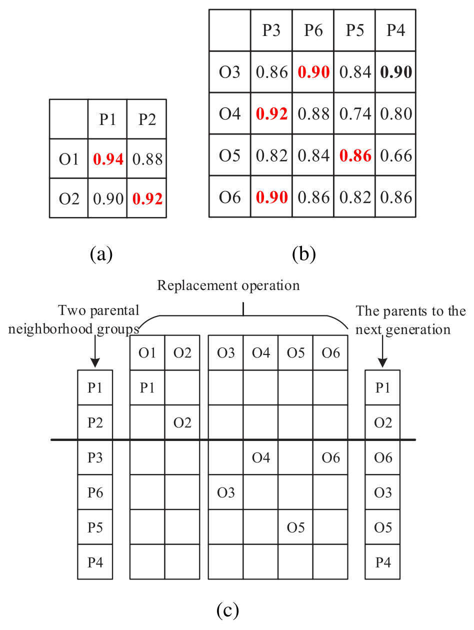

Fig. 6. Diagram of replacement operation. (a) Similarity matrix between the offspring and the parents of the first neighborhood group. (b) Similarity matrix between the offspring and the parents of the second neighborhood group. (c) Choice of the parents to the next generation.

way, finally, members of the second neighborhood group are settled. Thus far, we obtain the parents in the next generation.

H. Elite Selection Approach

When the terminal condition is met, the algorithm stops and provides the final solutions. Most MMOP algorithms simply offer the final *NP* solutions to decision makers. However, decision makers actually do not care about the whole search solution set, as they make choices depending on those representative ones. In consideration of improving the decision maker experience, we downsize the final solutions. Redundant and insufficient solutions are pruned, while the representative ones are finally provided.

A neighborhood group is supposed to cover a subspace and the best member of the group dominates other members. In this context, the best solution in the neighborhood group should be provided. Besides, we also consider the occasion that more than one optimum are included in the same neighborhood group, when these optima locate closely. The specific operations to determine which solutions are provided are described as follows. The shortest length of the final solution set is found out and set to $L_{\min}$. $L_{\min}$ is amplified $(1 + \epsilon)$ to obtain the threshold length $L_{\text{threshold}}$. The threshold ratio ϵ is set to 0.01 in this article. We also create an empty archive to store the provided solutions. Afterward, we go through neighborhood groups to collect representative solutions. For every neighborhood group, we sort the members by the tour length, in a descending order. Before examine them one by one, we first check whether they exist in the archive. If the candidate solution has already existed, we skip it. Otherwise, we examine the candidate solution on three occasions as follows.

1) If the length of the candidate is in accordance with $L_{\min}$. This candidate is added to the archive directly.

Algorithm 5 $\mathbb{S} \leftarrow$ Preserve(*Parent*)

```
1: $L_{\min} \leftarrow$ MinLength($Parent$)
2: $L_{\text{threshold}} \leftarrow L_{\min} \times (1+\epsilon)$
3: $\mathbb{S} = \{\}$
4: for each $NeighborGroup \in Parent$ do
5:     sort($NeighborGroup$.Len, "acs") /*Sort the members in $NeighborGroup$ by length, in a descending order.*/
6:     for $i = 1$ to $|NeighborGroup|$ do
7:         $c \leftarrow NeighborGroup[i]$
8:         if existed($\mathbb{S}$, $c$) then
9:             continue
10:        end if
11:        /* Occasion 1 */
12:        if $c$.Len == $L_{\min}$ then
13:            $\mathbb{S} \leftarrow \mathbb{S} + c$ /*Preserve the chromosome best so far.*/
14:            /* Occasion 2 */
15:        else if $i == 1$ and $c$.Len $\leq L_{\text{threshold}}$ then
16:            $Sh_{\max} \leftarrow 0$
17:            for $s \in \mathbb{S}$ do
18:                if $S(s, c) > Sh_{\max}$ then
19:                    $Sh_{\max} \leftarrow S(s, c)$
20:                end if
21:            end for
22:            if $Sh_{\max} < 0.8$ then
23:                $\mathbb{S} \leftarrow \mathbb{S} + c$ /*Preserve the superior and diverse solution.*/
24:                break
25:            end if
26:            /*Occasion 3 */
27:        else
28:            break
29:        end if
30:    end for
31: end for
```

2) If the candidate solution is the best member of its associated group and its length is greater than $L_{\min}$ but less than or equal to $L_{\text{threshold}}$, then the candidate solution falls in the second occasion. The sharing distance between the candidate solution and members in the archive is calculated one by one. The maximum sharing distance is recorded as $Sh_{\max}$. The candidate solution is distinguished from other members in the archive when the $Sh_{\max}$ is less than $0.8 \times N$ (N denotes the number of cities). When the candidate solution is distinguished or when the archive is empty, the candidate solution is allowed to enter the archive.
3) If the candidate is not qualified for the first two occasions, it indicates that the candidate solution is insufficient and undistinguished and, therefore, the solution is ignored. The overall procedure is concluded in **Algorithm 5**. Note that the proposed elite selection approach is defined on the neighborhoods (or niches), which is inapplicable to some niching strategies without explicit niche partition, such as those in the crowding-based GA (CGA) and sharing-based GA (ShGA) [63].

Explanatory Diagram: To make a clear explanation, suppose that there are three neighborhood groups with $N = 50$. The process of the elite selection approach on the groups is further described in Fig. 7. In the diagram, the three sorted groups with their tour lengths are shown on the left, and the archive in process is shown on the right. From the provided graph, we have the minimum length $L_{\min} = 500$ and the threshold length $L_{\text{threshold}} = 500 \times (1 + 0.01) = 505$. Initially, the archive is empty. In the first neighborhood group, the best chromosome O2 has the minimum length 500, which is less than $L_{\text{threshold}}$. It satisfies the second occasion, and thus O2 is inserted into the archive directly. Then it is O1's turn, as it has the tour length 560 exceeding $L_{\text{threshold}}$ and thus meets the third occasion. Thereafter, O1 is rejected. Then, in the second neighborhood group, O3 first checks its qualification to enter the archive. O3 has the tour length 502 exceeding $L_{\text{threshold}}$, but its sharing distance with the only member in the archive O2 is 0.88 larger than 0.8. Therefore, O3 meets the rejection condition in the second occasion. Then, O4 is refused as it meets the third occasion. As to the third group, O8 has the acceptable length 506 and possesses the allowable sharing distance 0.60 to O2 and, consequently, it is appended to the archive. The rest members O7, O5, and O6 are rejected as they all in the third occasion. In the end, we obtain two superior (i.e., O2 and O8) and distinct solutions from the eight chromosomes.

