Evolutionary Multimodal Multiobjective Optimization for Traveling Salesman Problems

Yiping Liu, *Member, IEEE*, Liting Xu, Yuyan Han, Xiangxiang Zeng, Gary G. Yen, *Fellow, IEEE*, and Hisao Ishibuchi, *Fellow, IEEE*

***Abstract*—Multimodal multiobjective optimization problems (MMOPs) are commonly seen in real-world applications. Many evolutionary algorithms have been proposed to solve continuous MMOPs. However, little effort has been made to solve combinatorial (or discrete) MMOPs. Searching for equivalent Pareto-optimal solutions in the discrete decision space is challenging. Moreover, the true Pareto-optimal solutions of a combinatorial MMOP are usually difficult to know, which has limited the development of its optimizer. In this article, we first propose a test problem generator for multimodal multiobjective traveling salesman problems (MMTSPs). It can readily generate MMTSPs with known Pareto-optimal solutions. Then, we propose a novel evolutionary algorithm to solve MMTSPs. In our proposed algorithm, we develop two new edge assembly crossover operators, which are specialized in searching for superior solutions to MMTSPs. Moreover, the proposed algorithm uses a new environmental selection operator to maintain a good balance between the objective space diversity and decision space diversity. We compare our algorithm with five state-of-the-art designs. Experimental results convincingly show that our algorithm is powerful in solving MMTSPs.**

***Index Terms*—Combinatorial optimization, evolutionary multimodal multiobjective optimization, test problems, traveling salesman problem (TSP).**

Manuscript received 16 May 2022; revised 2 October 2022 and 9 December 2022; accepted 20 January 2023. Date of publication 24 January 2023; date of current version 2 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62106073, Grant 62250710163, Grant 61876075, and Grant 61973203; in part by the Natural Science Foundation of Hunan Province of China under Grant 2021JJ40116; in part by the Fundamental Research Funds for the Central Universities under Grant HNU:531118010537; in part by the Guangyue Young Scholar Innovation Team of Liaocheng University under Grant LCUGYTD2022-03; in part by the Guangdong Provincial Key Laboratory under Grant 2020B121201001; in part by the Program for Guangdong Introducing Innovative and Enterpreneurial Teams under Grant 2017ZT07X386; in part by the Stable Support Plan Program of Shenzhen Natural Science Fund under Grant 20200925174447003; and in part by the Shenzhen Science and Technology Program under Grant KQTD2016112514355531. *(Corresponding author: Gary G. Yen.)*

Yiping Liu, Liting Xu, and Xiangxiang Zeng are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China (e-mail: yiping0liu@gmail.com; litingxu@hnu.edu.cn; xzeng@foxmail.com).

Yuyan Han is with the School of Computer Science, Liaocheng University, Liaocheng 252059, China (e-mail: hanyuyan@lcu-cs.com).

Gary G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078 USA (e-mail: gyen@okstate.edu).

Hisao Ishibuchi is with the Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: hisao@sustech.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2023.3239546.

Digital Object Identifier 10.1109/TEVC.2023.3239546

I. Introduction

SOLUTIONS to combinatorial optimization problems are discrete. Typical combinatorial optimization problems, such as traveling salesman problems (TSPs), knapsack problems, and job-shop scheduling problems, are NP-hard. Evolutionary computation is a popular approach for solving those problems within a reasonable time frame.

There usually exist multiple global optima or local optima to a combinatorial optimization problem. Then, the problem is termed to be multimodal. For example, there may exist several different optimal paths to a multimodal TSP [1]. The general goal of applying evolutionary algorithms is to search for global optima and good local optima as many as possible.

A combinatorial optimization problem often involves multiple objective functions. For example, the objective functions of a TSP include the total travel distance and time [2]. Due to the conflicts between the objectives, there is a set of Pareto-optimal solutions to the problem rather than a single optimal solution. The image of the Pareto-optimal solution set in the objective space is called the Pareto front. Numerous evolutionary multiobjective algorithms have been proposed to search for a solution set to approximate the Pareto front.

A combinatorial optimization problem is multimodal and multiobjective if there exist 1) local Pareto-optimal solutions of interest and/or 2) multiple Pareto-optimal solutions corresponding to a single point on the Pareto front. Although evolutionary multimodal multiobjective algorithms (EMMAs) have been a hot research topic, most studies solely focus on solving continuous multimodal multiobjective optimization problems (MMOPs). Whereas decision variables are often discrete in real-world applications, few studies on combinatorial MMOPs were reported in the literature. Searching for a set of Pareto-optimal solutions well distributed in the discrete decision space is challenging, to say the least.

In a preliminary study [3], several small-scale multimodal multiobjective TSPs (MMTSPs) were proposed and discussed as test problems. On the one hand, small-scale MMTSPs are not sufficient to test the performance of an optimizer. On the other hand, enumerating Pareto-optimal solutions to large-scale MMTSPs is computationally intractable. Without knowing Pareto-optimal solutions, it is difficult, if not impossible,

to understand how far we are from solving the problems satisfactorily (i.e., it is difficult to evaluate the performance of an optimizer). An evolutionary algorithm was also proposed to solve those small-scale MMTSPs in [3]. However, it was not fully developed by examining the characteristic of MMTSPs and may not perform well on large-scale MMTSPs.

In this article, we first propose a test problem generator for MMTSPs. Then, we propose a novel evolutionary algorithm for solving MMTSPs, named TSPXEA. In this algorithm, we develop two new edge assembly crossover (EAX) [4] operators that are specialized in solving MMTSPs. We also designed a new environmental selection operator to balance diversities in the objective and decision spaces. The contributions made in this article are in twofold.

1) We propose a test problem generator for MMTSPs. It can readily generate a test problem with known Pareto-optimal solutions. To the best of our knowledge, it is the first test problem generator to produce large-scale combinatorial MMOPs with known Pareto-optimal solutions. Moreover, the convergence difficulty of the underlying test problem is controllable in our generator. Such a test problem is instrumental in developing and examining optimizers for combinatorial MMOPs.
2) We propose a novel evolutionary algorithm named TSPXEA to solve MMTSPs. TSPXEA has two main features.
   a) It applies two new EAX operators, EAX-W and EAX-ND. EAX-W is good at searching for solutions close to the Pareto front, and its time consumption is low. Whereas EAX-ND is more time consuming than EAX-W, it can search for well-converged and well-distributed solutions.
   b) In environmental selection, k-nearest neighbor and entropy are combined as a selection criterion to maintain diversity in both the objective and decision spaces. Experimental results show that TSPXEA significantly outperforms state-of-the-art optimizers on MMTSPs.

The source codes of our test problem generator and the proposed evolutionary algorithm for MMTSP are downloadable from https://github.com/yiping0liu.

The remainder of this article is organized as follows. In Section II, related studies are surveyed for the completeness of the presentation. The proposed test problem generator and TSPXEA are described in detail in Sections III and IV, respectively. Section V shows experimental results. Section VI concludes this article and provides future research directions.

II. Related Works

In this section, we first introduce the concepts of MMOPs and review the test problems in Section II-A. Then, we survey existing evolutionary algorithms for solving MMOPs in Section II-B. Finally, we discuss the related works on TSPs in Section II-C.

A. Multimodal Multiobjective Optimization Problems

Without loss of generality, an MMOP can be formulated as follows:

$$\text{Minimize } \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})) \quad \text{s.t.} \quad \boldsymbol{x} \in S \qquad (1)$$

where $\boldsymbol{x}$ represents a solution in the decision space S. $f_m(\boldsymbol{x})$, $m = 1, \ldots, M$, is the mth objective to be minimized, and M is the number of objectives considered. The definitions are widely accepted in multiobjective optimization.

Pareto Dominance: For any two solutions, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, if $\forall m = 1, \ldots, M$, $f_m(\boldsymbol{x}_1) \leq f_m(\boldsymbol{x}_2)$, and $\exists i \in \{1, \ldots, M\}$, $f_i(\boldsymbol{x}_1) < f_i(\boldsymbol{x}_2)$, then $\boldsymbol{x}_1$ dominates $\boldsymbol{x}_2$, denoted as $\boldsymbol{x}_1 \prec \boldsymbol{x}_2$.

Pareto-Optimal Solution: For a solution $\boldsymbol{x}^*$, if $\nexists\, \boldsymbol{x}$ such that $\boldsymbol{x} \prec \boldsymbol{x}^*$, then $\boldsymbol{x}^*$ is termed a (global) Pareto-optimal solution.

Local Pareto-Optimal Solution: For a solution $\boldsymbol{x}'$, if $\nexists\, \boldsymbol{x} \in \{\boldsymbol{x} | d(\boldsymbol{x}, \boldsymbol{x}') < \epsilon\}$ such that $\boldsymbol{x} \prec \boldsymbol{x}'$, then $\boldsymbol{x}'$ is termed a local Pareto-optimal solution. Here, ϵ is a small positive value, and $d(\cdot, \cdot)$ is a distance measure between solutions in the decision space.

Based upon these definitions, we can further define the equivalent Pareto-optimal solution which plays a critical role in multimodal multiobjective optimization formulated as below.

Equivalent Pareto-Optimal Solutions: For two Pareto-optimal solutions $\boldsymbol{x}_1^* \neq \boldsymbol{x}_2^*$, if $\boldsymbol{f}(\boldsymbol{x}_1^*) = \boldsymbol{f}(\boldsymbol{x}_2^*)$, then they are equivalent Pareto-optimal solutions to each other.

MMOP: For an MMOP in (1), if there exist equivalent Pareto-optimal solutions or local Pareto-optimal solutions of interest, then it is regarded as an MMOP.

In most existing studies, the general goal of solving an MMOP is to search for a prespecified number of well-distributed solutions over the entire Pareto front in the objective space and over the entire equivalent Pareto sets in the decision space with limited time and space resources. Recently, some attempts to search for well-converged and well-distributed local Pareto-optimal solutions were reported [5], [6]. This article aims to generate challenging combinatorial MMTSPs with equivalent Pareto-optimal solutions and to search for a set of those solutions with good diversity in both the objective and decision spaces with limited time and space resources.

Obtaining equivalent Pareto-optimal solutions is essential in practice. On the one hand, the decision maker receives an approximated Pareto front in the traditional multiobjective optimization and then chooses a solution in the area of interest on the front. However, there may be factors that are difficult to capture mathematically in real-world problems [7], which may make the chosen solution inapplicable. Even if the decision maker wants to choose other solutions in the area of interest, they are likely to be inapplicable since they are similar in the decision space. Traditional multiobjective optimizers do not maintain diversity in the decision space. If we can provide equivalent Pareto-optimal solutions that are distinct from each other in the decision space, the decision maker has more options for consideration with factors that are not captured in the mathematical model.

On the other hand, finding equivalent Pareto-optimal solutions may help to reveal hidden properties or relations of the underlying optimization problem, and seeking these solutions may improve the performance of a meta-heuristic algorithm [7].

In the 2000s, several representative MMOP benchmark sets were proposed, such as TWO-ON-ONE [8], SYM-PART [9], and Omni-test problems [10]. There usually

exist multiple continuous solution sets corresponding to the whole Pareto front in those test problems. These solution sets are called equivalent Pareto subsets. For example, the Pareto-optimal solutions to SYM-PART are nine line segments in the 2-D decision space. The line segments are equivalent Pareto subsets to each other. Every point on the Pareto front of SYM-PART corresponds to nine Pareto-optimal solutions that are separately located on the nine line segments.

In recent years, MMMOP [11] and MMF [12], [13] test problems were designed with various problem characteristics. For instance, the equivalent Pareto subsets to MMF3 overlap in every dimension. In MMMOP2 and MMMOP5, a solution set uniformly distributed on the Pareto front is not uniformly distributed in the decision space. In MMMOP4, some points on the Pareto front correspond to a large number of equivalent Pareto-optimal solutions, while others do not.

