Seeking Multiple Solutions of Combinatorial Optimization Problems: A Proof of Principle Study

Ting Huang, Yue-Jiao Gong and Jun Zhang
Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information
School of Computer Science and Engineering
South China University of Technology, Guangzhou 510006, China
gongyuejiao@gmail.com; junzhang@ieee.org

***Abstract*—Problems with multiple optimal solutions widely exist in the real world. In some applications, it is required to locate multiple optima. However, most studies are dedicated to the continuous multi-solution optimization, while few works contribute to the discrete multi-solution optimization. To promote the multi-solution research in the discrete area, we design a benchmark test suite for multi-solution traveling salesman problems and propose two evaluation indicators. Further, in order to solve the problems, the genetic algorithm is incorporated with a niching technique defined in the discrete space. The proposed algorithm is compared with an existing algorithm. Experimental results demonstrate that the proposed algorithm outperforms the compared algorithm concerning the quality and diversity of obtained solutions.**

Keywords—multi-solution traveling salesman problem, multimodal optimization, genetic algorithm, neighborhood-based niching strategy

I. INTRODUCTION

Combinatorial Optimization Problems (COPs) are a class of discrete optimization problems that need to find an optimal solution in a finite solution space according to some selection criteria. Many COPs are known to be NP-hard, such as the Traveling Salesman Problem (TSP). It is quite difficult to solve them by deterministic algorithms, especially when the solution space is large. In the past decades, many evolutionary algorithms (EAs) have been successfully developed to solve COPs [1-4], owing to their non-deterministic search characteristic and powerful global optimization ability. However, most studies focus on locating one single optimal solution. But many practical applications require to identify more than one optimal solution, such as the navigation system and the robot path planning. With a diverse solution set, the decision makers can choose the most proper one according to their preferences, or they can have alternatives in case of some unexpected situations.

The area of finding multiple optima for a single problem is generally known as multimodal optimization (MMO) or multi-solution optimization, which has undergone intensive studies in the past few years [5-8]. However, most studies are conducted on the continuous MMO, whereas seldom research is devoted to the discrete MMO. To extend the MMO to the discrete optimization area, there are three main issues required to be considered: (1) the baseline algorithms that are suitable to deal with discrete MMO problems; (2) the techniques to maintain multiple potential solutions in the discrete problem space; (3) an appropriate benchmark test suite. The first two are mainly considered by optimizer developers, while the third one is a crucial factor for evaluating the algorithm performance. In this paper, we conduct a proof of principle study for addressing these three issues, which are briefly introduced as follows.

The first issue regards the choice of an appropriate baseline algorithm for encoding and evolving solutions in the discrete problem space. Some EAs, such as Differential Evolution [9], Particle Swarm Optimization [10], and I-Ching divination EA [11, 12], use floating-point encoding by nature, and hence they cannot solve discrete problems directly. Other EAs, such as Genetic Algorithm (GA) [13, 14] and Ant Colony Optimization [15], are more suitable to use an integer encoding scheme and then work in the discrete or combinatorial space. Therefore, in this work, we choose GA as our baseline algorithm. However, GA cannot solve discrete MMO problems directly because it will converge towards an optimum eventually. The convergence is caused by the diversity loss.

Thus, the second issue focuses on preserving the population diversity, which is crucial for the MMO. In the continuous MMO, the primary methods to preserve population diversity are niching techniques, such as crowding [16, 17], speciation [18], and neighborhood strategy [5]. Niching techniques limit solutions evolving within the local space, which avoids the global convergence. Along this light of consideration, we can incorporate the niching strategies into a suitable baseline algorithm to obtain diverse candidates simultaneously. Specifically, we develop a neighborhood-based niching strategy in the discrete space and then incorporate the strategy into GA to deal with discrete MMO problems.

The third issue concerns the selection of the benchmark suite of discrete MMO problems. Although the multi-solution optimization of COPs is critical for many practical applications, currently this area has received seldom attention and lacked a standard test suite to evaluate the related algorithms. The lack of an appropriate benchmark test suite may hinder the research and development for the discrete MMO area. To promote the development of this area, in this paper, we develop a benchmark test suite for the combinatorial MMO. Because TSP is the most popular and representative COP, we take it as an example and design a set of multi-solution TSP instances (MSTSPs). To our best knowledge, there are several early MSTSP instances have been reported [19, 20]. However, these instances either contain too many optimal tours (e.g., 16 cities with 938 optima) or possess a small number of cities (fewer than 10

This work was supported by the National Natural Science Foundation of China under Grants. 61502542, 61332002, and U1701267, and was also supported by the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing.