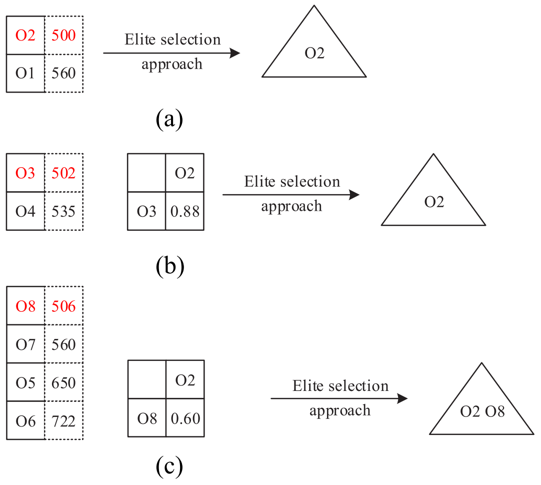

Fig. 7. Diagram of elite selection approach. (a) First neighborhood group. (b) Second neighborhood group. (c) Third neighborhood group.

I. Complexity Analysis

In TSP or MSTSP, evaluating a solution is to sum up the length of edges in the tour represented by the solution. The complexity of the fitness evaluation is thus $O(N)$ for one evaluation. The similarity calculation measures the number of the common edges of two solutions, whose complexity is also $O(N)$. Next, we discuss the computational complexity of each major component of NMA in each generation. For simplicity, the neighborhood size is represented as a fixed value m. The computational overhead of the adaptive neighborhood strategy in **Algorithm 2** is dominated by calculating the sharing distance between each pair of individuals, which is $O(N \times NP^2/m)$. The diversity enhancement strategy in **Algorithm 3** costs $(NP/m) \times O(N \times m) = O(NP \times N)$ to check whether the niche is converged. For the CES update strategy in **Algorithm 4**, its most costly step is to identify the common edges among different persistent leaders. Since there are at most (NP/m) persistent leaders, this strategy is $O(N \times NP/m)$ complex. The reproduction of NMA consumes $O(NP \times N)$ because of the crossover and mutation operations. In the worst case, the selective local search also consumes $O(NP \times N^2)$. The fitness evaluations for the population averagely cost $O(NP \times N)$ in each generation. Finally, the elite selection approach in **Algorithm 5** takes $O(NP \times N \times AN)$ to calculate the distance between solutions in the population and those in the archive, where AN is the archive size. However, this step is only performed at the end of optimization, which can be neglected in calculating the complexity of the entire algorithm. Summarizing the above analysis, the computational complexity of the algorithm in each generation is dominated by the complexity of the local search, i.e., $O(NP \times N^2)$, which is identical to the classical MA using the 2-opt as local search. In addition, Table V in the experiment section compares the real execution time of different algorithms, which indicates that the proposed NMA is the fastest among the peer algorithms.

IV. Experimental Results of MSTSP

A. Experimental Setup

In the experiments, the proposed NMA is compared with four discrete MMOP algorithms. First, NACS [17] and NGA [18] are two previously published MMOP algorithms that have been reviewed in Section II of this article. The parameters of these two algorithms are set according to their corresponding publications [17], [18]. Besides, we implement two basic MMOP algorithms, i.e., CGA [63] and fitness sharing-based GA (ShGA) [63]. They incorporate two niching strategies, i.e., crowding and fitness sharing, which are commonly used in the continuous MMOPs, into GA. In addition, they two measure the distance between two permutations π_i and π_j using $1 - S(\pi_i, \pi_j)$. Their crossover operation and mutation operation are PMX [61] and EM [62], respectively. The GA-related parameters are as follows: the crossover rate $P_c = 0.9$, the mutation rate $P_m = 0.01$, the crowding factor $CF = NP$ for CGA, and the niche radius is set to 0.2 for CGA and ShGA. Besides, in the adaptive neighborhood strategy of our NMA, the neighborhood size m is restricted in the range $[M_{\min} = 4, M_{\max} = 12]$. For all the algorithms, the population size NP is uniformly set to 150, the maximum fitness evaluations (MaxFEs) are listed in Table I, and the terminal condition is the exhaustion of the MaxFEs. The five algorithms are tested on the 25 MSTSTP test instances [18], and more available details about the benchmark suite are available on the website.[1] All algorithms run 50 times independently.

B. Performance Measure

Two evaluation indicators are in accordance with that in the technical report [18]. For the sake of completeness, we give the detailed information below.

1) F_β Measure: F_β is the comprehensive indicator of the precision value P and the recall value R to evaluate the quality of obtained solutions. P is the fraction of the obtained solutions that are optimal solution:

$$P = \frac{TP}{TP + FP} \tag{5}$$

where TP is the number of optimal solutions in returned solutions, and FP is the number of non-optimal solution in the final solution set. R is the fraction of the ground-truth solutions that are successfully located:

$$R = \frac{TP}{TP + FN} \tag{6}$$

where FN is the number of the optimal solutions that the algorithm misses. Actually, the sum of TP and FN is the number of total desired solutions in the benchmark. Based on the values of P and R, F_β [64] is calculated as

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{\beta^2 \times P + R}. \tag{7}$$

Typically, β^2 is set to 1 that attaches the same importance to P and R. However, for the test instance with a large number of optima, it is more important to obtain the most representative ones than to exhaustively locate them all. So that the value of β^2 is set to 0.3 to magnify the effect of precision to evaluate the obtained solutions. In addition, the three indicators (P, R, and F_β) are real values in [0, 1]. Particularly there are two extreme cases: when the algorithm provides the exactly desired solutions in an ideal condition, the F_β score is 1; when the algorithm fails to locate any satisfactory solution, the F_β score is 0.