The distance minimization problem (DMP) series [5], [14], [15], [16], [17] are also prevalent test problems. In a DMP, the objectives are to minimize the distances to the vertices of several polygons in the decision space. Polygons are usually equivalent Pareto subsets to each other. In [18], the imbalance between convergence and diversity in the decision space for solving MMOPs was discussed, and imbalanced DMPs were suggested. In the imbalanced DMPs, some polygons are difficult to find by an optimizer, while others are not. Applying the imbalanced DMPs as test problems exposes the deficiency in convergence-first EMMAs to maintain the diversity in the decision space.

All the test problems mentioned above are continuous MMOPs, whereas decision variables are often discrete in real-world applications. For example, a solution to job-shop scheduling problems is a job sequence. Different job sequences may end up having equal makespan and minimum wasted resource [19]. In drug discovery, a drug molecule is usually represented as a text sequence or a graph, which makes the design space discrete. The objectives of designing drugs include maximizing biological activity and minimizing toxicity. Maintaining diversity in the design space is beneficial in searching for promising drug molecules [20], [21]. In truss-structure optimization, multiple optimal truss topologies may correspond to the same weight and cost [22]. These alternatives present to the decision maker with needed flexibility to meet constraints under real-world complications.

Currently, few test problems for combinatorial MMOPs have been proposed. In IEEE CEC 2021, a set of multimodal multiobjective path-planning optimization problems [23] were presented as test problems for competition. The traffic networks in cities inspired the design of those problems. However, one critical issue is that the true Pareto-optimal solutions to those problems are difficult to know. Thus, examining the performance of an optimizer on those problems would be difficult. Some MMTSPs were presented in [3]. Those MMTSPs are simple to understand, and their true Pareto-optimal solutions are known. However, it is inadequate to test the scalable performance of an optimizer on those simple problems.

In Section III, we propose a test problem generator for MMTSPs. It can quickly generate large-scale MMTSPs with known Pareto-optimal solutions.

B. Evolutionary Multimodal Multiobjective Algorithms

Early attempts to solve MMOPs include Omni-optimizer [10], niching-covariance matrix adaptation [24], and diversity integrating optimizer [25]. They were developed based on evolutionary multiobjective algorithms with diversity promotion strategies in the decision space. For example, Omni-optimizer is based on the nondominated sorting genetic algorithm II (NSGA-II) [26] to evaluate the crowding distances of solutions in both the objective and decision spaces.

More EMMAs were proposed following the above paradigm. The double-niched evolutionary algorithm (DNEA) is based on the niche-based sharing function in both the objective and decision spaces [14]. A decomposition-based evolutionary multiobjective algorithm was integrated with addition and deletion operators [27]. A niching indicator-based multimodal multiobjective optimizer was proposed in [28]. In addition, the effects of archives and subset selection from archives are examined for EMMAs [29].

Some EMMAs with advanced strategies to handle multimodality were proposed in recent years, such as the multiobjective particle swarm optimization algorithm using ring topology and special crowding distance [12], the multimodal multiobjective evolutionary algorithm using two-archive and recombination strategies [11], the evolutionary algorithm using a convergence-penalized density method [18], the multimodal multiobjective evolutionary algorithm with dual clustering in the decision and objective spaces [30], the multipopulation multimodal evolutionary algorithm [31], the grid search-based multipopulation particle swarm optimization algorithm [32], and the two-stage double-niched evolution strategy [33].

However, all the above optimizers only work on continuous MMOPs. Few optimizers were designed to solve combinatorial MMOPs. Some optimizers [34], [35] were proposed to solve multimodal multiobjective path-planning optimization problems [23]. However, they do not develop a strategy to promote diversity in the decision space. An improved version of DNEA (DNEA-TSP) was proposed to solve MMTSPs in [3]. However, it was only investigated on some simple test problems and may not perform well on complex ones. In Section IV, we will propose a novel evolutionary algorithm to solve large-scale MMTSPs efficiently and effectively.

It should be noted that existing EMMAs (as well as the proposed algorithm in this article) can provide many optimal solutions with good diversity in both decision and objective spaces. It is not realistic from a decision-making viewpoint to show a huge number of diverse solutions to the decision maker. In this case, we can use the standard decision-making procedure in evolutionary multiobjective optimization. First, diverse solutions in the objective space is shown to the decision maker. Then, the decision maker will choose a singe final solution (or a few candidate solutions for further examination).

After this standard decision-making procedure in evolutionary multiobjective optimization, our multimodal multiobjective algorithm can show a small number of alternative solutions which are almost the same in the objective space and totally different in the decision space. If the decision maker is not interested in different solutions in the decision space, he/she does not have to examine those alternative solutions. However, in many cases, the decision maker is interested in examining a small number of alternative solutions with good diversity in the decision space. This decision making procedure does not severely increase the complexity of decision making for the decision maker and offer a clear merit to the decision maker (i.e., the choice from a small number of alternative solutions which are almost the same in the objective space and totally different in the decision space). The above decision-making process is summarized as the following steps.

Step 1 *(Decision Making in the Objective Space):*
- Step 1-1: Presentation of nondominated solutions with good diversity in the objective space.
- Step 1-2: Selection of a single solution (or a few solutions) considering the tradeoff among the objectives and constraints imposed upon the objectives.

Step 2 *(Decision Making in the Decision Space):*
- Step 2-1: Presentation of equivalent solutions with good diversity in the decision space.
- Step 2-2: Selection of a single solution (or a few solutions) considering the tradeoff among the relationship between decision variables and objectives and constraints imposed upon the decision variables.

C. Traveling Salesman Problems

In a TSP, we have a graph of vertices and edges. A circuit that visits every vertex once with no repeat is termed a Hamiltonian circuit. The goal of the TSP is to find a Hamiltonian circuit with minimum objective values, such as the total distance, risk, and time.

Studies on applying the evolutionary algorithm to solve the single-objective TSP arose in the 1980s. Classic genetic operators, including order crossover [36], partially mapped crossover [37], and cycle crossover [38], were proposed. However, they are not very efficient compared to other heuristic approaches. Thus, some advanced genetic operators were later proposed, such as EAX [4], [39] and cycle crossover II [40]. The importance of promoting population diversity in the decision space for solving the single-objective TSP was addressed in [41] and [42]. For example, an evolutionary algorithm with a diversity-based replacement strategy (DYN) [41] transforms a single-objective TSP into a two-objective optimization problem. One objective is the original objective of the single-objective TSP, and the other is a solution's diversity measured by the distance to its nearest neighbor. Several studies [1], [43], [44] discussed evolutionary computation for the multimodal single-objective TSP. For example, a niching memetic algorithm (NMA) was proposed to search for multiple optimal solutions parallelly [1].

Price	SHA	HKG	BKK	PAR	NYC	TYO
SHA	0	282	570	1176	1277	725
HKG	282	0	101	511	1335	311
BKK	570	101	0	407	717	649
PAR	1176	511	407	0	433	662
NYC	1277	1335	717	433	0	1194
TYO	725	311	649	662	1194	0

(a)

Time	SHA	HKG	BKK	PAR	NYC	TYO
SHA	0.0	2.9	6.8	33.6	15.0	9.7
HKG	2.9	0.0	3.0	16.6	16.0	5.3
BKK	6.8	3.0	0.0	12.8	20.9	6.3
PAR	33.6	16.6	12.8	0.0	8.4	16.4
NYC	15.0	16.0	20.9	8.4	0.0	16.3
TYO	9.7	5.3	6.3	16.4	16.3	0.0

(b)

Fig. 1. (a) Original ticket price between cities (unit: dollar). (b) Original flight time between cities (unit: hour).

Price	SHA	HKG	BKK	PAR	NYC	TYO
SHA	0	300	550	1200	1300	750
HKG	300	0	100	500	1350	300
BKK	550	100	0	400	700	650
PAR	1200	500	400	0	450	650
NYC	1300	1350	700	450	0	1200
TYO	750	300	650	650	1200	0

(a)

Time	SHA	HKG	BKK	PAR	NYC	TYO
SHA	0.0	3.0	7.0	33.5	15.0	9.5
HKG	3.0	0.0	3.0	16.5	16.0	5.5
BKK	7.0	3.0	0.0	13.0	21.0	6.5
PAR	33.5	16.5	13.0	0.0	8.5	16.5
NYC	15.0	16.0	21.0	8.5	0.0	16.5
TYO	9.5	5.5	6.5	16.5	16.5	0.0

(b)

Fig. 2. (a) Modified ticket price between cities (unit: dollar). (b) Modified flight time between cities (unit: hour).

There also have been many studies on solving the multiobjective TSP. An external archive guided algorithm (EAG) was proposed based on nondomination-based ranking and decomposition in [45]. The Physarum-inspired computational model was proposed to enhance the distribution of solutions in [46]. A subpopulation-based approach that integrates multiple algorithmic features was designed in [47]. A framework called dynamic constrained decomposition with grids (DCDGs) was proposed in [48]. However, all the above algorithms were incapable of solving MMTSPs. They were not intended to search for a solution set that is well distributed in the decision space.

How often we encounter MMTSPs depends on how large differences in the objective values are viewed as being no distinction by the decision maker. Let us see a real-world example of how a traditional multiobjective TSP could be considered an MMTSP.

The traditional multiobjective TSP is described as follows. Suppose that a businessman wants to arrange a travel plan to visit six cities with no repeat by flight. They are Shanghai (SHA), Hong Kong (HKG), Bangkok (BKK), Paris (PAR), New York (NYC), and Tokyo (TYO). The ticket price and flight time between each pair of cities are shown in Fig. 1. The person wants to minimize the total ticket price and flight time (i.e., he/she has two objectives). Given the scale of the problem, we enumerate all solutions and find out that there is no equivalent Pareto-optimal solution.

However, the problem is often viewed as an MMTSP in practice since we often handle ticket price and flight time using some information granules (i.e., discretization of continuous values shown in Fig. 1 into discrete values shown in Fig. 2). As an example, the ticket price and flight time between SHA and HKG are rounded into \$300 and 3 h, respectively.

Fig. 3 shows all the solutions to the MMTSP defined by Fig. 2 in the objective space (i.e., total ticket price versus total flight time). In this figure, three points are located on the Pareto front (i.e., the purple star, the green circle, and the red square). However, as shown in the box in Fig. 3, five different routes in the decision space correspond to these three points.

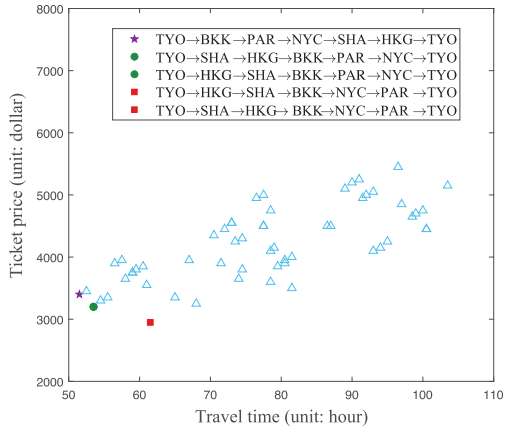


Fig. 3. All 60 solutions of the underlying example in the objective space. The Pareto front consists of the solutions labeled in the purple star, the green circle, and the red square. There are equivalent Pareto-optimal solutions corresponding to the green circle and the red square. The blue triangles denote all nonoptimal solutions (i.e., dominated solutions).

Presenting all of those five tours to the decision maker is meaningful since some factors may be difficult to model in a real-world environment. For instance, a typhoon developing in the South China Sea may render the option of flying from Shanghai to Bangkok impossible, which makes solution (TYO → HKG → SHA → BKK → NYC → PAR → TYO) infeasible. In this case, it would be great if the equivalent solution (TYO → SHA → HKG → BKK → NYC → PAR → TYO) could be also provided to the decision maker as a backup alternative.