978-1-5386-9276-9/18/$31.00 ©2018 IEEE

cities), which are inappropriate and insufficient to be utilized as a benchmark test suite. In this paper, we propose a benchmark suite with 25 MSTSP instances that are classified into three categories, i.e., the simple MSTSPs, the geometry MSTSPs, and the composite MSTSPs. The number of cities ranges from 9 to 66, and the number of optima scales from 2 to 196.

The rest of this paper is organized as follows. Section II formulates TSP. Section III describes the proposed 25 MSTSPs. The proposed algorithm is presented in Section IV. Section V defines the experimental settings and reports experimental results. Finally, the conclusion is drawn in Section VI.

II. Traveling Salesman Problem

Given a number of cities and the distance information between each pair of them, a salesman visits each city only once to construct a Hamilton path. The target of TSP is to find the shortest Hamilton path. Mathematically, consider a graph $G = (V, E)$, where $V = \{1, 2, 3,\ldots, N\}$ is a set of cities (denoted by the indices), and $E = \{(i, j) \mid i, j \in N, i \neq j\}$ is a set of edges indicating the connection between cities i and j. Each connection (i, j) has a weight value d_{ij} to measure the distance between the two cities. Following the edge weight type EUC_2D defined in [21], we round the distance to the nearest integer.

The Hamilton path can be formulated as a permutation π for the city set. Thereafter, TSP is to find a shortest path in all permutations. More precisely, we consider

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

where N is the number of cities and $\pi(k)$ is the kth element of the permutation π.

III. Summary of the Test Suite

Our benchmark set includes 25 MSTSPs. They are divided into three categories based on their design methods, which are generally described below.

The first category (MSTSP1 - MSTSP6) consists of six simple MSTSPs, whose cities are randomly generated. For these instances, we obtain the ground-truth best solutions by the brute-force search. However, since the solution space increases exponentially with the increasing number of cities, for relatively large-scale instances, it is impossible to traverse all possible permutations considering the time limitation. Consequently, in this category, the number of cities ranges from 9 to 12. Besides, the number of optima ranges from 2 to 13. As an example, the MSTSP1 are plotted in Fig. 1, where the black circles represent cities and the red lines constitute the optimal tour. Particularly, the MSTSP1 has three optimal tours, each has been depicted in a subgraph of Fig. 1. The length of each optimum is displayed above the subgraph. It can be observed that the three optima possess exactly the same length of 680, but different tours.

The second category (MSTSP7 - MSTSP12) includes six geometry MSTSPs. Unlike the first category that the instances are randomly generated, now we utilize the symmetrical geometry to construct MSTSPs. By designing different geometric topologies, these instances can have

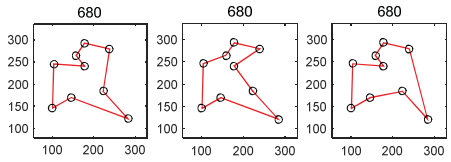

Fig. 1.MSTSP1 with 9 cities and 3 optimal tours

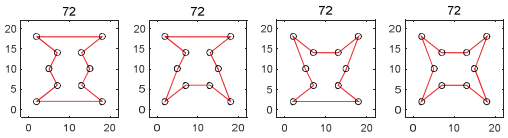

Fig. 2.MSTSP9 with 10 cities and 4 optimal tours

diverse numbers of optima. Thus, in MSTSP7 - MSTSP12, the number of optima are between 4 and 196, while the number of cities are between 10 and 15. To be specific, different symmetrical geometries are used, including the rectangle, the regular pentagon, and the regular hexagon. Cities are located on vertexes of each geometry. Under different geometries, the optima have significantly different tours for each instance. We take MSTSP9 as an example, which is drawn in Fig. 2. A regular hexagon is nested inside a large rectangle, which generates four optimal tours. It can be observed from Fig. 2 that the four tours possess totally different topologies.