2) Diversity Indicator: Diversity indicator (DI) is another essential measure for evaluating the algorithm performance. When algorithms fail to find any desired solutions, their F_β values are all zero. In these cases, DI helps to further differentiate the performance of different algorithms. Inspired by the evolutionary multi-objective optimization area where the algorithm also provides a solution set that needs to be evaluated [21], DI measures the diversity of the finally provided solution set $\mathbb{S}$ based on the convergence of the solutions toward different optimal solutions in the ground-truth set $\mathbb{P}$. To be specific, DI is defined based on the average maximum similarity between the obtained solutions and the ground-truth solutions, which is calculated as

$$\mathrm{DI}(\mathbb{P}, \mathbb{S}) = \frac{\sum_{i=1}^{|\mathbb{P}|} \max S(p_i, s_j)\big|_{j=1,\ldots,|\mathbb{S}|}}{N \times |\mathbb{P}|} \tag{8}$$

where p_i is the ith permutation of $\mathbb{P}$, while s_j is the jth permutation of $\mathbb{S}$. $\max S(p_i, s_j)|_{j=1,\ldots,|\mathbb{S}|}$ measures the maximum sharing distance between the permutation p_i and every permutation in $\mathbb{S}$ using Eq. (2).

C. Comparisons With Discrete Multi-Solution Optimizers

1) F_β Score: F_β score measures the solution quality. A high score indicates a good solution quality. We conduct simulation

TABLE I
MaxFEs Applied for Two Ranges of MSTSP Instances

Two Ranges of MSTSP instances	MaxFEs
MSTSP1 - MSTSP12	6.00E+04
MSTSP13 - MSTSP25	1.20E+06

[1]https://github.com/GnauhGnit/MSTSP

TABLE II
F_β SCORE ON 25 MSTSPs OVER 50 RUNS. THE PROPOSED ALGORITHM IS COMPARED WITH FOUR OTHER DISCRETE MMOP ALGORITHMS. THE COMPARISON RESULTS WITH WILCOXON RANK-SUM TEST AT $\alpha = 0.05$ ARE LISTED ON THE RIGHT, AND THE CORRESPONDING SUMMARY RESULTS ARE SHOWN IN THE LAST ROW

F_β	NACS		NGA		CGA		ShGA		NMA
MSTSP1	0.684	−	0.973	−	0.024	−	0.026	−	**1.000**
MSTSP2	0.804	−	0.959	−	0.030	−	0.034	−	**1.000**
MSTSP3	0.497	−	0.936	−	0.078	−	0.110	−	**1.000**
MSTSP4	0.724	−	0.932	−	0.034	−	0.034	−	**1.000**
MSTSP5	0.989	≈	0.846	−	0.017	−	0.017	−	**1.000**
MSTSP6	0.643	−	0.877	−	0.034	−	0.034	−	**1.000**
MSTSP7	0.125	−	0.769	−	0.261	−	0.435	−	**0.923**
MSTSP8	0.137	−	0.578	−	0.337	−	0.700	−	**0.772**
MSTSP9	0.768	−	0.974	−	0.034	−	0.034	−	**1.000**
MSTSP10	0.813	−	0.969	−	0.034	−	0.034	−	**1.000**
MSTSP11	0.459	−	0.949	−	0.072	−	0.118	−	**1.000**
MSTSP12	0.090	−	0.331	−	0.275	−	**0.919**	+	0.535
MSTSP13	0.025	−	0.096	−	0.255	−	0.003	−	**0.611**
MSTSP14	0.087	−	0.172	−	0.090	−	0.000	−	**0.883**
MSTSP15	0.004	−	0.416	−	0.219	−	0.003	−	**0.732**
MSTSP16	0.000	−	0.054	−	0.211	−	0.000	−	**0.554**
MSTSP17	0.000	−	0.044	−	0.044	−	0.000	−	**0.605**
MSTSP18	0.000	−	0.031	−	0.062	−	0.000	−	**0.571**
MSTSP19	0.000	−	0.007	−	0.040	−	0.000	−	**0.168**
MSTSP20	0.000	−	0.000	−	0.015	−	0.000	−	**0.165**
MSTSP21	0.012	≈	0.000	−	0.001	≈	0.000	−	**0.023**
MSTSP22	0.000	−	0.000	−	0.000	≈	0.000	−	**0.013**
MSTSP23	0.000	−	0.000	−	0.000	−	0.000	−	**0.016**
MSTSP24	0.000	−	0.000	−	0.009	≈	0.000	−	**0.010**
MSTSP25	0.000	≈	0.000	≈	0.000	≈	0.000	≈	**0.002**
+/≈/−	0/3/22		0/1/24		0/4/21		1/1/23		

TABLE III
P, R, AND F_β OF FOUR COMPARED ALGORITHMS AND NMA ON MSTSP10 AND MSTSP12 ARE LISTED

Instance	Value	NACS	NGA	CGA	ShGA	NMA
MSTSP10	P	**1.000**	**1.000**	0.027	0.027	**1.000**
	R	0.500	0.428	**1.000**	**1.000**	**1.000**
	F_β	0.813	0.969	0.034	0.034	**1.000**
MSTSP12	P	0.770	0.557	0.295	0.984	**1.000**
	R	0.023	0.192	0.226	**0.753**	0.211
	F_β	0.090	0.331	0.275	**0.919**	0.535

TABLE IV
PROPOSED ALGORITHM IS COMPARED WITH FOUR DISCRETE MMOP ALGORITHMS IN TERMS OF DI. THE SUMMARY COMPARISON RESULTS WITH WILCOXON RANK-SUM TEST AT α=0.05 ARE LISTED