One challenge in developing an MMTSP optimizer is how to acquire a test problem with known Pareto-optimal solutions. Analyzing the multimodality of the test problem and examining the optimizer's performance are difficult without knowing Pareto-optimal solutions. Therefore, we propose a test problem generator in the next section. The generator creates multimodality in an MMTSP by making different routes with the same contribution to the objective values.

III. Test Problem Generator

This section proposes a test problem generator that readily generates an MMTSP with known Pareto-optimal solutions. Let M be the number of objectives, n be the number of vertices, and $V = \{1, \ldots, n\}$ be the set of vertices. $e = (i, j)$ is referred to as the edge between vertices i and j, where $i, j \in V$ and $i \neq j$. Let $L_m, m = 1, \ldots, M$ be an $n \times n$ matrix. $l_m(i, j), m = 1, \ldots, M$, is the edge length corresponding to the mth objective, indicating the travel distance, risk, or time between the vertices i and j. Note that we only consider the symmetric TSP in this article, i.e., L_m is a symmetric matrix since $l_m(i, j) = l_m(j, i)$ and $l_m(i, i) = 0$. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n, x_1)$ be a path of a salesman traveling the vertices, where $x_i \in V$, and $x_i \neq x_j, j = (1, \ldots, i-1, i+1, \ldots, n)$. An M-objective MMTSP generated by the test problem generator is formulated as

$$\text{Minimize } f_m(\boldsymbol{x}) = \sum_{i=1}^{n-1} l_m(x_i, x_{i+1}) + l_m(x_n, x_1) \quad m = 1, \ldots, M. \tag{2}$$

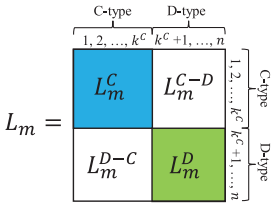


Fig. 4. $L_m, m = 1, \ldots, M$, contains the lengths of edges between vertices in an MMTSP in (2). In our benchmark generator, vertices are divided into C- and D-types. Then, L_m is divided into four submatrices, L_m^C, L_m^D, L_m^{C-D}, and L_m^{D-C}.

We will describe how to generate an MMTSP in (2), i.e., how to generate $L_m, m = 1, \ldots, M$, in Section III-A. Then, we explain how to obtain the true Pareto-optimal solutions to the MMTSP in Section III-B.

A. Generating L_m

In our test problem generator, the first k^C vertices and the last k^D vertices in V are termed C- and D-type vertices. Note that $k^C + k^D = n$. As shown in Fig. 4, we divide each $L_m, m = 1, \ldots, M$, into four submatrices, L_m^C, L_m^D, L_m^{C-D}, and L_m^{D-C}. L_m^C contains the edge lengths between C-type vertices. The setting of L_m^C affects the convergence difficulty of the problem. L_m^D contains the edge lengths between D-type vertices. The setting of L_m^D mainly affects the distribution of the Pareto-optimal solutions. L_m^{C-D} and L_m^{D-C} contain the edge lengths between C- and D-type vertices. Fig. 5 shows the building process of generating L_m by our test problem generator, where we first generate L_m^C, then L_m^D, and finally L_m^{C-D} (L_m^{D-C}). The details of generating L_m are described as follows.

Step 1 (Generating L_m^C): Let us choose any single-objective TSP with at least one known optimal solution. There may be multiple optimal solutions if the single-objective TSP is multimodal. Let L^S be the matrix that contains the length of edges between vertices in the single-objective TSP. Set the C-type vertices of the MMTSP to be the vertices of the single-objective TSP. That is, for $m = 1, \ldots, M$, let $L_m^C = L^S$. k_C is the number of vertices of the single-objective TSP.

The Pareto-optimal solutions to the MMTSP are related to the optimal solution to the single-objective TSP (or one of the optimal solutions if the problem is multimodal). We will explain the reason in Section III-B. In this article, we choose single-objective TSPs from TSPLIB [49]. TSPLIB is a repository with many single-objective TSPs with known optimal solutions.

Step 2 (Generating L_m^D): The main idea of generating L_m^D is that we first give random values to the elements in L_m^D and then set some elements in L_m^D to the same values. Algorithm 1 shows the details in step 2. Note that the number of D-type vertices, k^D, is recommended to be a relatively small value since we need to enumerate all the possible permutations of D-type vertices to obtain the Pareto-optimal solutions in Section III-B.

In lines 1–13, we have two ways of giving random values to the elements in L_m^D. For $i = 1, \ldots, k^D - 1$ (line 1) and $j = i + 1, \ldots, k^D$ (line 2), let r be a random value in $(0, 1)$ (line 3). $p_c \in [0, 1]$ is a parameter to control the number of

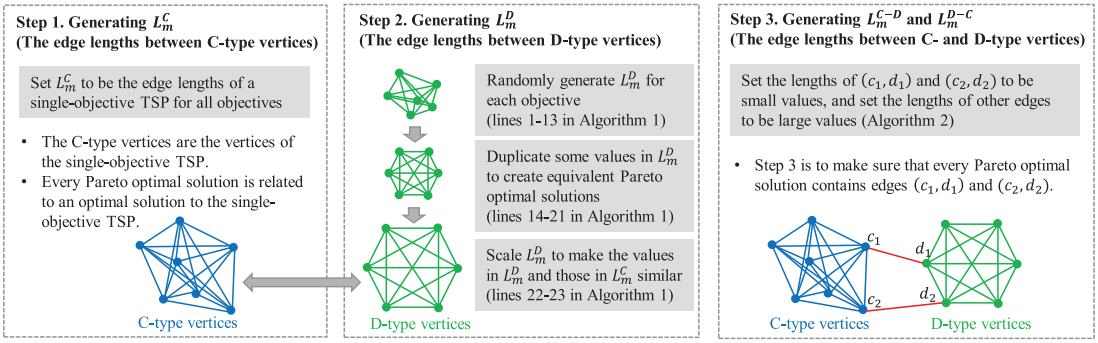


Fig. 5. Building process of generating L_m by our test problem generator.

Algorithm 1 Generating $L_m^D, m = 1, \ldots, M$

Require: k^D (the number of D-type vertices), $p_c \in [0, 1]$ (parameter for confliction among objectives), p_m (parameter for multi-modality), $k' \in [4, k^D]$ (length of S^a and S^b)

1: **for** $i = 1, \ldots, k^D - 1$ **do**
2: **for** $j = i + 1, \ldots, k^D$ **do**
3: Let r be a random value in $(0, 1)$;
4: **if** $r < p_c$ **then**
5: Randomly generate M values, $r_1, \ldots, r_M$, in $[0, 1]$ $(r_1 + \cdots + r_M > 0)$;
6: $l_m(k^C + i, k^C + j) = l_m(k^C + j, k^C + i) = r_m/(r_1 + \cdots + r_M), m = 1, \ldots, M$;
7: **else**
8: Randomly generate M values, $r_1, \ldots, r_M$, in $[0, 2/M]$;
9: $l_m(k^C + i, k^C + j) = l_m(k^C + j, k^C + i) = r_m, m = 1, \ldots, M$;
10: **end if**
11: **end for**
12: $l_m(k^C + i, k^C + i) = 0, m = 1, \ldots, M$;
13: **end for**
14: **for** $i = 1, \ldots, p_m$ **do**
15: Randomly choose k' different elements from $(k^C + 1, \ldots, n)$ and arrange them in a random order. Denote the string as $S^a = (s_1^a, \ldots, s_{k'}^a)$;
16: $S^b = (s_1^b, \ldots, s_{k'}^b) = S^a$;
17: Randomly choose two elements from $(s_2^b, \ldots, s_{k'-1}^b)$ and change their positions in S^b;
18: **for** $j = 1, \ldots, k' - 1$ **do**
19: $l_m(k_C + s_j^b, k_C + s_{j+1}^b) = l_m(k_C + s_{j+1}^b, k_C + s_j^b) = l_m(k_C + s_j^a, k_C + s_{j+1}^a), m = 1, \ldots, M$;
20: **end for**
21: **end for**
22: Calculate the scaling factor p_s by (3);
23: $L_m^D = p_s \cdot L_m^D, m = 1, \ldots, M$;
24: **return** $L_m^D, m = 1, \ldots, M$

Pareto-optimal solutions. If $r \le p_c$, we randomly generate M values, $r_1, \ldots, r_M$, in $[0, 1]$ $(r_1 + \cdots + r_M > 0)$ and then set $l_m(k^C + i, k^C + j) = l_m(k^C + j, k^C + i) = r_m/(r_1 + \cdots + r_M), m = 1, \ldots, M$ (lines 4–6). A large value of $l_u(i, j)$ results in small values of $l_m(i, j), m \ne u, m, u \in \{1, \ldots, M\}$, and vice versa, which always brings conflicts among objectives. Otherwise (i.e., if $r > p_c$), we randomly generate M values, $r_1, \ldots, r_M$, in $[0, 2/M]$ and set $l_m(k^C + i, k^C + j) = l_m(k^C + j, k^C + i) = r_m, m = 1, \ldots, M$ (lines 7–9). In both cases, the mean value of the elements is expected to be $1/M$. A large value of p_c is likely to result in a large number of Pareto-optimal solutions. Particularly, when $p_c = 1$, the number of Pareto-optimal solutions reaches the maximum value $(k_D - 2)!$ according to Section III-B.

In lines 14–21, we duplicate the values of some elements in L_m^D for p_m times to create equivalent Pareto-optimal solutions. A larger value of p_m usually implies more duplicated values in L_m^D, which leads to a larger chance of equivalent Pareto-optimal solutions. At each time, we randomly generate a string S^a, which indicates a segment of a solution (line 15). The length of S^a is $k' \in [4, k^D]$. Then, we generate another string S^b by randomly changing the positions of two elements in S^a except for the first and the last elements (lines 16 and 17). The elements in L_m^D related to S^b are set to the same values as those in S^a (lines 18–20). That makes segments S^a and S^b have equal contributions to the objective values. A solution that contains S^a and a solution that contains S^b may be equivalent Pareto-optimal solutions.

In lines 22 and 23, $L_m^D, m = 1, \ldots, M$, is multiplied by a scaling factor p_s, which is calculated by

$$p_s = \frac{\sum_{i=1}^{k^C} l_i^{\min C}}{k^C} \cdot \frac{M \cdot k^D}{\sum_{m=1}^{M} \sum_{j=1}^{k^D} l_{m,j}^{\min D}} \tag{3}$$

where $l_i^{\min C}$ is the minimum value of elements in the ith row of L_1^C except for the diagonal element, and $l_{m,j}^{\min D}$ is the minimum value of elements in the jth row of L_m^D except for the diagonal element. Multiplying p_s makes the values of elements in L_m^C and L_m^D similar. If the values of elements in L_m^C are much smaller than those in L_m^D, edges between C-type vertices have only minor effects on the objective values. Conversely, if the values of elements in L_m^C are much larger than those in L_m^D, edges between D-type vertices have only minor effects on the objective values.

Step 3 (Generating L_m^{C-D} and L_m^{D-C}): We use Algorithm 2 to generate L_m^{C-D}. Denote an optimal solution to the single-objective TSP (chosen in step 1) as $\boldsymbol{x}^S = (x_1^S, x_2^S, \ldots, x_{k^C}^S, x_1^S)$. In line 1, let $l_{\max}$ be the largest value in $L_1^C \cup L_1^D \cup \ldots \cup L_M^D$.