The third category (MSTSP13 - MSTSP25) is comprised of 13 composite MSTSPs, which are relatively large-scale instances. The composite MSTSPs are constructed with some basic small-scale MSTSPs. Each small-scale MSTSP is considered as a city cluster, and the city clusters are distributed at different geometric locations in the composite MSTSP. On the one hand, some city clusters possess diverse sub-tours with equal lengths, and thus provide multiple optimal tours for the composite MSTSPs. On the other hand, the geometric distribution of city clusters provides additional possibilities for the composite instances to have multiple diverse solutions. Thus, to summarize, the diversity of the optimal tours comes from both the intra-cluster relationship between cities and the inter-cluster relationship between city clusters. For MSTSP13 - MSTSP25, the maximum city size is raised to 66, while the number of optima ranges from 4 to 72. More specifically, the city clusters can be designed with geometric locations (such as the cases from MSTSP13 to MSTSP 16) or with randomly generated locations (such as the cases from MSTSP17 to MSTSP25). Moreover, concerning the inter-cluster relationship, MSTSP13 and MSTSP14 possess a single optimal tour for the city clusters, while the other composite MSTSPs have multiple optimal tours among city clusters. As an example, the MSTSP21 has two optimal inter-cluster topologies, which are shown in the two sub-figures of Fig. 3(a). Besides, the instance has four city clusters, which are labeled with A, B, C, and D, respectively, using a subscript with the index of the optimal topology. If we zoom in one of the city cluster (e.g., B_1) to see more details, as shown in Fig. 3(b), the cluster has two different optimal intra-cluster topologies. Similarly, the cluster B_2, displayed in Fig. 3 (c),

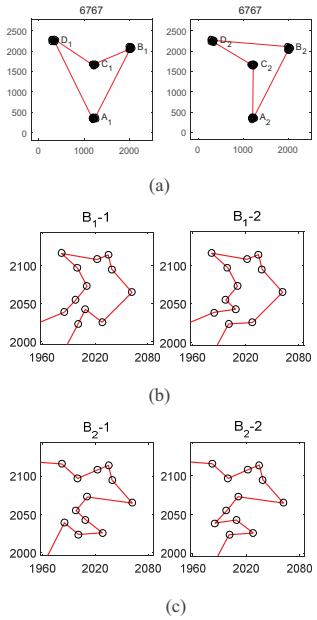

Fig. 3.MSTSP9 with 48 cities and 4 optimal tours

possesses two distinct intra-cluster topologies with the equal subtour length.

The above three categories contain 25 MSTSPs in total, which are summarized in Table I, including the category information, the index, the name, the number of cities, the number of optima, and the optimal length. For more details, please refer to the supplementary file and the source code of the benchmark suite. They are available to download on the Internet [1].

IV. The Neighborhood-based Genetic Algorithm

To settle the proposed 25 MSTSPs, a neighborhood-based genetic algorithm (NGA) is proposed. GA is a popular optimizer to tackle TSP, for the easy implementation and good search ability. The neighborhood-based strategy restricts the search of neighborhood members within respective local spaces, allowing locating diverse candidate solutions simultaneously. Eventually, NGA can obtain various solutions at the end of optimization. In the following, we describe the details of NGA.

A. Overall Framework

To begin with, NGA initializes *NP* chromosomes and evaluates their tour lengths. Next, the algorithm enters the evolution loop. The neighborhood strategy is adopted to divide the entire population into several groups and form a mating pool eventually. Then, pairwise parents in the mating pool execute partially mapped crossover (PMX) [23]. After that, we mutate the chromosomes by reversing genomes between two randomly selected positions, termed reverse sequence mutation (RSM) [24]. Thus far, we obtain offspring as well as their tour lengths. Afterwards, we select chromosomes from the offspring to determine the parents of the next generation. The above procedures repeat until the termination condition is met. When the algorithm terminates, a post-processing method is applied to identify the representative solutions of the final populations and