Wilcoxon Rank-Sum Test (+/≈/−)	DI
NACS vs. NMA	0/2/23
NGA vs. NMA	0/0/25
CGA vs. NMA	7/12/6
ShGA vs. NMA	3/9/13

experiments with four compared algorithms and NMA on 25 MSTSPs concerning F_β score. In order to further validate the performance, a Wilcoxon rank-sum test is also applied. The notations "+", "≈", and "−" indicate that the algorithm is significantly better than, similar to, and significantly worse than the proposed NMA, respectively. Note that the notations of the significant results in the remaining parts are shown in the same way. The numerical results along with the significant results are shown in Table II. The first row shows the algorithm names, and the numerical results are reported in the following rows, where the corresponding significant results are placed on the right. In addition, summary results are presented in the last row. As seen in the table, NMA achieves significantly better results on most MSTSPs (at least 21 out of 25), especially on simple MSTSPs and geometry MSTSPs. Two modified basic MMOP algorithms, CGA and ShGA, perform poorly on most MSTSPs, except for MSTSP12 with ShGA. Particularly, MSTSP12 holds the largest number of optima, i.e., 196. These F_β values of the five algorithms fall rapidly along with city size increasing, but NMA performs slightly better among them.

To make a further investigation of the performance concerning F_β, considering that F_β is evaluated with P and R in Eq. (7), we display values of P, R, and F_β on MSTSP10 and MSTSP12 in Table III. For MSTSP10, NACS and NGA reach 1.000 precision values but less than or equal to 0.500 recall values, which still results in high F_β values (greater than 0.8); CGA and ShGA obtain small precision values (smaller than 0.03) but the highest 1.000 recall values, which leads to small F_β values (lower than 0.04); the precision and recall values of NMA are both 1.000, therefore, F_β is 1.000 according to Eq. (7). The observations indicate that although CGA and ShGA have advantages over covering more desired solutions, they may obtain poor F_β values with the provision of a quite large population of solutions. For MSTSP12 with 196 optima, ShGA gets a high precision value (greater than 0.9) and a moderate recall value (greater than 0.7), while NMA obtains the highest precision value but a small recall value (smaller than 0.3). The outcome shows that ShGA is better than the other MMOP algorithms with respect of F_β score. NACS, NGA, and NMA provide the best solutions of each niches, while CGA and ShGA offer with overall obtained solutions that increase the possibility of obtaining spatial near optima. But for such instance with massive optima, NMA can still obtain good performance with a high precision value by providing a few and distinct obtained solutions.

The experimental results validate that NMA outperforms the other compared algorithms concerning F_β score, but they all encounter difficulty in dealing with MSTSPs with a large city size. Furthermore, NMA can achieve a high precision value for instances with various optima size to further obtain good F_β value.

2) DI: DI evaluates the solution diversity. Higher DI means a better diversity. We illustrate with a quick overview of DI score by plotting pseudocolor image in Fig. 8. The figure indicates that, relatively speaking, CGA and NMA have a good diversity on overall MSTSP instances; ShGA performs well on simple and geometry MSTSPs but behaves badly on composite MSTSPs; NACS and NGA present unsatisfactory results on overall problem instances. The significant test is also conducted and the results are reported in Table IV. As seen from the table, NMA is significantly better than NACS and NGA, 23 out of 25 and 25 out of 25, respectively. In contrast, CGA and ShGA have a good performance, and possess similar significant results when compared with NMA, 12 out of 25 and 9 out of 25, respectively. These results suggest that CGA and ShGA can obtain a high DI as they provide a large archive, while NMA can also achieve the similar performance by providing representative optima.

3) Execution Time: The execution time is another essential metric for assessing algorithms. The time overhead for tackling

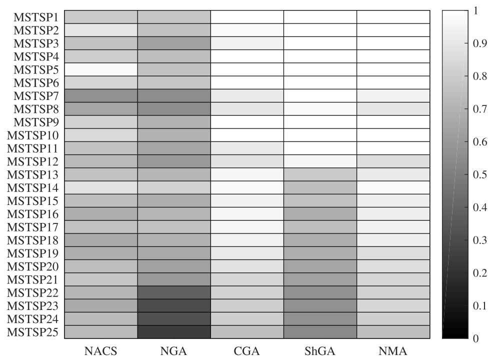

Fig. 8. Pseudocolor plot of 25 MSTSPs in terms of DI with four compared algorithms and the proposed algorithm.

NP-hard problems rises dramatically along with the increasing city size. The average execution time of the algorithms is summarized in Table V, where the best results are displayed in bold. In the table, over the overall test instances, NMA or NACS always performs best. Taking a close look, NACS ranks the first on simple MSTSPs and geometry MSTSPs, followed by NMA. When it comes to composite MSTSPs, NACS consumes more time than NMA does, and thus NMA is obviously the winner. For MSTSP1 with nine cities, NACS, NMA, NGA, CGA, and ShGA averagely consume 0.20 s, 0.37 s, 1.66 s, 12.57 s, and 33.17 s, respectively, in an ascending order. As to MSTSP25 with 66 cities, the time consuming order is slightly changed. NMA, NACS, NGA, CGA, and ShGA averagely consume 23.08 s, 642.49 s, 957.32 s, 6280.10 s, and 14103.00 s, respectively. The running time of NGA, CGA, and ShGA increases rapidly as the city size increases. NMA is slightly affected by the city size.