Algorithm 2 Generating $L_m^{C-D}, m = 1, \ldots, M$

Require: L_1^C, $L_m^D, m = 1, \ldots, M$, $\boldsymbol{x}^S = (x_1^S, x_2^S, \ldots, x_{k^C}^S, x_1^S)$ (an optimal solution of the single TSP in Step 1)

1: Set $l_{\max}$ to be the largest value in $L_1^C \cup L_1^D \cup \ldots \cup L_M^D$;
2: Randomly choose vertices c_1 and c_2 from C-type vertices under the following constraints: (1) (c_1, c_2) is an edge in $\boldsymbol{x}^S$; (2) if there exist other optimal solutions to the single-objective TSP, they do not contain (c_1, c_2).
3: Randomly choose vertices d_1 and d_2 from D-type vertices.
4: **for** $m = 1, \ldots, M$ **do**
5: Set $l_m(c_1, d_1)$ and $l_m(c_2, d_2)$ to random values in $(0, 2p_s/M)$. p_s is calculated by (3);
6: $L_m^X = L_m^{C-D} - \{l_m(c_1, d_1), l_m(c_2, d_2)\}$;
7: **for all** $l^X \in L_m^X$ **do**
8: Set l^X to be a random value in $(3l_{\max}, 4l_{\max})$;
9: **end for**
10: **end for**
11: **return** $L_m^{C-D}, m = 1, \ldots, M$

In line 2, we randomly choose vertices c_1 and c_2 from C-type vertices under the following constraints: 1) (c_1, c_2) is an edge in $\boldsymbol{x}^S$ and 2) if there exist other optimal solutions to the single-objective TSP, they do not contain (c_1, c_2). In line 3, we randomly choose vertices d_1 and d_2 from D-type vertices. In line 5, $l_m(c_1, d_1)$ and $l_m(c_2, d_2)$ are set to random values in $(0, 2p_s/M)$ [see p_s in (3)]. In lines 6–9, the other elements in L_m^{C-D} (denoted as L_m^X) are set to random values in $(3l_{\max}, 4l_{\max})$. The general idea of generating L_m^{C-D} is setting $l_m(c_1, d_1)$ and $l_m(c_2, d_2)$ to small values and setting the other elements to very large values. We transpose L_m^{C-D} to obtain L_m^{D-C}.

Without loss of generality, let x_1^S be c_1 and $x_{k^C}^S$ be c_2. The reason of setting elements L_m^{C-D} and L_m^{D-C} to those values is to make sure that every Pareto-optimal solution to the MMTSP contains segment $(d_1, c_1, x_2^S, \ldots, x_{k^C-1}^S, c_2, d_2)$. Then, we can quickly obtain the Pareto-optimal solutions.

B. Obtaining Pareto-Optimal Solutions

This section explains how to derive the Pareto-optimal solutions from the generated MMTSP. We have the following statements.

Statement 1: Every Pareto-optimal solution must contain edges (c_1, d_1) and (c_2, d_2) and must not contain any other edge that links a C-type vertex and a D-type vertex.

Statement 2: Every Pareto-optimal solution must contain segment $(c_1, x_2^S, \ldots, x_{k^C-1}^S, c_2)$. That segment is related to the path of the optimal solution $\boldsymbol{x}^S$ to the single-objective TSP.

We prove these statements in Section S-I in the supplementary material. According to Statements 1 and 2, we can quickly obtain the Pareto-optimal solutions. Fig. 6 illustrates the structure of a Pareto-optimal solution to our MMTSP, where the numbers of C- and D-type vertices are 8 and 5, respectively. The path of C-type vertices is related to an optimal solution of the single-objective TSP. The two edges between C- and D-type vertices are (c_1, d_1) and (c_2, d_2). We first enumerate all possible solutions that satisfy Statements 1 and 2. That is, we enumerate all possible permutations of D-type vertices (except for d_1 and d_2). The number of all possible solutions is $(k_D-2)!$. Then, we apply a nondominated ranking method [26] to obtain the Pareto-optimal solutions. The time complexity of our approach is $O(M(k_D!)^2)$.

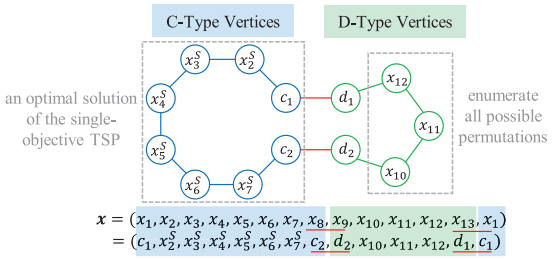


Fig. 6. Structure of a Pareto-optimal solution to our MMTSP, where the numbers of C- and D-type vertices are 8 and 5, respectively.

A brute force approach is usually employed to obtain the Pareto-optimal solutions to an MMTSP that is not generated by our generator. That is, we enumerate all the possible solutions and apply a nondominated ranking method to obtain the Pareto-optimal solutions among those solutions. The number of all the possible solutions is $(n-1)!/2$. Thus, the time complexity of the brute force approach is $O(M(n!)^2)$.

When $k_D \ll n$, our approach is much faster than the brute force approach. For example, when $k_D = 10$ and $n = 100$, $(k_D - 2)! = 4.032 \times 10^4$ solutions should be enumerated by our test problem generator, while $(n-1)!/2 \approx 4.667 \times 10^{155}$ solutions should be enumerated by the brute force approach.

It is worth noting that we have another statement as follows.

Statement 3: When $p_c = 1$ (refer to step 2 in Section III-A), the number of Pareto-optimal solutions is $(k_D - 2)!$, i.e., the number of all the possible solutions that satisfy Statements 1 and 2.

In Section S-I in the supplementary material, we prove that when $p_c = 1$, all the possible solutions that satisfy Statements 1 and 2 are nondominated by each other. Thus, they are Pareto-optimal solutions. When $p_c < 1$, those solutions are not necessarily nondominated by each other. Then, the number of Pareto-optimal solutions is usually smaller than $(k_D - 2)!$.

Section S-IX in the supplementary material discusses more characteristics of the problems generated by our test problem generator.

IV. Proposed Evolutionary Algorithm

This section proposes a novel evolutionary algorithm called TSPXEA to solve MMTSPs. Algorithm 3 gives the general framework of TSPXEA. We first initialize population Pop by randomly generating N solutions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ (line 1). Then, the number of fitness evaluations, FE, is set to N (line 2). While FE is smaller than the maximum number of fitness evaluations $\text{FE}_{\max}$ (lines 3 and 7), we generate offspring Pop′ by two new EAX operators (line 4) and form a new population by environmental selection (line 5). FE increases by |Pop′| in each generation (line 6). Finally, the algorithm returns Pop (line 8).

Algorithm 3 General Framework of TSPXEA

Require: N (population size), $FE_{\max}$ (maximum number of fitness evaluations)

```
Initialize Pop = {x_1, ..., x_N};
FE = N
while FE < FE_max do
    Pop' = Reproduction (Pop)
    Pop = Environmental_Selection (Pop, Pop');
    FE = FE + |Pop'|;
end while
return Pop
```

The new EAX operators and environmental selection are the main features of our algorithm. We explain them in the following two sections.

A. New Edge Assembly Crossover

EAX was proposed by Nagata [4], and it was further improved in some [39], [42]. EAX has been proven to be a powerful genetic operator for solving single-objective TSPs. An illustration of EAX from [39] is shown in Fig. S.26 in the supplementary material. The main steps of EAX are as follows. Let P_A and P_B be a pair of selected parents. First, *AB-cycles* are formed, where edges in P_A and edges in P_B are alternately linked in each cycle. Then, an *E-set* is constructed by selecting some *AB-cycles* according to a given selection strategy, where an *E-set* is defined as the union of *AB-cycles*. Next, an intermediate solution from P_A (or P_B) is generated by removing the edges in P_A (or P_B) and adding the edges in P_B (or P_A) in the *E-set*. Finally, an offspring solution is generated from the intermediate solution by connecting all subtours into a tour. Please refer to [39] for the details of EAX.

Although EAX is successful in solving single-objective TSPs, it cannot be directly applied to multiobjective TSPs as well as MMTSPs. In EAX, when generating an offspring solution from the intermediate solution, 4-tuples of edges $E^{4*} = \{e^*, e'^*, e''^*, e'''^*\}$ should be found to connect each subtour (see [39, step 5-3]). Those edges are obtained by

$$E^{4*} = \arg\min\left\{-d(e) - d\left(e'\right) + d\left(e''\right) + d\left(e'''\right)\right\} \quad (4)$$

where e and e' are two edges to be removed, and e'' and e''' are two edges to be added to connect the breakpoints. Note that $e \in U$ and $e' \in E_C \setminus U$, where U is the set of edges in the subtour and E_C is the set of edges in the intermediate solution. $d(e)$ is the length of edge e. That is, $d(e)$ is $l_1(e)$ in (2) when there is only one objective ($M = 1$). When there are multiple objectives, we cannot obtain E^{4*} by (4).

We propose two methods of finding E^{4*} to apply EAX in solving MMTSPs. The first method is called EAX-W, where W indicates weight. It uses the weighted sum function to obtain E^{4*} as follows:

$$E^{4*} = \arg\min\left\{-f^W(e) - f^W\left(e'\right) + f^W\left(e''\right) + f^W\left(e'''\right)\right\} \quad (5)$$

where $f^W(e) = w_1 l_1(e) + \cdots + w_M l_M(e)$, and $w_1, \ldots, w_M$ are given weights (M is the number of objectives). We can set $w_1 = \cdots = w_M = 1$ if we assume every objective is equally important. We can set $w_i = 1$ ($i \in 1, \ldots, M$) and $w_j = 0$ ($i \neq j$) if we focus on the mth objective.

The time consumption of EAX-W is low, and it is good at searching for solutions close to the Pareto front. However, it is likely to lose diversity since it forces the population to focus on searching one direction in the objective space.

The other method is called EAX-ND, where ND indicates nondominated. EAX-ND is expected to generate offspring that promotes both convergence and diversity. Denote a candidate of E^{4*} as $E_i^4 = \{e_i, e_i', e_i'', e_i'''\}, i = 1, \ldots, n_c$, where n_c is the number of candidates. Let $f_m^{\mathrm{ND}}(E_i^4) = -l_m(e_i) - l_m(e_i') + l_m(e_i'') + l_m(e_i'''), m = 1, \ldots, M$. We define that E_a^4 dominates E_b^4 ($a, b \in \{1, \ldots, n_c\}$) if $f_m^{\mathrm{ND}}(E_a^4) \leq f_m^{\mathrm{ND}}(E_b^4)$ for every objective and $f_m^{\mathrm{ND}}(E_a^4) < f_m^{\mathrm{ND}}(E_b^4)$ for at least one objective.

If there is only one nondominated candidate, we choose it as E^{4*}. If there are multiple nondominated candidates, we choose one of them as follows:

$$E^{4*} = \arg\min_{i \in \{1,\ldots,n_c\}, E_i^4 \text{ is nondominated}} \left\{f^C\left(E_i^4\right)\right\}$$
$$f^C\left(E_i^4\right) = -c(e_i) - c\left(e_i'\right) + c\left(e_i''\right) + c\left(e_i'''\right) \quad (6)$$

where $c(e_i)$ is the number of solutions that contain e_i in the population. Choosing a nondominated candidate with the minimum value of f^C as E^{4*} may lead to an offspring that is dissimilar to solutions in the population. If there are multiple nondominated candidates with the same minimum value of f^C, we randomly choose one of them as E^{4*}.

Suppose that we have N_e candidates of E^{4*}, the time complexity of obtaining E^{4*} by (5) in EAX-W is $O(MN_e)$, where M is the number of objectives. In EAX-ND, the time complexity of obtaining the nondominated candidates of E^{4*} is $O(MN_e^2)$. If all the candidates are nondominated, the time complexity of obtaining E^{4*} by (6) is $O(nNN_e)$, where n is the number of vertices of the MMTSP and N is the population size. The overall time complexity of EAX-ND is $O(nNN_e + MN_e^2)$. Thus, using EAX-ND with a huge population size to solve a large-scale problem is likely to consume much more time than using EAX-W. Section S-II in the supplementary material analyzes the running time of EAX-W and EAX-ND.