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

TABLE I. Test Instances of MSTSPs

Category	Index	Name	#City	#Optima	Optimal length
Simple	MSTSP1	simple1_9	9	3	680
	MSTSP2	simple2_10	10	4	1265
	MSTSP3	simple3_10	10	13	832
	MSTSP4	simple4_11	11	4	803
	MSTSP5	simple5_12	12	2	754
	MSTSP6	simple6_12	12	4	845
Geometry	MSTSP7	geometry1_10	10	56	130
	MSTSP8	geometry2_12	12	110	1344
	MSTSP9	geometry3_10	10	4	72
	MSTSP10	geometry4_10	10	4	72
	MSTSP11	geometry5_10	10	14	78
	MSTSP12	geometry6_15	15	196	130
Composite	MSTSP13	composite1_28	28	70	3055
	MSTSP14	composite2_34	34	16	3575
	MSTSP15	composite3_22	22	72	9455
	MSTSP16	composite4_33	33	64	8761
	MSTSP17	composite5_35	35	10	9061
	MSTSP18	composite6_39	39	20	23763
	MSTSP19	composite7_42	42	20	14408
	MSTSP20	composite8_45	45	20	10973
	MSTSP21	composite9_48	48	4	6767
	MSTSP22	composite10_55	55	9	10442
	MSTSP23	composite11_59	59	10	24451
	MSTSP24	composite12_60	60	36	9614
	MSTSP25	composite13_66	66	26	9521

Algorithm 1 NGA

Input: A MMTSP test instance T, the population size NP, the neighborhood size m, the crossover rate P_c, the mutation ratio P_m, and the termination criterion.

Output: The representative solution set $\mathbb{S}$.

1: *Parent* ← Initialize(*NP*)
2: Evaluate(*Parent*)
3: **While** the termination criterion not satisfied **do**
4: *MatingPool* ← Neighborhood(*Parent, m*)
5: *Offspring* ← Crossover(*MatingPool*, "PMX") /* apply the PMX [23] with a probability P_c */
6: *Offspring* ← Mutation(*Offspring*, "RSM") /* apply the RSM [24] with a probability P_m */
7: Evaluate(*Offspring*)
8: *Parent* ← Selection(*Offspring, m*)
9: **End While**
10: $\mathbb{S}$ ← Preserve (*Parent*, *m*)

Algorithm 2 Neighborhood(*Parent, m*)

Output: The chromosome set *MatingPool*
1: *tmpParent* ← *Parent*
2: *MatingPool* = ∅
3: **While** *tmpParent* ≠∅ **do**
4: *Leader* ← FindBest(*tmpParent*)
5: CalculateShareDist(*tmpParent*, *Leader*)
6: *sortParent* ← Sort(*tmpParent*.shareDist, "ascending")
7: *NeighborGroup* ← *sortParent* [1, …., *m*]
8: **If** all the members of *NeighborGroup* are the same **then**
9: DiversityEnhancement(*NeighborGroup*)
10: **End If**
11: Shuffle(*NeighborGroup*)
12: *MatingPool* = *MatingPool* + *NeighborGroup*
13: *tmpParent* = *tmpParent* - *NeighborGroup*
14: **End While**

output a solution set. The overall procedure is presented in **Algorithm 1**.

B. Neighborhood Grouping Strategy

The neighborhood grouping strategy divides the entire population based on the individual distances. As presented in **Algorithm 2**, first, *tmpParent*, a copy of *Parent*, is created. Then, the chromosome with the shortest tour is identified as the neighborhood leader *Leader*. Subsequently, members who are close to *Leader* more tend to form a group. To this end, a similarity measure between solutions of MSTSPs is required.

Note that the solution of an MSTSP is encoded as a permutation, and hence it is meaningless to directly compare the city indexes on the corresponding positions. Instead, the adjacency information between cities is more important. Therefore, we define the similarity measure based on the common edges between two tours. Suppose π_i and π_j are two permutations, while $\Phi(\pi_i)$ and $\Phi(\pi_j)$ denote their edge sets. Then, the similarity between π_i and π_j is calculated as

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

where $|\cdot|$ denotes the number of edges in the intersection set.

After that, the similarity values between *Leader* and all members are calculated. The calculated values are used to sort *tmpParent* in an ascending order, obtaining *sortParent*. The first *m* chromosomes of *sortParent* together form *NeighborGroup*. In addition, we also consider the diversity loss caused by the trap of the local optima. A diversity enhancement approach is performed on *NeighborGroup* to avoid the search getting into the local optima. Specifically, if all the members of *NeighborGroup* are the same, we reinitialized *m*-1 members, leaving one member unchanged. Then, *NeighborGroup* is shuffled. When the members of *NeighborGroup* are settled, they are added to *MatingPool*. In the meantime, the newly added members are eliminated from *tmpParent*. The above process is repeated until the set *tmpParent* is empty.