TABLE V
AVERAGE EXECUTION TIME ON 25 MSTSPS. THE OVERALL AVERAGE RUNNING TIME IS LISTED IN THE LAST ROW

Avg. Time (s)	NACS	NGA	CGA	ShGA	NMA
MMTSP1	**0.20**	1.66	12.57	33.17	0.37
MMTSP2	**0.28**	2.01	15.09	39.66	0.33
MMTSP3	**0.29**	1.92	14.72	40.71	0.75
MMTSP4	**0.22**	2.25	17.16	45.65	0.33
MMTSP5	**0.24**	2.74	19.62	51.96	0.32
MMTSP6	**0.23**	2.69	19.31	52.14	0.30
MMTSP7	**0.17**	2.06	14.82	42.93	0.82
MMTSP8	**0.27**	2.64	19.67	51.65	0.58
MMTSP9	**0.17**	1.91	14.93	39.10	0.40
MMTSP10	**0.17**	1.93	14.97	40.82	0.38
MMTSP11	**0.22**	2.00	13.81	42.13	0.49
MMTSP12	**0.50**	4.01	25.29	63.23	0.66
MMTSP13	55.46	192.33	1099.20	3886.70	**12.39**
MMTSP14	54.50	276.97	1451.90	4251.40	**15.95**
MMTSP15	100.98	104.48	738.52	2715.50	**10.86**
MMTSP16	275.20	262.93	1414.40	4077.60	**14.91**
MMTSP17	217.79	235.72	1580.40	5249.20	**17.28**
MMTSP18	542.46	377.04	2520.80	7374.50	**17.85**
MMTSP19	385.10	405.14	2241.60	6652.50	**19.72**
MMTSP20	492.69	500.81	3237.30	8672.90	**20.24**
MMTSP21	293.42	561.17	3575.00	9149.90	**21.18**
MMTSP22	614.00	706.78	4512.10	10903.00	**23.66**
MMTSP23	1033.10	785.61	5012.10	11876.00	**24.59**
MMTSP24	382.57	766.16	5526.20	10757.00	**24.42**
MMTSP25	642.49	957.32	6280.10	14103.00	**23.08**
Overall	106.36	119.16	724.30	2171.17	**10.07**

TABLE VI
AVERAGE PERFORMANCE EVALUATIONS ON 25 MSTSPS. THE PROPOSED ALGORITHM IS COMPARED WITH THE NICHING EAS EXTENDED FROM THE CONTINUOUS AREA BY USING THE DISCRETIZATION-BASED METHOD AND THE RANDOM KEY-BASED METHOD

Avg. Value	d_dADE	rk_dADE	d_NSDE	rk_NSDE	d_DSDE	rk_DSDE	NMA
F_β	0.145	0.034	0.123	0.027	0.133	0.048	**0.623**
DI	0.836	0.652	0.810	0.612	0.815	0.585	**0.925**
Used Time	1257.772	41.861	143.749	17.455	207.747	11.046	**10.074**
Ω	0.445	0.180	0.413	0.132	0.382	0.138	**0.519**

D. Comparisons With Algorithms Extended From the Continuous Multimodal Optimization Area

The MSTSP is a relatively new area, for which the benchmark was released in 2018 [18]. There are only a few algorithms existed in this area, which are available for comparison. Besides, the most related area to MSTSP is the continuous MMOP. There are many state-of-the-art continuous niching EAs for continuous MMOPs. We choose three state-of-the-art and representative algorithms to perform additional comparisons: the dynamic archive niching DE (dADE) [65], the neighborhood-based speciation DE (NSDE) [19], and the dual-strategy DE (DSDE) with crowding technique [22]. In order to extend the three continuous niching EAs to the discrete domain specifically for MSTSPs, we adopt two methods: 1) a discretization-based method and 2) a random key-based method [66], [67].

For the discretization-based method, the following modifications are made to the algorithms.

1) *Solution Encoding:* In DE, the candidate solution is encoded into a real number parameter vector. The encoding method is changed into a permutation-based encoding scheme as NMA.
2) *Evolution Operators:* Using DE as the baseline, the crossover and mutation operators in the algorithms are unsuitable to deal with MSTSPs. Thus, we substitute the PMX [61] and the EM [62] for the crossover and mutation operators to make the algorithms available for MSTSPs.
3) *Distance Measure:* The Euclidean distance measure used in the compared algorithms is replaced by a measure defined on our proposed sharing distance (i.e., $1 - S(\pi_i, \pi_j)$).

The random key-based method represents the solution with real values and maps the solution to the literal space by numeric sorting. Subsequently, it enables the algorithm to evolve population in the continuous space and to evaluate solutions in the literal space. Incorporating the random key-based method has the advantages that it allows to make the minimal adjustments to the continuous niching EAs and ensures candidate solutions feasible. The disadvantage is also prominent: the represented solution in the evolution space and the mapped solution in the literal space are loosely coupled, so that the effect of the niching techniques (to identify multiple diverse solutions) is weakened.

The competitors are tested on 25 MSTSPs with the same parameter settings proposed in Section IV-A. Table VI provides the comparison results. For the sake of brevity, the names of the tailored EAs prefixed with "d_" and "rk_" indicate the usages of the discretization-based method and the random key-based method, respectively. It can be seen that NMA possesses superior performance over the other compared algorithms in terms of F_β, DI, and execution time. In addition, the Ω in the last row of Table VI is a supplementary indicator for the solution quality F_β. The comparison algorithms directly return the final solution set including some unnecessary inferior solutions, which degrade the F_β value. For further performance investigation, the Ω divides the number of obtained optima (presented by the intersection of the finally provided solution set and the optimal solution set $\mathbb{P}$) by the number of known optima, i.e., $\Omega = |\mathbb{P} \cap \mathbb{S}|/\mathbb{P}$. The data of Ω show that NMA can find more optimal solutions than the competitors do. Additionally, the pair comparisons between the extended algorithms show that the discretization-based method outperforms the random key-based method except for the elapsed time.

The above results indicate that although the existed continuous niching EAs can be extended into discrete ones for MSTSPs, the tailored algorithms may endure very low performance. Neither the direct discretization nor the mapping method can achieve a satisfactory result. This also gives the reason why more research efforts should be paid specifically into the discrete multi-solution optimization domain.