Algorithm 4 describes how to generate offspring by EAX-W and EAX-ND.

In line 1, the set of offspring Pop$'$ is empty. In line 2, let $r(\cdot)$ be a random permutation of $1, \ldots, N$. For $i = 1, \ldots, N$, we choose $P_A = \boldsymbol{x}_{r(i)}$ and $P_B = \boldsymbol{x}_{r(i+1)}$ be a pair of parents (lines 3–5). Note that $r(N+1) = r(1)$. When FE $< \eta \cdot \mathrm{FE}_{\max}$ ($\eta \in [0, 1]$), we assume that the evolution is in the early stage and a fast convergence rate is desirable (line 6). Thus, we apply EAX-W to generate an offspring set O (line 7). In the later stage of evolution, we apply EAX-ND to promote population diversity (lines 8 and 9). O is moved into Pop$'$ in each iteration (line 11). The maximum size of O is set to 2 in this article. Note that O may be empty since EAX does not always produce offspring. Pop$'$ is finally returned (line 13). We recommend the setting of $\eta = 0$ if we have enough computing resources. That is, we suggest the use of only EAX-ND to search for a well-distributed solution set using a long computation time. However, if we do not have enough computing resources, we

Algorithm 4 Reproduction

Require: $Pop = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ (population), $\eta \in [0, 1]$ (parameter for crossover type)

```
Pop' = ∅;
Let r(·) be a random permutation of 1, ..., N;
for i = 1, ..., N do
    P_A = x_{r(i)};
    P_B = x_{r(i+1)};                    ▷ r(N + 1) = r(1)
    if FE < η · FE_max then
        O = EAX-W (P_A, P_B);
    else
        O = EAX-ND (P_A, P_B);
    end if
    Pop' = Pop' ∪ O;
end for
return Pop'
```

can set $\eta > 0$ for a fast convergence rate. Please refer to Section V-C1 for the investigation on the setting of η.

B. Environmental Selection

This section explains how to choose a new population in environmental selection. In multimodal multiobjective optimization, we usually want to obtain a solution set that is good in three aspects: 1) convergence to the Pareto front; 2) diversity in the objective space; and 3) diversity in the decision space. In our environmental selection operator, we use 1) nondominated ranking; 2) k-nearest neighbor; and 3) entropy to address the three aspects, respectively.

There are two selection criteria in our environmental selection operator. The primary selection criterion is nondominated ranking based on the Pareto dominance relation. Solutions in lower ranks are selected preferentially. The secondary selection criterion based on k-nearest neighbor and entropy is applied to solutions in the same rank.

We evaluate a solution's contribution to diversity in the objective space based on k-nearest neighbor as follows:

$$f^K(\boldsymbol{x}) = \sum_{k=1, \boldsymbol{x}, \boldsymbol{x}_k^{\text{nst}} \in Q}^{K} \text{distance}\left(\boldsymbol{x}, \boldsymbol{x}_k^{\text{nst}}\right) \tag{7}$$

where $\boldsymbol{x}_k^{\text{nst}}$ is the kth nearest solution to $\boldsymbol{x}$ in the candidate solution set Q, and $\text{distance}(\boldsymbol{x}, \boldsymbol{x}_k^{\text{nst}})$ refers to the Euclidean distance between the solutions in the objective space. K is set to 3 as in [50]. A larger value of $f^K(\boldsymbol{x})$ indicates that $\boldsymbol{x}$ locates in a sparser area in the objective space.

Entropy is a popular population diversity measure in combinatorial optimization. We use a solution's contribution to entropy as its contribution to the diversity in the decision space. A solution's contribution to entropy, $f^E(\boldsymbol{x})$, is obtained by

$$\begin{aligned} f^E(\boldsymbol{x}) &= -H(Q \setminus \boldsymbol{x}) \\ H(X) &= \sum_{e \in E_X} -\frac{n_e}{n|X|} \log \frac{n_e}{n|X|} \end{aligned} \tag{8}$$

where $H(X)$ is the entropy of a solution set X, E_X is the set of edges in X, n_e is the number of solutions that contain edge e in X, and n is the number of vertices (i.e., the number of total edges in a solution). $H(Q \setminus \boldsymbol{x})$ means the entropy of the candidate solution set Q without $\boldsymbol{x}$. A smaller value of $H(Q \setminus \boldsymbol{x})$ implies that $\boldsymbol{x}$ is more important to entropy. That is, if we remove such $\boldsymbol{x}$ from Q, it is likely to obtain a solution set with poor diversity in the decision space. Therefore, we prefer a solutions with the largest value of $f^E(\boldsymbol{x})$.

In the secondary selection criterion, we use both $f^K(\boldsymbol{x})$ and $f^E(\boldsymbol{x})$ to evaluate a solution. Because $f^K(\boldsymbol{x})$ and $f^E(\boldsymbol{x})$ are in different scales, it is inappropriate to simply sum them up. We use the following equation for evaluation:

$$f^{KE}(\boldsymbol{x}) = \frac{f^K(\boldsymbol{x}) - f^K_{\text{mean}}}{f^K_{\text{std}}} + \frac{f^E(\boldsymbol{x}) - f^E_{\text{mean}}}{f^E_{\text{std}}} \tag{9}$$

where f^K_{mean} and f^E_{mean} are the mean values of $f^K(\boldsymbol{x})$ and $f^E(\boldsymbol{x})$ over all solutions in Q, respectively, and f^K_{std} and f^E_{std} are the standard deviations of $f^K(\boldsymbol{x})$ and $f^E(\boldsymbol{x})$ over all solutions in Q, respectively. That is, we first normalize $f^K(\boldsymbol{x})$ and $f^E(\boldsymbol{x})$ into similar scales and then sum them up. A solution with the smallest value of f^{KE} is regarded as the worst solution.

Please refer to Section V-C2 for the investigation of the effectiveness of our proposed environmental selection operator.

V. Experiments

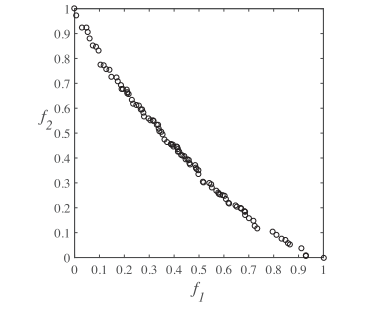
A. Generating Test Problems

In this section, we use our test problem generator to derive MMTSPs with two objectives for experiments. We choose "bays29," "att48," "lin105," and "a280" from TSPLIB [49] as the base single-objective TSP to generate L_1^C and L_2^C. Thus, the number of C-type vertices, k_C, can be 29, 48, 105, and 280 depending on the choice of a single-objective TSP from them. In addition, we also examine a special case of $k_C = 1$, where the C-type vertex is regarded as both c_1 and c_2. We denote this case as "uni1." Generally, the larger the value of k_C, the more difficult it is to search for Pareto-optimal solutions.

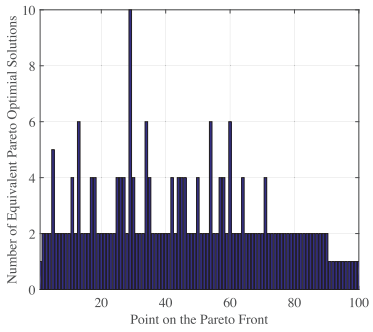
We generate 10 sets of D-type vertices by setting $k_D = 10$, $p_c = 0.9$, $p_m = 5$, and $k' = 5$, and denote them as "D1" to "D10." We combine "DX" (X is from 1 to 10) with "uni1," "bays29," "att48," "lin105," and "a280," respectively. Note that p_s in (3) is calculated from each set of C-type vertices. The five generated MMTSP sets are denoted as "uni1+DX," "bays29+DX," "att48+DX," "lin105+DX," and "a280+DX." The number of vertices of the five MMTSPs (i.e., their scales) is 11, 39, 58, 115, and 290, respectively. The Pareto fronts of the five MMTSPs are the same in the normalized objective space. Note that the edge lengths in these problems are rounded to integers for easy computation and increasing multimodality. This also increases the possibility of equivalent Pareto-optimal solutions. Table I shows the number of points on the Pareto front (N_{PF}) and the number of Pareto-optimal solutions (N_{PS}) of the MMTSPs, where "DX" indicates the MMTSPs whose D-type vertices are "DX." Fig. 7(a) shows the Pareto front (normalized to [0, 1]) of an MMTSP whose D-type vertices are "D1." Fig. 7(b) shows the number of equivalent Pareto-optimal solutions corresponding to each point on the Pareto front. We show the corresponding figure for each of the other MMTSPs as Figs. S.27–S.35 in the supplementary material.

TABLE I
NUMBER OF POINTS ON THE PARETO FRONT (N_{PF}) AND THE NUMBER OF PARETO-OPTIMAL SOLUTIONS (N_{PS}) OF OUR MMTSPS

	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10
N_{PF}	100	149	159	166	242	268	377	446	515	560
N_{PS}	250	258	262	411	463	495	1030	862	1564	1882



(a)



(b)

Fig. 7. (a) Pareto front (normalized to $[0, 1]$) of an MMTSP whose D-type vertices are "D1." (b) Number of equivalent Pareto-optimal solutions corresponding to each point on the Pareto front.

In the following subsections, we use those MMTSPs to test the performance of our evolutionary algorithm.

B. Performance Indicators

In multimodal multiobjective optimization, we usually need to apply multiple indicators to measure the performance of algorithms in different perspectives. We use the following performance indicators to evaluate the performance of a solution set obtained by an optimizer: inverted generational distance-multimodal (IGDM) [11], hypervolume (HV) [51], and generational distance (GD) [52]. In addition, we have included the results of IGD [52] and IGDX [53] for the completeness of the experimental study in the supplementary material.

Among existing performance indicators, IGDM is the only overall performance indicator for examining the convergence and diversity ability in both the objective and decision spaces. IGDM requires a set of well-distributed reference points on the Pareto front in the objective space. IGDM also requires equivalent Pareto-optimal solutions in the decision space corresponding to each reference point. In IGDM, the distance between each reference point and each solution in the solution set is calculated in both the objective and decision spaces (in the decision space, each of the corresponding equivalent Pareto-optimal solutions is used). In this article, the distance between two solutions in the decision space is the number of different edges in these solutions. It is worth noting that such an overall performance indicator usually has a bias [54], [55]. Developing other overall performance indicators for multimodal multiobjective optimization is an interesting future research topic.

TABLE II
AVERAGE PERFORMANCE SCORES OF TSPXEA WITH DIFFERENT SETTINGS OF η ACCORDING TO IGDM, HV, AND GD

	η =	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
IGDM	uni1	1.7	0.7	1.6	0.9	1	0.7	0.8	0.8	0.7	1.2	0
	att48	1.6	1.1	1.3	1.3	1.3	0.8	1	0.8	0.9	0.9	0
	lin105	8.4	6.6	0.2	0.6	0.1	0.3	0.6	0.2	0.8	0.1	0
	a280	7.3	6.6	6.2	0.3	0.4	0.1	0.3	0.3	0.3	0.6	1.8
HV	uni1	1.5	1.3	1.9	1	1.8	1.2	1.6	0.9	1.3	1	0
	att48	1.3	0.8	1	0.8	0.9	0.9	0.9	0.8	0.8	0.9	0
	lin105	8.6	6.6	0.2	0.5	0.1	0.4	0.6	0.2	0.9	0.1	0
	a280	8.4	8.4	7.3	0.2	0.2	0.1	0.3	0.2	0.2	0.6	1.4
GD	uni1	1	0.5	1	0.2	0.2	0.2	0.3	0.1	0.4	0.3	3.6
	att48	1.1	0.6	0.4	0.3	0.2	0.5	0.9	0.2	0.3	0.2	2.7
	lin105	8.4	6	0.1	0.4	0	0.2	0.3	0.1	0.5	0	0.3
	a280	7.3	7.6	7.1	0.1	0.5	0.1	0.1	0.4	0.1	0.3	1.6

In addition to IGDM, we apply some indicators to compare the algorithms in the objective or decision space. HV and IGD simultaneously measure the convergence and diversity in the objective space of a solution set. On the other hand, IGDX measures the convergence and diversity in the decision space of a solution set. GD assesses a solution set's convergence to the Pareto front. Smaller IGDM, smaller IGD, smaller IGDX, smaller GD, and larger HV mean a better solution set.