Algorithm 3 Preserve(*Parent*, *m*)

Output: The representative solution set $\mathbb{S}$.
1: *ShortestLength*← SelectBestFitness(*NeighborGroup*) /* return the best fitness value. */
2: *thrLength* ← *ShortestLength*·(1 + ε)
3: $\mathbb{S}$ = ∅
4: **For** *i = 1 to NP/m* **do**
5: *NeighborGroup* ←(*Parent*[(*i*-1)**m* + 1, …, *i***m*])
6: **For every** *e* ∈ *NeighborGroup* **do**
7: **If** Exsit($\mathbb{S}$, *e*) **then**
8: **continue**
9: **End If**
10: **If** *e*.Length == *ShortestLength* **then**
11: $\mathbb{S} \leftarrow \mathbb{S} \cup s$ /* preserve the chromosome best so far */
12: **Else If** *e*.Length ≤ *thrLength* **then**
13: *maxSim* ← 0
14: **For every** *o* ∈ $\mathbb{S}$ **do**
15: **If** *S*(*e*, *o*) > *maxSim* **then**
16: *maxSim*← *S*(*e*, *o*)
17: **End If**
18: **End For**
19: **If** *maxSim* < *thrSim* **then**
20: $\mathbb{S} \leftarrow \mathbb{S} \cup e$ /* preserve diverse chromosome*/
21: **End If**
22: **End If**
23: **End For**
24: **End For**

C. Evolutionary Operations

The basic evolutionary operations of GA include crossover, mutation, and selection, which are shown in the lines 5 - 8 of **Algorithm 1**. The three operations are applied within the same neighborhood group to avoid global convergence. To be specific, crossover and mutation operations are adopted with parent pairs of the same neighborhood group. That is to say, most gene fragments of offspring are inherited from the members of the same neighborhood group, and hence the offspring and the parents tends to be close to each other. Selection operation keeps the superior and diverse solutions while discard the others. For each child, the most similar parent of the corresponding neighborhood group is identified. The length values between the child and this parent are compared. If the child is shorter, it will take the place of the chosen parent.

D. Post-processing Method

When the algorithm terminates, it will obtain *NP* chromosomes, which correspond to *NP* solutions. However, some of them are unnecessarily to be provided because of redundancy or inferiority. Considering this, a post-processing method is called to identify the representative

ones and then offer them in the final output set. The procedure is described in **Algorithm 3**. To begin with, find the solution with the shortest length and record the length as *ShortestLength*. Then, a selection threshold *thrLength* is defined as an ε-relaxion of the *ShortestLength*. Afterwards, we deal with the *NP* solutions one by one: we first check whether they have already existed in the final output set $\mathbb{S}$. If yes, the redundant solutions will be discarded. Otherwise, we allow a solution to join $\mathbb{S}$ under two conditions: 1) its length is exactly the same as the length of so far optima (shown in the lines 10 - 11 of **Algorithm 3**), and 2) its length is shorter than *thrLength* and is distinguished from the members in $\mathbb{S}$, which is controlled by *thrSim* (described in the lines 12 - 22 of **Algorithm 3**). Here, ε is set to 0.01 and *thrSim* is 2*ln(*N*)/*N*.

V. Experimental and Analysis

In this section, the experimental setups, including the benchmark suite, the comparison algorithms, and parameter setting are presented in Section V-A. Then, the performance measures are described in Section V-B. At last, Section V-C reports the experimental results.

A. Experimental Setup

To evaluate the performance of discrete MMO algorithms, we adopt the proposed benchmark suite of 25 MSTSPs, which are described in Section III. Existing discrete MMO algorithms include niching ant colony system (NACS) [20] and multi-chromosomal cramping based genetic algorithm (MCC-GA) [19]. NACS incorporates a diversity-preserving mechanism into an ant colony system to solve MSTSPs. By the diversity-preserving mechanism, multiple pheromone matrices are preserved, which guide the search of ants towards distinct directions, and thus enables the algorithm to obtain diverse solutions in parallel. MCC-GA encodes *l* solutions into a chromosome. The population search is devoted to finding exact *l* solutions. The value of *l* should be given in advance. However, it is difficult to appropriately set without any prior knowledge of test instances. Therefore, as a proof of principle study, we investigate the performance of NGA and NACS on the proposed test suite.

The parameters of the NGA are empirically set: crossover rate P_c = 0.9, mutation rate P_m = 0.1, and neighborhood size m = 6. The parameters of NACS are set according to the corresponding publication [20]. The size of population is set to 150. All the algorithms terminate when the given maximum fitness evaluations (MaxFEs) are exhausted. The MaxFEs of 25 MSTSPs has two settings, which are listed in Table II. The algorithm should run 50 times independently to obtain the statistical results.