E. Investigations on Parameters and Subcomponents of NMA

In Section SII in the supplementary material, we provide experimental investigations on the parameter settings as well as the effects of different subcomponents in NMA. Due to the page limit, here, we provide the summary of the investigations only. Section SII-A in the supplementary material suggests P_c between [0.9, 1.0], P_m smaller than 0.1, and NP between [100, 250]. The Section SII-B in the supplementary material reports the effects of subcomponents: 1) the proposed adaptive neighborhood strategy helps NMA to maintain good performance in terms of solution quality and diversity on different MSTSP instances; 2) the selective local search improves the search efficiency by avoiding unnecessary exploitation; and 3) the elite selection strategy finally returns representative solutions without impairing the solution set quality.

V. Experimental Results on the TSPLIB

TSPLIB is a well-known TSP library, which includes 112 TSP instances. The corresponding known optimal solution are published on the website.[2] However, the publication of TSP library only considers the global optimality but ignores the multi-solution characteristic. In order to mine the latent characteristic of multi-solution, we employ NACS, NGA, and NMA to solve eight TSP instances. The MaxFEs is set to $50000 \times N$ (N denotes the number of cities). These algorithms will provide distinct solutions. The solutions are identified as optima when their lengths are shorter than a certain threshold length $(1+\sigma) \times$ (tour length). The σ is set to 0.1 here. Average numbers of qualified solutions are presented in Table VII. The best results are boldface. NMA always ranks the first, which indicates that NMA obtains more satisfactory solutions than that obtained by the other two compared algorithms.

In order to show the multi-solution characteristic more clearly, we choose four representative solutions (i.e., the best solution and the second best solution, as well as their corresponding the most dissimilar ones) obtained by NMA and further draw them in Fig. 9. In each subfigure, the best solution and its corresponding dissimilar solution locate in the top-left and top-right, while the second best solution and its corresponding dissimilar one are in the bottom-left and bottom-right. In Fig. 9(a), we first compare tours in the left column, which shows the best solution with a length of 436 and the second best solution with a length of 437 for eil51. They have equally good tour length, but possess very different topologies. Moreover, horizontal comparison indicates that even the very different tours can achieve a similar performance with a small gap. The observation in Fig. 9(b) is similar to the observation obtained by Fig. 9(a). Although NMA fails to obtain the known optimal solutions (for eil51, optimal length/best obtained length is 426/436; for st70, optimal length/ best obtained length is 675/683), it provides good and diverse solutions instead. Even so, the observations are in agreement with the multi-solution characteristic of TSP.

TABLE VII
Number of Satisfactory Solutions Obtained by NACS, NGA, and NMA Are Listed, Respectively

No. Solution	NACS	NGA	NMA
eil51	1.18	1.14	**5.84**
berlin52	1.28	1.02	**4.54**
st70	1.26	1.02	**6.00**
pr76	1.40	0.84	**5.12**
kroA100	1.08	0.90	**5.36**
lin105	1.02	0.06	**4.54**
Overall	1.20	0.83	**5.23**

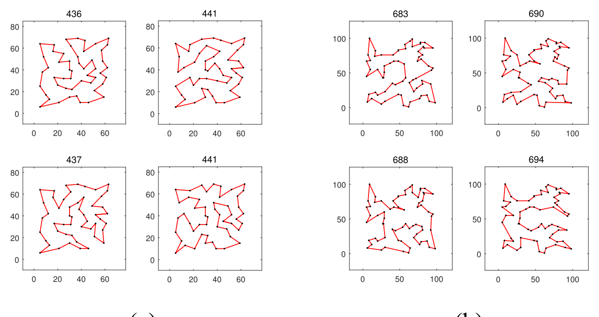

Fig. 9. Diagram of representative solutions obtained by NMA. The subgraphs on the top-left and on the bottom-left are the best and the second best tours. Their most dissimilar tours are drawn on their right. (a) TSP instance eil51. (b) TSP instance st70.

VI. Conclusion

This article presented an NMA to solve MSTSPs. The neighborhood niching strategy and MA are effectively employed to accomplish the target of locating multiple solutions of MSTSP. The neighborhood niching strategy maintains diverse candidates during the search, and MA promises the search ability. To further enhance the performance, four augmentation methods are developed. The adaptive neighborhood strategy allocates the search effort with an adjusting neighborhood size, and further balances the exploration and exploitation. The diversity enhancement approach perturbs converged

[2]http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.htmls

group members to alleviate the problem of being trapped in local optima. The CEA strategies provide additional guidance to evolutionary operations and avert unnecessary exploitation of local search. The selective local search biases the local search to potential candidate solutions to maximize fitness evaluations utilization.

To validate the efficiency and effectiveness of the proposed algorithm, the proposed NMA is compared with two existing discrete MMOP algorithms and two modified niching MMOP algorithms. NMA outperforms the four compared algorithms in terms of two evaluation metrics: F_β and DI. Meanwhile, the comparison of running time demonstrates that NMA is less influenced by the city size, which consumes the least running time in average when compared to the competitors. Moreover, the effect of each subcomponent of NMA is thoroughly tested and validated in our investigation experiments. The additional experiments on the TSPLIB also show that NMA obtains more diverse and better solutions than the other algorithms do. Much work remains to be done in the future. There is still room to improve the performance of NMA to solve instances with large city sizes. In addition, some other niching algorithms can be adapted and improved to settle discrete multi-solution problems. Besides, the multi-solution (or multimodal) characteristic, which is always omitted in some global optimization problems, can be further exploited and utilized with the help of MMOP techniques.

References

[1] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*. New York, NY, USA: Springer, 2007, pp. 1–15.

[2] K. Ilavarasi and K. S. Joseph, "Variants of travelling salesman problem: A survey," in *Proc. Int. Conf. Inf. Commun. Embedded Syst.*, Feb. 2014, pp. 1–7.

[3] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.

[4] J.-H. Wang, Y. Zhou, Y. Wang, J. Zhang, C. L. P. Chen, and Z.-B. Zheng, "Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, instances, and algorithms," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 582–594, Mar. 2016.

[5] R. Roberti and M. Wen, "The electric traveling salesman problem with time windows," *Transp. Res. E Logist. Transp. Rev.*, vol. 89, pp. 32–52, May 2016.