Before calculating the above performance indicators, the objective space of each test problem is normalized so that the true ideal and nadir points are $(0, 0)$ and $(1, 1)$, respectively. The reference point for HV is $(1.1, 1.1)$. We use the normalized HV value of a solution set, which is its HV value divided by the HV value of the true Pareto front. Therefore, the normalized HV value is in $[0, 1]$.

We run each optimizer on a test problem 30 independent times. Then, based on the results of 30 runs, we used Wilcoxon's rank-sum test at a level of 0.05 to determine whether an optimizer performs statistically significantly better than another according to a performance indicator.

C. Investigation on the Proposed Evolutionary Algorithm

In this section, we first investigate the search behavior of TSPXEA with different settings of η, and then we show the effectiveness of our environmental selection operator. The population size N is set to 100, and the maximum number of evaluated solutions, FE_{max}, is set to 50 000.

1) Effect of η: Table II shows the average performance scores of TSPXEA with different settings of η according to IGDM, HV, and GD. The performance score of TSPXEA with a specific setting of η is the number of the algorithms with other settings of η which perform significantly worse than it with respect to a performance indicator. The average performance score on "uni1" means the average value of the performance scores on 10 test problems of the "uni1" series (i.e., uni1+D1 to uni1+D10). The average performance scores

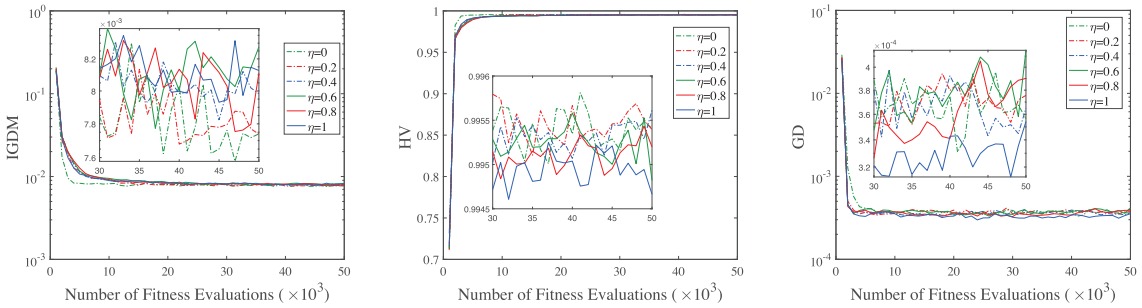


Fig. 8. Average IGDM, HV, and GD values over 30 runs with regard to the number of evaluated solutions on uni1+D1. (a) IGDM. (b) HV. (c) GD.

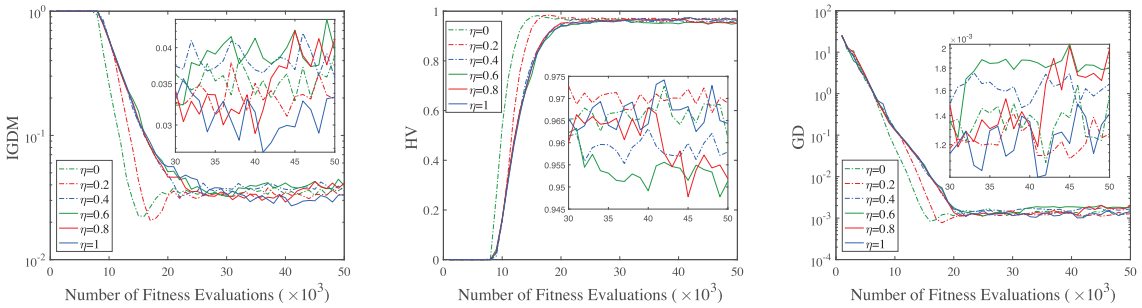


Fig. 9. Average IGDM, HV, and GD values over 30 runs with regard to the number of evaluated solutions on a280+D1. (a) IGDM. (b) HV. (c) GD.

on "att48," "lin105," and "a280" have similar meanings. Please refer to Tables S.X–S.XII in the supplementary material for the details of those results.

From those tables, we can see that setting η to a medium value (in $[0.3, 0.9]$) does not clearly affect the results. Setting $\eta = 1$ (i.e., use of EAX-W only) may yield excellent GD values, especially, on small-scale test problems, such as uni1+D2, uni1+D4, uni1+D8, att48+D2, att48+D5, and att48+D8. This indicates that applying only EAX-W can occasionally result in a solution set with excellent convergence performance at the end of evolution. However, for the large-scale test problems, a small value of η usually leads to superior overall performance at the end of evolution, such as $\eta \leq 0.1$ for the "lin105" series and $\eta \leq 0.2$ for "lin280" series. The reason is that EAX-ND focus on promoting both convergence and diversity, while applying EAX-W may result in losing population diversity and trapping in local optimal regions.

Fig. 8 show the mean IGDM, HV, and GD values over 30 runs with regard to the number of evaluated solutions on uni1+D1, where η is set to $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Fig. 9 shows the results on a280+D1 in the same way. From those figures, we have the following observations: 1) on uni1+D1, EAX-W (i.e., $\eta = 1$) leads to a fast decrease in GD while EAX-ND (i.e., $\eta = 0$) leads to a fast decrease in IGDM and a rapid increase of HV at the early stage of evolution; 2) on a280+D1, EAX-W (i.e., $\eta = 1$) shows slower performance improvement than EAX-ND (i.e., $\eta = 0$) does for all performance indicators; and 3) on both uni1+D1 and a280+D1, all the curves oscillate within a small range at the later stage of evolution. The difference among the algorithms with different settings of η is not significant. These observations indicate that 1) EAX-W is better than EAX-ND in convergence on a small-scale test problem and 2) EAX-ND is better than EAX-W in both convergence and diversity on a large-scale test problem.

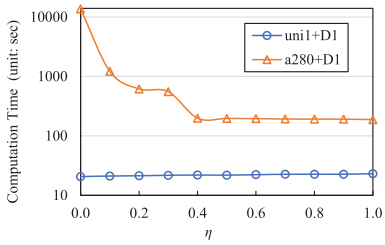


Fig. 10. Average computation time (unit: second) over 30 runs under the different settings of η on uni1+D1 and a280+D1.

However, there is a price for the good performance of EAX-ND on large-scale test problems. Fig. 10 shows the average computation time (unit: second) over 30 runs under the different settings of η on uni1+D1 and a280+D1. We can see from Fig. 10 that 1) the setting of η does not clearly affect the computation time for a small-scale test problem (i.e., uni1+D1) and 2) a small value of η significantly increase the computation time for a large-scale test problem (i.e., a280+D1). Setting $\eta = 0$ (i.e., use of EAX-ND only) results in extremely high time consumption. The reason is that *AB-cycles* usually have many edges at the early stage of evolution for a large-scale test problem. Then, EAX-ND spends many computing resources on nondominated ranking and calculating f^C in (6).

TABLE III
RESULTS OBTAINED BY TSPXEA WITH DIFFERENT SETTINGS OF η ON A280+D1 UNDER A SEVERELY LIMITED COMPUTATION TIME

a280+D1	η=0	η=0.1	η=0.2	η=0.3	η=0.4	η=0.5	η=0.6	η=0.7	η=0.8	η=0.9	η=1
IGDM	NaN	1.000E+0 0	1.000E+0 0	1.983E-1 2	4.508E-2 3	2.858E-2 8	3.326E-2 5	3.561E-2 4	3.728E-2 4	3.556E-2 4	3.172E-2 6
HV	NaN	0.000E+0 0	0.000E+0 0	6.366E-1 2	9.258E-1 3	9.697E-1 7	9.646E-1 5	9.567E-1 4	9.571E-1 4	9.604E-1 4	9.682E-1 7
GD	NaN	4.802E+0 0	6.354E-1 1	3.727E-2 2	5.101E-3 3	1.390E-3 4	1.488E-3 4	1.731E-3 4	1.603E-3 4	1.594E-3 4	1.095E-3 8

TABLE IV
RESULTS OBTAINED BY TSPXEA AND TSPXEA* ON UNI1+D1 TO UNI1+D10

	IGDM		HV		GD	
	TSPXEA	TSPXEA*	TSPXEA	TSPXEA*	TSPXEA	TSPXEA*
uni1+D1	7.819E-3	1.364E-2 +	9.954E-1	9.853E-1 +	3.644E-4	6.455E-4 +
uni1+D2	1.676E-2	2.773E-2 +	9.846E-1	9.615E-1 +	1.505E-3	1.490E-3 -
uni1+D3	4.069E-3	1.123E-2 +	9.981E-1	9.909E-1 +	2.363E-4	6.749E-4 +
uni1+D4	9.826E-3	2.466E-2 +	9.944E-1	9.719E-1 +	4.788E-4	9.306E-4 +
uni1+D5	8.788E-3	2.284E-2 +	9.943E-1	9.749E-1 +	1.038E-4	9.377E-4 +
uni1+D6	9.494E-3	2.309E-2 +	9.925E-1	9.637E-1 +	3.443E-4	1.622E-3 +
uni1+D7	7.311E-3	2.134E-2 +	9.899E-1	9.713E-1 +	5.984E-4	1.165E-3 +
uni1+D8	8.284E-3	1.295E-2 +	9.930E-1	9.883E-1 +	2.806E-4	2.917E-4 =
uni1+D9	9.066E-3	4.191E-2 +	9.934E-1	9.676E-1 +	0.000E+0	3.395E-4 +
uni1+D10	8.993E-3	3.031E-2 +	9.933E-1	9.765E-1 +	8.547E-5	3.213E-4 =
+/=/-		10/0/0		10/0/0		7/2/1

It is interesting to investigate the performance of TSPXEA on a large-scale problem under a severely limited computation time. Table III shows the average IGDM, HV, and GD values and the performance scores obtained by TSPXEA with different settings of η on a280+D1, where the termination condition is the computation time reaches 100 s (i.e., Algorithm 3, line 3). The condition of using EAX-W is that the computation time is less than $100 \cdot \eta$ s (i.e., Algorithm 4, line 6). We can see from Table III that the results of setting $\eta = 0$ (i.e., use of EAX-ND only) are NaN. The algorithm cannot even produce a new population in the first generation in 100 s. A larger value of η generally leads to a better GD value. The setting of $\eta = 1$ (i.e., use of EAX-W only) obtains the best GD value. This indicates that EAX-W is good at promoting convergence under a severely limited computation time. The results of IGDM and HV are also acceptable with $\eta = 1$. The setting of $\eta = 0.5$ leads to the best IGDM and HV values, which means that properly applying both EAX-W and EAX-ND can result in the best overall performance.

To sum up, we suggest setting $\eta = 0$ if we have enough computing resources. Otherwise, we can tune $\eta \in (0, 1)$ to obtain a good balance between the computing resources and the performance. In the following experiments, we set $\eta = 0$.

2) Effect of Environmental Selection: To the best of our knowledge, DNEA-TSP [3] is the only one that can handle MMTSPs except for TSPXEA. Therefore, we compare their environmental selection operators in this section. Both operators use nondominated ranking as the primary selection criterion. The difference between them is their secondary selection criteria. That is, the difference is how they evaluate a solution's contribution to population diversity. Here, we replace the environmental selection operator with that of DNEA-TSP in our algorithm. The new algorithm is denoted as TSPXEA*. Table IV shows the results obtained by TSPXEA and TSPXEA* on uni1+D1 to uni1+D10. "+"/"−" indicates that TSPXEA performs statistically significantly better/worse than TSPXEA*, and "=" means that there is no statistically significant difference between them. We do not show the results of other test problems since diversity is our primary concern.