B. Performance Measures

1) Identification of the Optimal Tours

Each of the above instances has a set of ground-truth optimal solutions, denoted as $\mathbb{P}$. The aim of an optimization algorithm is to locate all optima. For evaluation, we need first compare the solution set $\mathbb{S}$ returned by the algorithm with the ground truths to identify the optimal tours found by the algorithm by utilizing Eq. (2). A solution is considered to be optimal as long as its similarity with the one of ground-truth solutions equals to 1.

TABLE II. MaxFEs Applied for 25 MSTSP Instances

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

1) F_β Measure

Typically, an optimization algorithm will provide a final solution set $\mathbb{S}$ when they meet the termination condition. However, the solution set may contain many inferior and redundant solutions. To quantify the quality of the solution set, we introduce F_β, which is widely used in pattern recognition and information retrieval, as a measure. F_β is a comprehensive indicator for accessing the precision value *P* and the recall value *R* of obtained solutions. *P* is the fraction of the obtained solutions that are optimal solutions:

$$P = \frac{TP}{TP + FP} \tag{3}$$

where *TP* is the number of optimal solutions in $\mathbb{S}$, and *FP* is the number of non-optimal solutions in $\mathbb{S}$. *R* is the fraction of the ground-truth solutions that are successfully located:

$$R = \frac{TP}{TP + FN} \tag{4}$$

where *FN* is the number of 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 precision and recall, F_β [25] is calculated as

$$F_\beta = \frac{(1+\beta^2)\cdot P \cdot R}{\beta^2 \cdot P + R} \tag{5}$$

When β is set to 1, it assigns the same importance to both *P* and *R*. However, for test instance with numerous optima, it is more important to locate the most representative ones than to locate all of them. So that we set β^2 to 0.3 to magnify the effect of precision in evaluating the solutions. Besides, *P*, *R*, and F_β are real values between 0 and 1. Ideally, *P* equals to 1 when all the solutions offered by the algorithm are optimal; *R* meets 1 when all the ground-truth solutions are located by the algorithm; and subsequently, F_β achieves 1 when the values of *P* and *R* both equal to 1.

To make a further explanation, we take MSTSP9 with 4 optima as an example and compare with two different algorithms A and B. The algorithm A offers 100 solutions, whereas the algorithm B provides only 4 representative solutions. We suppose that these two algorithms successfully locate all the ground-truth solutions. It is obvious that the solution quality obtained by B is better than that obtained by A, since B provides only necessary solutions while A has redundant and inferior solutions. In the following, we give a quantitative description. For algorithm A, it has *TP* = 4, *FP* = 96, and *FN* = 0. Therefore, we can obtain *P* = 0.04 and *R* = 1 according to Eqs. (3) and (4). For B, it has *TP* = 4, *FP* = 0, and *FN* = 0 according to Eqs. (3) and (4). Therefore, we can have *P* = 1 and *R* = 1. In summary, based on Eq. (5), F_β value of A is about 0.051, while that of B is 1. From this perspective, the conclusion is consistent with the previous intuition.

2) Diversity Indicator

Diversity indicator (DI) is another essential measure for evaluating the algorithm performance. When algorithms fail to locate any desired solutions, their F_β values are all zero. In these cases, the 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 [22], DI measures the diversity of $\mathbb{S}$ based on the convergence of the solutions towards 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

$$\mathrm{DI}(\mathbb{P},\mathbb{S}) = \frac{\sum_{i=1}^{|\mathbb{P}|} \max_{j=1,\ldots,|\mathbb{S}|} S(p_i, s_j)}{|\mathbb{P}|} \tag{6}$$

where p_i is the ith permutation of $\mathbb{P}$, s_j is jth permutation of $\mathbb{S}$, and $S(p_i, s_j)$ is the similarity between the permutations p_i and s_j using Eq.(1).

C. Simulation Results

The F_β and DI obtained by NACS and NGA are compared in Table III and Table IV. Better results are marked in boldface. From Table III, we find that NGA are superior to NACS on 18 out of 25 test instances in terms of F_β. However, for the last seven test instances (MSTSP19 - MSTSP25), they both get nearly zero F_β values, which indicates they fail to find any satisfactory solution in the MSTSPs with relatively large-scale cities. The observations show that the MSTSP instances with relatively large-scale cities pose a challenge to discrete MMO algorithms.