[6] A. Zheng, Y. Yuan, J.-T. Zhou, Y.-F. Guo, H.-T. Yang, and O. C. Au, "Adaptive block coding order for intra prediction in HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 11, pp. 2152–2158, Nov. 2016.

[7] M. J. Arnesen, M. Gjestvang, X. Wang, K. Fagerholt, K. Thun, and J. G. Rakke, "A traveling salesman problem with pickups and deliveries, time windows and draft limits: Case study from chemical shipping," *Comput. Oper. Res.*, vol. 77, pp. 20–31, Jan. 2017.

[8] T. Lust and J. Teghem, "The multiobjective traveling salesman problem: A survey and a new approach," in *Proc. Adv. Multi Objective Nat. Inspired Comput.*, 2010, pp. 119–141.

[9] L.-J. Ke, Q.-F. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.

[10] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 5, pp. 682–691, Sep. 2012.

[11] A. Khan, E. Yanmaz, and B. Rinner, "Information exchange and decision making in micro aerial vehicle networks for cooperative search," *IEEE Trans. Control Netw. Syst.*, vol. 2, no. 4, pp. 335–347, Dec. 2015.

[12] J. Faigl and P. Váňa, "Surveillance planning with Bézier curves," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 750–757, Apr. 2018.

[13] K. J. Obermeyer, P. Oberlin, and S. Darbha, "Sampling-based path planning for a visual reconnaissance unmanned air vehicle," *J. Guid. Control Dyn.*, vol. 35, no. 2, pp. 619–631, 2012.

[14] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.

[15] S. Ronald, "Finding multiple solutions with an evolutionary algorithm," in *Proc. IEEE Int. Conf. Evol. Comput.*, vol. 2, Nov. 1995, pp. 641–646.

[16] D. Angus, "Niching for population-based ant colony optimization," in *Proc. IEEE Int. Conf. e-Sci. Grid Comput.*, Dec. 2006, p. 115.

[17] X.-C. Han, H.-W. Ke, Y.-J. Gong, Y. Lin, W.-L. Liu, and J. Zhang, "Multimodal optimization of traveling salesman problem: A niching ant colony system," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2018, pp. 87–88.

[18] T. Huang, Y.-J. Gong, and J. Zhang, "Seeking multiple solutions of combinatorial optimization problems: A proof of principle study," in *Proc. IEEE Symp. Comput. Intell.*, 2018, pp. 1212–1218.

[19] B.-Y. Qu, P. N. Suganthan, and J.-J. Liang, "Differential evolution with neighborhood mutation for multimodal optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 5, pp. 601–614, Oct. 2012.

[20] Y.-J. Gong, J. Zhang, and Y. Zhou, "Learning multimodal parameters: A bare-bones niching differential evolution approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2944–2959, Jul. 2018.

[21] S. Biswas, S. Kundu, and S. Das, "Inducing niching behavior in differential evolution through local information sharing," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 246–263, Apr. 2015.

[22] Z.-J. Wang *et al.*, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.

[23] X.-D. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[24] C. Shi, Q.-D. Qin, W. Zhou, Y.-H. Shi, and Q.-Y. Zhang, "Multimodal optimization using particle swarm optimization algorithms: CEC 2015 competition on single objective multi-niche optimization," in *Proc. IEEE Congr. Evol. Comput.*, May 2015, pp. 1075–1082.

[25] M. Preuss, "Niching the CMA-ES via nearest-better clustering," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2010, pp. 1711–1718.

[26] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann, "Detecting funnel structures by means of exploratory landscape analysis," in *Proc. Genet. Evol. Comput. Conf. Companion*, 2015, pp. 265–272.

[27] A. Ahrari, K. Deb, and M. Preuss, "Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations," *Evol. Comput.*, vol. 25, no. 3, pp. 439–471, 2017.

[28] Q. Yang *et al.*, "Adaptive multimodal continuous ant colony optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 2, pp. 191–205, Apr. 2017.

[29] X.-D. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking multiple solutions: An updated survey on niching methods and their applications," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 518–538, Aug. 2017.

[30] Y.-X. Liu *et al.*, "Solving NP-hard problems with physarum-based ant colony system," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 1, pp. 108–120, Jan./Feb. 2017.

[31] R.-W. Gan, Q.-S. Guo, H.-Y. Chang, and Y. Yi, "Improved ant colony optimization algorithm for the traveling salesman problems," *J. Syst. Eng. Electron.*, vol. 21, no. 2, pp. 329–333, Apr. 2010.

[32] M. Mavrovouniotis, F. M. Müller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1743–1756, Jul. 2017.

[33] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-Opt algorithms for traveling salesman problem," *Appl. Soft Comput.*, vol. 30, pp. 484–490, May 2015.

[34] A. F. El-Samak and W. Ashour, "Optimization of traveling salesman problem using affinity propagation clustering and genetic algorithm," *J. AI Intell. Soft Comput. Res.*, vol. 5, no. 4, pp. 239–245, 2015.

[35] Y. Wang, H.-X. Li, G. G. Yen, and W. Song, "MOMMOP: Multiobjective optimization for locating multiple optimal solutions of multimodal optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 830–843, Apr. 2015.

[36] S. W. Mahfoud, "Crowding and preselection revisited," in *Proc. Parallel Problem Solving Nat.*, 1992, pp. 27–34.

[37] O. J. Mengshoel and D. E. Goldberg, "Probabilistic crowding: Deterministic crowding with probabilistic replacement," in *Proc. Genet. Evol. Comput. Conf.*, 1999, pp. 409–416.

[38] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 207–234, Sep. 2002.

[39] W.-F. Gao, G. G. Yen, and S.-Y. Liu, "A cluster-based differential evolution with self-adaptive strategy for multimodal optimization," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1314–1327, Aug. 2014.