We can see from Table IV that TSPXEA obtained significantly better IGDM and HV results than those of TSPXEA* on all test problems. This means that TSPXEA can maintain good diversity in both the objective and decision spaces than TSPXEA* does. TSPXEA also performs better than TSPXEA* with respect to GD on several test problems, which indicates that good diversity may lead to good convergence.

D. Comparing With Other Algorithms

1) Compared Algorithms: We compare our TSPXEA with the following five state-of-the-art algorithms: DNEA-TSP [3], EAG [45], DCDG [48], DYN [41], and NMA [1]. These algorithms have been briefly introduced in Section II. DNEA-TSP is the only existing optimizer designed to solve MMTSPs, except for TSPXEA.

DYN was designed to solve single-objective TSPs by enhancing the diversity in the decision space. Here, it is easy to modify it to handle multiple objectives. To solve an M-objective MMTSP, DYN optimizes $M + 1$ objectives, i.e., M original objectives and one objective about the diversity in the decision space.

NMA is a multimodal single-objective TSP optimizer. We apply NMA to solve an MMTSP by the following steps: 1) we generate a set of weight vectors that are uniformly distributed in the objective space; 2) based on each weight vector, we transform the MMTSP into a single-objective problem by the weighted sum method; 3) we apply NMA to solve each single-objective problem; and 4) the union of the final solution set obtained by solving each single-objective problem is the solution set to the MMTSP.

We do not modify anything in EAG and DCDG. An MMTSP is essentially a MMOP. EAG and DCDG are typical multiobjective optimization algorithms. We want to observe the performance of these algorithms on MMTSPs. However, in Section S-VIII in the supplementary material, we modify EAG and DCDG by adding an archive of an unlimited size that collects all the nondominated solutions during the evolutionary process. We want to observe whether they can obtain equivalent Pareto-optimal solutions with such a modification.

In all the compared algorithms, the population size N is set to 100, and the maximum number of evaluated solutions $\mathrm{FE}_{\max}$ is set to 50 000. Note that we generate 100 weight vectors for NMA. Therefore, the total number of solutions evaluated by NMA is $100 \times \mathrm{FE}_{\max} = 5\,000\,000$. Although NMA is given more computing resources than the others, we show that the proposed TSPXEA performs better than NMA.

TABLE V
RESULTS OF IGDM OBTAINED BY THE COMPARED ALGORITHMS

IGDM	TSPXEA	DNEA-TSP	EAG	DCDG	DYN	NMA
uni1+D1	7.819E-3	7.795E-3 =	1.626E-2 +	8.920E-2 +	3.635E-2 +	5.111E-2 +
uni1+D2	1.676E-2	5.141E-3 -	1.501E-2 =	7.348E-2 +	2.982E-2 +	4.275E-2 +
uni1+D3	4.069E-3	3.219E-3 -	1.488E-2 +	4.353E-2 +	2.385E-2 +	3.130E-1 +
uni1+D4	9.826E-3	7.943E-3 -	1.554E-2 +	6.460E-2 +	3.349E-2 +	7.992E-2 +
uni1+D5	8.788E-3	7.437E-3 -	1.434E-2 +	2.076E-2 +	3.051E-2 +	3.194E-2 +
uni1+D6	9.494E-3	5.306E-3 -	1.436E-2 +	4.408E-2 +	3.750E-2 +	1.264E-1 +
uni1+D7	7.311E-3	5.777E-3 -	9.164E-3 +	5.221E-2 +	2.685E-2 +	4.008E-1 +
uni1+D8	8.284E-3	6.163E-3 -	1.125E-2 +	4.366E-2 +	2.811E-2 +	2.067E-1 +
uni1+D9	9.066E-3	8.248E-3 -	2.070E-2 +	5.930E-2 +	2.772E-2 +	2.398E-2 +
uni1+D10	8.993E-3	7.825E-3 -	1.344E-2 +	5.731E-2 +	2.547E-2 +	7.080E-2 +
+/=/- (uni1)		0/1/9	9/1/0	10/0/0	10/0/0	10/0/0
bays29+D1	8.168E-3	1.518E-1 +	9.440E-1 +	2.833E-1 +	3.091E-1 +	3.525E-1 +
bays29+D2	1.695E-2	2.186E-1 +	9.284E-1 +	4.230E-1 +	3.900E-1 +	7.042E-1 +
bays29+D3	4.348E-3	3.690E-1 +	9.339E-1 +	4.727E-1 +	4.108E-1 +	7.919E-1 +
bays29+D4	8.332E-3	3.566E-1 +	8.050E-1 +	4.918E-1 +	3.263E-1 +	2.644E-1 +
bays29+D5	9.736E-3	2.152E-1 +	8.456E-1 +	6.931E-1 +	3.914E-1 +	5.550E-1 +
bays29+D6	9.299E-3	2.503E-1 +	9.114E-1 +	5.433E-1 +	3.773E-1 +	6.088E-1 +
bays29+D7	6.512E-3	4.095E-1 +	8.922E-1 +	1.585E-1 +	2.620E-1 +	2.505E-1 +
bays29+D8	1.001E-2	4.070E-1 +	9.005E-1 +	6.445E-1 +	4.370E-1 +	6.907E-1 +
bays29+D9	9.415E-3	3.292E-1 +	8.254E-1 +	5.620E-1 +	3.051E-1 +	4.906E-1 +
bays29+D10	9.462E-3	5.035E-1 +	9.486E-1 +	7.239E-1 +	3.649E-1 +	4.551E-1 +
+/=/-(bays29)		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0
att48+D1	8.160E-3	6.602E-1 +	1.000E+0 +	5.335E-1 +	9.101E-1 +	1.000E+0 +
att48+D2	1.726E-2	7.324E-1 +	1.000E+0 +	6.657E-1 +	9.942E-1 +	1.000E+0 +
att48+D3	5.609E-3	7.430E-1 +	1.000E+0 +	5.254E-1 +	9.903E-1 +	1.000E+0 +
att48+D4	8.197E-3	6.453E-1 +	9.675E-1 +	4.862E-1 +	8.942E-1 +	9.978E-1 +
att48+D5	1.045E-2	8.297E-1 +	1.000E+0 +	6.532E-1 +	9.990E-1 +	1.000E+0 +
att48+D6	1.158E-2	8.846E-1 +	1.000E+0 +	7.376E-1 +	9.951E-1 +	1.000E+0 +
att48+D7	6.976E-3	6.610E-1 +	9.858E-1 +	5.155E-1 +	8.710E-1 +	9.999E-1 +
att48+D8	1.087E-2	8.100E-1 +	1.000E+0 +	8.325E-1 +	1.000E+0 +	1.000E+0 +
att48+D9	9.745E-3	6.652E-1 +	1.000E+0 +	8.342E-1 +	9.875E-1 +	1.000E+0 +
att48+D10	9.989E-3	7.941E-1 +	9.909E-1 +	8.500E-1 +	9.985E-1 +	1.000E+0 +
+/=/-(att48)		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0
lin105+D1	1.874E-2	1.000E+0 +	1.000E+0 +	9.118E-1 +	1.000E+0 +	1.000E+0 +
lin105+D2	4.265E-2	1.000E+0 +	1.000E+0 +	8.854E-1 +	1.000E+0 +	1.000E+0 +
lin105+D3	2.231E-2	1.000E+0 +	1.000E+0 +	9.675E-1 +	1.000E+0 +	1.000E+0 +
lin105+D4	1.181E-2	1.000E+0 +	1.000E+0 +	8.938E-1 +	1.000E+0 +	1.000E+0 +
lin105+D5	1.864E-2	1.000E+0 +	1.000E+0 +	8.853E-1 +	1.000E+0 +	1.000E+0 +
lin105+D6	2.747E-2	1.000E+0 +	1.000E+0 +	9.701E-1 +	1.000E+0 +	1.000E+0 +
lin105+D7	1.197E-2	1.000E+0 +	1.000E+0 +	7.826E-1 +	1.000E+0 +	1.000E+0 +
lin105+D8	2.075E-2	1.000E+0 +	1.000E+0 +	9.361E-1 +	1.000E+0 +	1.000E+0 +
lin105+D9	2.603E-2	1.000E+0 +	1.000E+0 +	9.673E-1 +	1.000E+0 +	1.000E+0 +
lin105+D10	1.705E-2	1.000E+0 +	1.000E+0 +	8.778E-1 +	1.000E+0 +	1.000E+0 +
+/=/-(lin105)		10/0/0	10/0/0	10/0/0	10/0/0	10/0/0

The additional parameter settings of the compared algorithms follow the suggestions in their original studies. In EGA, the neighborhood size is 10, and the number of learning generations is 8. In DCDG, the initial number of intervals on each objective is 10. In NMA, the neighborhood size is in the range of [4, 12].

2) Results: Table V shows the mean values of IGDM obtained by each algorithm. Tables S.XIII–S.XVI in the supplementary material show the mean values of HV, IGD, GD and IGDX obtained by each algorithm. In these tables, "+"/"−" indicates that TSPXEA performs statistically significantly better/worse than a compared algorithm, and "=" means that there is no statistically significant difference between them. We do not show the results on the test problems of the "a280" series since all the compare algorithms except TSPXEA perform very poorly on those problems. Please refer to the results obtained by TSPXEA on the test problems of the "a280" series in Tables S.X–S.XII in the supplementary material.

We can see from these tables that for the test problems of "bays29," "att48," and "lin105" series, TSPXEA outperforms all the compared algorithms on all the examined test problems. These results verify TSPXEA's high performance in solving MMTSPs. Each of the other algorithms seems to have some difficulties in solving those test problems. Generally, DNEA-TSP is the second-best on the test problems of the "bays29" series, while DCDG ranks the second-best on the test problems of the "att48" and "lin105" series. Note that if a solution set is far away from the Pareto front, it has the maximum value of IGDM (i.e., 1) and the minimum value of HV (i.e., 0).

The results of the test problems of the "uni1" series are interesting. It is easy for an optimizer to approximate the Pareto-optimal solutions to the test problems of the "uni1" series. We can see from Table S.XIV in the supplementary material that the GD values obtained by TSPXEA on the "uni1" series are good. However, the GD values obtained by some compared algorithms are better than TSPXEA, such as those by DNEA-TSP and NMA on uni1+D1 to uni1+D8, those by EAG on uni1+D2 and uni1+D7, and those by DCDG on uni1+D2, uni1+D3, uni1+D6, uni1+D7, and uni1+D8. We explain why the GD values obtained by TSPXEA on small-scale problems may be worse than those of the compared algorithm in Section S-VII in the supplementary material.

Section S-V in the supplementary material shows the visualized results of the solution sets obtained by the compared algorithms. Section S-VI in the supplementary material shows the convergence plots toward true Pareto fronts obtained by the compared algorithms. The results showed in these two sections further verify the superiority of TSPXEA.

VI. CONCLUSION

In this article, we propose a test problem generator for MMTSPs. It can readily generate an MMTSP with known Pareto-optimal solutions. We generate 50 test problems for experiments. We also propose a novel evolutionary algorithm named TSPXEA. TSPXEA uses two new EAX operators called EAX-W and EAX-ND, which are featured by the weighted sum and nondominated ranking methods, respectively. In the environmental selection of TSPXEA, k-nearest neighbor and entropy are combined as a selection criterion.

Our experimental results undoubtedly showed the following.

1) The use of EAX-ND only is usually better than the use of EAX-W only if there are enough computing resources. Otherwise, applying both EAX-ND and EAX-W leads to an excellent overall performance.
2) The environmental selection operator of TSPXEA can maintain a good diversity in both the objective and decision spaces.
3) TSPXEA performs significantly better than five state-of-the-art optimizers on most test problems.