DI is presented in Table IV. The higher DI implies that the obtained solutions are more diverse. In Table IV, NGA performs better on 17 out of 25 test instances concerning DI. For the simple MSTSPs and the geometry MSTSPs, i.e., from MSTSP1 to MSTSP12, NGA is always the winner except for MSTSP5. For the composite MSTSPs (MSTSP13 - MSTSP25), NGA performs better than NACS on about half test instances, 6 out of 13. The DI values of NGA degrade along with the number of cities increasing. The observations imply that the neighborhood-based search of NGA facilitates the population diversity, but it is inadequate for relative large-scale MSTSPs.

To further investigate the significant effect with respect of F_β and DI, we also conduct the Wilcoxon rank sum test at significant level $\alpha = 0.05$. The results are listed in Table V. From the table, we can conclude that NGA is significantly better than NACS concerning F_β on 16 out of 25 instances, while they tie on 8 test instances. As to another indicator, DI, NGA is significantly better than NACS on 17 out of 25 instances, while NGA loses on another 8 test instances. The overall significance test results point out that the NGA outperforms in terms of both solution quality and solution diversity.

TABLE III. F_β OF NACS AND NGA

Instance	MSTSP1	MSTSP2	MSTSP3	MSTSP4	MSTSP5
NACS	0.684	0.804	0.497	0.724	**0.989**
NGA	**0.973**	**0.959**	**0.935**	**0.932**	0.846
Instance	MSTSP6	MSTSP7	MSTSP8	MSTSP9	MSTSP10
NACS	0.643	0.125	0.137	0.768	0.813
NGA	**0.877**	**0.769**	**0.578**	**0.974**	**0.969**
Instance	MSTSP11	MSTSP12	MSTSP13	MSTSP14	MSTSP15
NACS	0.459	0.090	0.025	0.087	0.004
NGA	**0.949**	**0.331**	**0.096**	**0.172**	**0.416**
Instance	MSTSP16	MSTSP17	MSTSP18	MSTSP19	MSTSP20
NACS	0.000	0.000	0.000	0.000	0.000
NGA	**0.054**	**0.044**	**0.031**	**0.007**	0.000
Instance	MSTSP21	MSTSP22	MSTSP23	MSTSP24	MSTSP25
NACS	**0.012**	0.000	0.000	0.000	0.000
NGA	0.000	0.000	0.000	0.000	0.000

TABLE IV. DI OF NACS AND NGA

Instance	MSTSP1	MSTSP2	MSTSP3	MSTSP4	MSTSP5
NACS	0.788	0.894	0.757	0.809	**0.983**
NGA	**0.980**	**0.972**	**0.957**	**0.947**	0.916
Instance	MSTSP6	MSTSP7	MSTSP8	MSTSP9	MSTSP10
NACS	0.843	0.566	0.652	0.820	0.850
NGA	**0.943**	**0.869**	**0.838**	**0.975**	**0.969**
Instance	MSTSP11	MSTSP12	MSTSP13	MSTSP14	MSTSP15
NACS	0.758	0.732	0.752	**0.876**	0.744
NGA	**0.963**	**0.809**	**0.792**	0.844	**0.847**
Instance	MSTSP16	MSTSP17	MSTSP18	MSTSP19	MSTSP20
NACS	0.680	0.765	0.671	0.675	**0.745**
NGA	**0.783**	**0.803**	**0.704**	**0.699**	0.671
Instance	MSTSP21	MSTSP22	MSTSP23	MSTSP24	MSTSP25
NACS	**0.773**	**0.713**	**0.671**	**0.724**	**0.725**
NGA	0.628	0.409	0.344	0.319	0.270

TABLE V. THE SIGNIFICANT RESULTS FOR NGA AND NACS

Wilcoxon rank sum test	F_β	DI
NGA vs. NACS (significantly worse than/similar/significantly better than)	1/8/16	8/0/1

VI. CONCLUSION

This paper makes a preliminary study to extend the MMO into combinatorial area. To deal with combinatorial MMO problems, we incorporate a neighborhood-based strategy into GA to obtain diverse solutions. The GA facilitates global search to obtain the optimal solution, while the neighborhood-based strategy contributes to maintaining population diversity to locate potential candidates in parallel. Moreover, to evaluate performance of the combinatorial MMO algorithm, we propose a benchmark suite of 25 MSTSPs designed by three different methods. Furthermore, the proposed NGA and the compared NACS are utilized to solve 25 MSTSPs. We adopt two indicators to compare their performance. F_β are adospted to comprehensively assess the precision and recall values of the returned solutions, while DI measures the diversity and convergence of the obtained solutions. It is experimentally verified that NGA can achieve a competitive performance

concerning with F_β and DI on most MSTSPs. However, they both fail to obtain any desired solutions on the MSTSP instances with large city size.