[40] Q. Yang, W.-N. Chen, Y. Li, C. P. Chen, X.-M. Xu, and J. Zhang, "Multimodal estimation of distribution algorithms," *IEEE Trans. Cybern.*, vol. 47, no. 3, pp. 636–650, Mar. 2017.

[41] Y.-S. Ong, M. H. Lim, and X.-S. Chen, "Memetic computation—Past, present & future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.

[42] X.-S. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct. 2011.

[43] A. Gupta and Y.-S. Ong, *Memetic Computation: The Mainspring of Knowledge Transfer in a Data-Driven Optimization Era*. New York, NY, USA: Springer, 2018.
[44] L. Feng, A. Gupta, and Y.-S. Ong, "Compressed representation for higher-level MEME space evolution: A case study on big knapsack problems," *Memetic Comput.*, vol. 11, no. 1, pp. 3–17, 2017.
[45] Q. H. Nguyen, Y.-S. Ong, and M. H. Lim, "A probabilistic memetic framework," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 604–623, Jun. 2009.
[46] J.-Y. Lin and Y.-P. Chen, "Analysis on the collaboration between global search and local search in memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 608–623, Oct. 2011.
[47] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in *Proc. Conf. Genet. Evol. Comput.*, 2000, pp. 987–994.
[48] P. Merz and B. Freisleben, "Memetic algorithms for the traveling salesman problem," *Complex Syst.*, vol. 13, no. 4, pp. 297–346, 2001.
[49] L. T. Kóczy, P. Földesi, and B. Tüű-Szabó, "A discrete bacterial memetic evolutionary algorithm for the traveling salesman problem," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3261–3267.
[50] H. D. Nguyen, I. Yoshihara, K. Yamamori, and M. Yasunaga, "Implementation of an effective hybrid GA for large-scale traveling salesman problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 92–99, Feb. 2007.
[51] H. Ismkhan, "Effective heuristics for ant colony optimization to handle large-scale problems," *Swarm Evol. Comput.*, vol. 32, pp. 140–149, Feb. 2017.
[52] X.-Y. Chen, P. Zhang, G.-L. Du, and F. Li, "Ant colony optimization based memetic algorithm to solve bi-objective multiple traveling salesmen problem for multi-robot systems," *IEEE Access*, vol. 6, pp. 21745–21757, 2018.
[53] K. Deb and S. Gulati, "Design of truss-structures for minimum weight using genetic algorithms," *Finite Elements Anal. Design*, vol. 37, no. 5, pp. 447–465, 2001.
[54] K.-C. Wong, K.-S. Leung, and M.-H. Wong, "Protein structure prediction on a lattice model via multimodal optimization techniques," in *Proc. ACM Genet. Evol. Comput. Conf. Companion*, 2010, pp. 155–162.
[55] E. Pérez, F. Herrera, and C. Hernández, "Finding multiple solutions in job shop scheduling by niching genetic algorithms," *J. Intell. Manuf.*, vol. 14, nos. 3–4, pp. 323–339, Jun. 2003.
[56] E. Pérez, M. Posada, and F. Herrera, "Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling," *J. Intell. Manuf.*, vol. 23, no. 3, pp. 341–356, 2012.
[57] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, 1991.
[58] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, Sep. 2005, pp. 1785–1791.
[59] J.-Q. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[60] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 590–605, Aug. 2016.
[61] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proc. Int. Conf. Genet. Algorithms Appl.*, vol. 154, 1985, pp. 154–159.
[62] W. Banzhaf, "The 'molecular' traveling salesman," *Biol. Cybern.*, vol. 64, no. 1, pp. 7–14, 1990.
[63] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, 2004, pp. 1382–1389.
[64] P. Flach and M. Kull, "Precision-recall-gain curves: PR analysis done right," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 838–846.
[65] M. G. Epitropakis, X. Li, and E. K. Burke, "A dynamic archive niching differential evolution algorithm for multimodal optimization," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 79–86.
[66] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, 1994.
[67] F. Samanlioglu, M. B. Kurz, W. G. Ferrell, and S. Tangudu, "A hybrid random-key genetic algorithm for a symmetric travelling salesman problem," *Int. J. Oper. Res.*, vol. 2, no. 1, pp. 47–63, 2007.

Ting Huang is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China.

Her current research interests include evolutionary computation, swarm intelligence, multi-solution optimization, and their real-world applications.

Yue-Jiao Gong (S'10–M'15–SM'19) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

She is currently a Full Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation, swarm intelligence, and their applications to intelligent transportation and smart city scheduling. She has published over 80 papers, including over 30 IEEE TRANSACTIONS papers, in the above areas.

Sam Kwong (M'93–SM'04–F'14) received the B.Sc. degree in electrical engineering from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from Fernuniversitaet, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with Control Data Canada, Mississauga, ON, Canada. He later joined Bell Northern Research Canada, Ottawa, ON, Canada, as a Member of Scientific Staff, and the City University of Hong Kong, Hong Kong, as a Lecturer with the Department of Electronic Engineering, in 1990, where he is currently a Chair Professor with the Department of Computer Science. His current research interests include video coding, pattern recognition, and evolutionary algorithms.

Dr. Kwong was elevated to an IEEE Fellow for his contributions on optimization techniques for cybernetics and video coding in 2014. He is also appointed as an IEEE Distinguished Lecturer of the IEEE SMC Society in 2017. He is currently the Vice-President of Cybernetics with the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS and also serves as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and the *Journal of Information Science*.

Hua Wang received the Ph.D. degree from the University of Southern Queensland, Toowoomba, QLD, Australia.

He was a Professor with the University of Southern Queensland. He is currently a full time Professor with Victoria University, Melbourne, VIC, Australia. As a Chief Investigator, three Australian Research Council Discovery grants have been awarded since 2006, and 200 peer-reviewed scholar papers have been published. Six Ph.D. students have already graduated under his principal supervision. He has over ten years teaching and working experience in *Applied Informatics* with both enterprise and university. He has expertise in electronic commerce, business process modeling, and enterprise architecture.

Jun Zhang (F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high-performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.