One future research direction is the creation of MMTSPs with three or more objectives and the performance evaluation of the proposed algorithm on those test problems. Another direction is to extend our test problem generator to produce an MMTSP with some specific features, such as the shape of the Pareto front and the distribution of the Pareto-optimal solutions. Improving EAX-W and EAX-ND to explore a vast area where potential solutions locate is also a direction.

REFERENCES

[1] T. Huang, Y.-J. Gong, S. Kwong, H. Wang, and J. Zhang, "A niching memetic algorithm for multi-solution traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 24, no. 3, pp. 508–522, Jun. 2020.

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

[3] Y. Liu et al., "Multi-modal multi-objective traveling salesman problem and its evolutionary optimizer," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2021, pp. 770–777.

[4] Y. Nagata, "Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem," in *Proc. 7th Int. Conf. Genet. Alg.*, 1997, pp. 450–457.

[5] Y. Liu, H. Ishibuchi, Y. Nojima, N. Masuyama, and Y. Han, "Searching for local Pareto optimal solutions: A case study on polygon-based problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2019, pp. 896–903.

[6] W. Li, X. Yao, T. Zhang, R. Wang, and L. Wang, "Hierarchy ranking method for multimodal multi-objective optimization with local Pareto fronts," *IEEE Trans. Evol. Comput.*, early access, Mar. 2, 2022, doi: 10.1109/TEVC.2022.3155757.

[7] X. 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.

[8] M. Preuss, B. Naujoks, and G. Rudolph, "Pareto set and EMOA behavior for simple multimodal multiobjective functions," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2006, pp. 513–522.

[9] G. Rudolph, B. Naujoks, and M. Preuss, "Capabilities of EMOA to detect and preserve equivalent Pareto subsets," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2007, pp. 36–50.

[10] K. Deb and S. Tiwari, "Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization," *Eur. J. Oper. Res.*, vol. 185, no. 3, pp. 1062–1087, 2008.

[11] Y. Liu, G. G. Yen, and D. Gong, "A multimodal multiobjective evolutionary algorithm using two-archive and recombination strategies," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 660–674, Aug. 2019.

[12] C. Yue, B. Qu, and J. Liang, "A multiobjective particle swarm optimizer using ring topology for solving multimodal multiobjective problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 805–817, Oct. 2018.

[13] C. Yue, B. Qu, K. Yu, J. Liang, and X. Li, "A novel scalable test problem suite for multimodal multiobjective optimization," *Swarm Evol. Comput.*, vol. 48, pp. 62–71, Aug. 2019.

[14] Y. Liu, H. Ishibuchi, Y. Nojima, N. Masuyama, and K. Shang, "A double-niched evolutionary algorithm and its behavior on polygon-based problems," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2018, pp. 262–273.

[15] Y. Peng, H. Ishibuchi, and K. Shang, "Multi-modal multi-objective optimization: Problem analysis and case studies," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, 2019, pp. 1865–1872.

[16] Y. Nojima, T. Fukase, Y. Liu, N. Masuyama, and H. Ishibuchi, "Constrained multiobjective distance minimization problems," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 586–594.

[17] J. E. Fieldsend, T. Chugh, R. Allmendinger, and K. Miettinen, "A feature rich distance-based many-objective visualisable test problem generator," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 541–549.

[18] Y. Liu, H. Ishibuchi, G. G. Yen, Y. Nojima, and N. Masuyama, "Handling imbalance between convergence and diversity in the decision space in evolutionary multimodal multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 3, pp. 551–565, Jun. 2020.

[19] Y. Han, J. Li, H. Sang, Y. Liu, K. Gao, and Q. Pan, "Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106343.

[20] J. W. Kruisselbrink et al., "Enhancing search space diversity in multi-objective evolutionary drug molecule design using niching," in *Proc. 11th Annu. Conf. Genet. Evol. Comput.*, 2009, pp. 217–224.

[21] Y. Su, S. Li, C. Zheng, and X. Zhang, "A heuristic algorithm for identifying molecular signatures in cancer," *IEEE Trans. Nanobiosci.*, vol. 19, no. 1, pp. 132–141, Jan. 2020.

[22] G.-C. Luh and C.-Y. Lin, "Optimal design of truss-structures using particle swarm optimization," *Comput. Struct.*, vol. 89, nos. 23–24, pp. 2221–2232, 2011.

[23] J. Liang, C. Yue, G. Li, B. Qu, P. Suganthan, and K. Yu, "Problem definitions and evaluation criteria for the CEC 2021 on multimodal multiobjective path planning optimization," Zhengzhou Univ., Zhengzhou, China, Nanyang Technol. Univ., Singapore, Rep., Dec. 2020.

[24] O. M. Shir, M. Preuss, B. Naujoks, and M. T. Emmerich, "Enhancing decision space diversity in evolutionary multiobjective algorithms." in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2009, pp. 95–109.

[25] T. Ulrich, J. Bader, and L. Thiele, "Defining and optimizing indicator-based diversity measures in multiobjective search," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2010, pp. 707–717.

[26] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[27] R. Tanabe and H. Ishibuchi, "A decomposition-based evolutionary algorithm for multi-modal multi-objective optimization," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2018, pp. 249–261.

[28] R. Tanabe and H. Ishibuchi, "A niching indicator-based multi-modal many-objective optimizer," *Swarm Evol. Comput.*, vol. 49, pp. 134–146, Sep. 2019.

[29] Y. Peng and H. Ishibuchi, "A diversity-enhanced subset selection framework for multi-modal multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 886–900, Oct. 2022.

[30] Q. Lin, W. Lin, Z. Zhu, M. Gong, J. Li, and C. A. Coello Coello, "Multimodal multiobjective evolutionary optimization with dual clustering in decision and objective spaces," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 130–144, Feb. 2021.

[31] Y. Tian, R. Liu, X. Zhang, H. Ma, K. C. Tan, and Y. Jin, "A multipopulation evolutionary algorithm for solving large-scale multimodal multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 405–418, Jun. 2021.

[32] G. Li, W. Wang, W. Zhang, Z. Wang, H. Tu, and W. You, "Grid search based multi-population particle swarm optimization algorithm for multimodal multi-objective optimization," *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100843.

[33] K. Zhang, C. Shen, G. G. Yen, Z. Xu, and J. He, "Two-stage double niched evolution strategy for multimodal multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 754–768, Aug. 2021.

[34] J. Zhao et al., "Path planning based on multi-objective topological map," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2021, pp. 1719–1726.

[35] B. Jin, "Multi-objective A* algorithm for the multimodal multi-objective path planning optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)* 2021, pp. 1704–1711.

[36] L. Davis et al., "Applying adaptive algorithms to epistatic domains." in *Proc. IJCAI*, vol. 85, 1985, pp. 162–164.

[37] 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.

[38] I. Oliver, D. Smith, and J. R. C. Holland, "Study of permutation crossover operators on the traveling salesman problem," in *Proc. 2nd Int. Conf. Genet. Algorithms Appl.*, 1987, pp. 224–230.

[39] Y. Nagata and S. Kobayashi, "A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem," *INFORMS J. Comput.*, vol. 25, no. 2, pp. 346–363, 2013.

[40] A. Hussain, Y. S. Muhammad, M. N. Sajid, I. Hussain, A. M. Shoukry, and S. Gani, "Genetic algorithm for traveling salesman problem with modified cycle crossover operator," *Comput. Intell. Neurosci.*, vol. 2017, Oct. 2017, Art. no. 7430125.

[41] C. Segura, S. B. Rionda, A. H. Aguirre, and S. I. V. Peña, "A novel diversity-based evolutionary algorithm for the traveling salesman problem," in *Proc. Annu. Conf. Genet. Evol. Comput.*, 2015, pp. 489–496.

[42] Y. Nagata, "High-order entropy-based population diversity measures in the traveling salesman problem," *Evol. Comput.*, vol. 28, no. 4, pp. 595–619, Dec. 2020.

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

[44] D. Angus, "Niching for population-based ant colony optimization," in *Proc. 2nd IEEE Int. Conf. E-Sci. Grid Comput.*, 2006, p. 115.

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

[46] X. Chen, Y. Liu, X. Li, Z. Wang, S. Wang, and C. Gao, "A new evolutionary multiobjective model for traveling salesman problem," *IEEE Access*, vol. 7, pp. 66964–66979, 2019.

[47] D. H. Moraes, D. S. Sanches, J. da Silva Rocha, J. M. C. Garbelini, and M. F. Castoldi, "A novel multi-objective evolutionary algorithm based on subpopulations for the bi-objective traveling salesman problem," *Soft Comput.*, vol. 23, no. 15, pp. 6157–6168, 2019.

[48] X. Cai et al., "The collaborative local search based on dynamic-constrained decomposition with grids for combinatorial multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 5, pp. 2639–2650, May 2021.

[49] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA J. Comput.*, vol. 3, no. 4, pp. 376–384, 1991.

[50] M. Li, S. Yang, and X. Liu, "Bi-goal evolution for many-objective optimization problems," *Artif. Intell.*, vol. 228, pp. 45–65, Nov. 2015.

[51] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.
[52] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1, 2000, pp. 204–211.
[53] A. Zhou, Q. Zhang, and Y. Jin, "Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1167–1189, Oct. 2009.
[54] H. Ishibuchi, R. Imada, N. Masuyama, and Y. Nojima, "Comparison of hypervolume, IGD and IGD+ from the viewpoint of optimal distributions of solutions," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2019, pp. 332–345.
[55] H. Ishibuchi, L. M. Pang, and K. Shang, "Difficulties in fair performance comparison of multi-objective evolutionary algorithms [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 17, no. 1, pp. 86–101, Feb. 2022.

Yiping Liu (Member, IEEE) received the B.Eng. degree in electrical engineering and automation and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, Xuzhou, China in 2012 and 2017, respectively.

He has been an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, since 2020. His research interests include evolutionary computation, multiobjective optimization, and machine learning.

Liting Xu is currently pursuing the M.S. degree in computer technology with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China.

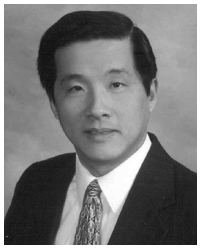
Her research interests include evolutionary multimodal multiobjective optimization and combinatorial optimization.

Yuyan Han received the M.S. degree from the School of Computer Science, Liaocheng University, Liaocheng, China, in 2012, and the Ph.D. degree in control theory and control engineering from the China University of Mining and Technology, Xuzhou, China, in 2016.

She is an Associate Professor with the School of Computer Science, Liaocheng University. Her current research interests include evolutionary computation, multiobjective optimization, and flow-shop scheduling.

Xiangxiang Zeng received the Ph.D. degree in system engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2011.

He is currently a Yuelu Distinguished Professor with the College of Information Science and Engineering, Hunan University, Changsha, China. His main research interests include computational intelligence, graph neural networks, and bioinformatics.



Gary G. Yen (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Notre Dame, Notre Dame, IN, USA, in 1992.

He is a Regents Professor with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. His research interests include intelligent control, computational intelligence, conditional health monitoring, signal processing, and their industrial/defense applications.

Dr. Yen is currently serving as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS. He is a Fellow of IET and IAPR.

Hisao Ishibuchi (Fellow, IEEE) received the B.S. and M.S. degrees from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree from Osaka Prefecture University, Sakai, Japan, in 1992.

He is a Chair Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His research interest is evolutionary multiobjective optimization.

Dr. Ishibuchi is currently an IEEE CIS AdCom Member from 2021 to 2023, an IEEE CIS Distinguished Lecturer from 2021 to 2023, and the General Chair of WCCI 2024 in Yokohama, Japan.