In the future work, we will improve NGA to work out the MSTSPs with large city size. In addition, we attempt to design more discrete MMO test instances to supplement the discrete benchmark.

REFERENCES

[1] K. Dorling, J. Heinrichs, G. G. Messier and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70-85, Jan. 2017.

[2] X. Cai, Y. Li, Z. Fan, and Q. Zhang, "An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 508–523, Aug. 2015.

[3] C. Patvardhan, S. Bansal, and A. Srivastav, "Quantum-inspired evolutionary algorithm for difficult knapsack problems," *Memetic Computing*, vol. 7, no. 2, pp. 135–155, Jun. 2015.

[4] I.-D. Psychas, E. Delimpasi, and Y. Marinakis, "Hybrid evolutionary algorithms for the Multiobjective Traveling Salesman Problem," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8956–8970, Dec. 2015.

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

[6] Y.-J. Gong, J. Zhang, and Y. Zhou, "Learning multimodal parameters: A bare-bones niching differential evolution approach," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–16, 2017.

[7] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[8] Y.-H. Zhang, Y.-J. Gong, H.-X. Zhang, T.-L. Gu, and J. Zhang, "Toward fast niching evolutionary algorithms: A locality sensitive hashing-based approach," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 347–362, Jun. 2017.

[9] K. Price, "Differential evolution: a fast and simple numerical optimizer," in *1996 Biennial Conference of the North American Fuzzy Information Processing Society*, 1996, Jun 1996, pp. 524–527.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948.

[11] C. L. P. Chen, T. Zhang, L. Chen, and S. C. Tam, "I-Ching divination evolutionary algorithm and its convergence analysis," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 2–13, Jan. 2017.

[12] T. Zhang, C. L. P. Chen, L. Chen, X. Xu, and B. Hu, "Design of highly nonlinear substitution boxes based on I-Ching operators," *IEEE Transactions on Cybernetics*, pp. 1–10, 2018.

[13] J. H. Holland, Adaptation in natural and artificial systems. Ann Arbor, MI: Univ. of Michigan Press, 1975

[14] Y. J. Gong et al., "Genetic learning particle swarm optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, 2016.

[15] T. Liao, K. Socha, M. A. M. de Oca, T. Stützle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 503–518, Aug. 2014

[16] S. W. Mahfoud, "Crowding and preselection revisited," in Parallel problem solving from nature 2, R. Manner and B. Manderick, Eds. ¨ Amsterdam: North-Holland, 1992, pp. 27–36.

[17] O. Mengsheol and D. Goldberg, "Probabilistic crowding: Deterministic crowding with probabilistic replacement," in *Proceedings of Genetic and Evolutionary Computation*, Jul. 1999, pp. 409–416.

[18] J. P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evolutionary Computation*, vol. 10, pp. 207-234, Sep. 2002.

[19] S. Ronald, "Finding multiple solutions with an evolutionary algorithm," in *Proceedings of IEEE International Conference on Evolutionary Computation*, Perth, WA, Australia, 1995, vol. 2, pp. 641–646 vol.2.

[20] X.-C. Han, H.-W. Ke, Y.-J. Gong, et al., "Multimodal optimization of traveling salesman problem: A niching ant colony system," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2018.

[21] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376-384, 1991.

[22] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[23] J. Lingle Robert, "Alleles, loci and the traveling salesman problem," in *Proceedings of International Conference on Genetic Algorithms and their Applications*, vol. 12, no. 92, pp. 154–159, 1985.

[24] O. Abdoun, J. Abouchabaka, and C. Tajani, "Analyzing the performance of mutation operators to solve the travelling salesman problem," *International Journal of Emerging Sciences*, vol. 2, no. 1, 2012.

[25] P. Flach and M. Kull, "Precision-Recall-Gain Curves: PR analysis done right," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 838–846.