Manifold Interpolation for Large-Scale Multiobjective Optimization via Generative Adversarial Networks

Zhenzhong Wang, Member, IEEE, Haokai Hong, Graduate Student Member, IEEE, Kai Ye, Guang-En Zhang, Min Jiang, Senior Member, IEEE, and Kay Chen Tan, Fellow, IEEE

Abstract—Large-scale multiobjective optimization problems (LSMOPs) are characterized as optimization problems involving hundreds or even thousands of decision variables and multiple conflicting objectives. To solve LSMOPs, some algorithms designed a variety of strategies to track Pareto-optimal solutions (POSs) by assuming that the distribution of POSs follows a low-dimensional manifold. However, traditional genetic operators for solving LSMOPs have some deficiencies in dealing with the manifold, which often results in poor diversity, local optima, and inefficient searches. In this work, a generative adversarial network (GAN)-based manifold interpolation framework is proposed to learn the manifold and generate high-quality solutions on the manifold, thereby improving the optimization performance of evolutionary algorithms. We compare the proposed approach with several state-of-the-art algorithms on various large-scale multiobjective benchmark functions. The experimental results demonstrate that significant improvements have been achieved by the proposed framework in solving LSMOPs.

Index Terms—Evolutionary algorithm, generative adversarial networks (GANs), large-scale optimization, multiobjective optimization.

I. INTRODUCTION

MANY real-world optimization problems involve hundreds or even thousands of decision variables and multiple conflicting objectives, and such problems are called large-scale multiobjective optimization problems (LSMOPs) [1]–[4]. For example, in the design of telecommunication networks [5], thousands of variables, such as locations of network nodes, transmission capacity between nodes, and power allocations of nodes, are involved. These numerous decision variables determine the multiple objectives such as energy consumption, applicability, and stability of the network. It is challenging to find optimal solutions for this type of problems because the volume of the search space is exponentially related to the number of decision variables, and thus the curse of dimensionality [6] is exhibited. Therefore, a large-scale multiobjective optimization algorithm (LSMOA) should overcome this issue to search in the search space efficiently and effectively, and this ability is crucial to solving LSMOPs.

It is well-known that traditional genetic operators encounter low search efficiency when solving LSMOPs [7]–[9]. This is because the offspring reproduced by genetic operations are often located near their parents and cannot escape from the local optimum, which leads to the population to premature. As simulated binary crossover [10] exhibits excellent performance and has been widely used [7], [11], we take simulated binary crossover as an example here: the similarity between the offspring and their parents can be defined by a factor $\beta$, and the probability distribution of $\beta$ is shown in Fig. 1. When $\beta = 1$, the offspring is the same as the parent. When $\beta$ is further away from 1, the offspring are farther away from their parents. It can be seen the offspring are generated near the parents with a great probability.1 Furthermore, the above phenomenon is easy to happen in the context of LSMOPs, since the number of decision variables is large and the fitness landscape becomes more complicated, which contains more local optima needing to be escaped from.

Due to the complicated landscape of LSMOPs and the inefficiency of conventional genetic operators, research on embedding machine learning techniques into evolutionary algorithms for designing better LSMOAs is attracting increasing attention in the evolutionary computation and machine learning community. Gaussian process [12], covariance matrix [13], and generative adversarial network (GAN) [9] have been incorporated

1Although simulated binary crossover can adjust the distribution of $\beta$ to control the position of offspring via hyper-parameters, it is a difficult task to adjust the hyper-parameters adaptively.
with LSMOAs for learning the distributions of solutions and generating offspring. In addition, principal component analysis (PCA) [14], pattern mining [15], and unsupervised neural networks [16] have been used for reducing the dimension of LSMOPs. In the area of complex optimization problems, Liu et al. [17] designed a neural network to transfer optimization information and generate the population for dynamic optimization problems. Zhao and Guo [18] and Wang et al. [19] chose self-organizing mapping neural network and SVM as the surrogate, respectively, to select promising individuals during the evolutionary search. Wang and Jin [20] designed a random forest-assisted evolutionary algorithm for optimizing trauma systems. Also, some works incorporating machine learning models solved combinatorial optimization problems, e.g., Hopfield neural networks [21]–[23], reinforcement learning [24]–[26], and deep learning [27]–[29].

Inspired by the ability of machine learning models to learn high-dimensional features, our work uses a GAN to learn the characteristics of the decision space of LSMOPs and generates better offspring. To be specific, we use manifold interpolation via GAN to generate offspring candidates to escape from the local optima and approach the promising search area. Initially, GAN-LMEF uses non-dominated solutions to construct a series of approximate manifolds. Then, by adopting the proposed GAN-based large-scale multiobjective evolutionary framework (GAN-LMEF) is proposed. The difference between our proposed and GMOEA is that our work focuses on how to introduce uniformity and diversity to the population through interpolated offspring solutions locating on the manifold, such that the population can better escape from the local optimum so as to be appropriate for LSMOPs. In addition, our proposed selection strategy aims to select samples for better approximating the manifold.

We believe that the key factor for solving LSMOPs is to find an effective way to learn the manifold and generate offspring on this manifold, and GAN can be used as a powerful weapon to solve the issues. An intuitive idea is to learn the manifold and interpolate solutions along the manifold via GANs, as shown in Fig. 2. These interpolated offspring solutions fill gaps of the manifold and are uniformly distributed on the manifold, thereby leading the population to draw on the optimal area efficiently. Initially, GAN-LMEF uses non-dominated solutions to construct a series of approximate manifolds. Then, by adopting the proposed GAN-based interpolation method, various solutions are synthesized along manifolds. Next, the population is updated by a manifold selection mechanism. The above procedures are repeated until the termination conditions are met. The contributions of this work are summarized as follows.

1) The proposed framework uses a GAN as a powerful weapon to learn the characteristics of high-dimensional solutions and generate solutions along the manifold. These generative solutions are of high quality and can be helpful in the evolutionary process. The framework is not entirely dependent on the genetic operators to generate offspring but learns the distribution of high-dimensional solutions by modeling low-dimensional variables, which avoids the search of large-scale space directly. This provides a novel way to solve LSMOPs.

Fig. 1. Probability distribution of $\beta$ of simulated binary crossover [10].

Fig. 2. Interpolate solutions on the manifold.
2) Considering that the gaps on the manifold vary with the evolutionary process, three interpolation strategies are proposed to manipulate the low-dimensional latent variables, such that more diverse offspring can be generated.

3) In the selection process, we propose a manifold selection mechanism to predict whether a generative solution is excellent without evaluating objective functions, which can reduce evaluations and avoid many meaningless evaluations.

The rest of this article is organized as follows. Section II introduces the existing LSMOAs for LSMOPs, and GANs are briefly reviewed. Section III details the proposed algorithm for GAN-LMEF. In Section IV, the empirical results of GAN-LMEF and various state-of-the-art LSMOAs on LSMOPs are presented. Finally, conclusions are drawn in Section V.

II. PRELIMINARY STUDIES

A. Large-Scale Multiobjective Optimization

The mathematical form of LSMOPs is as follows:

\[
\min_{x \in \Omega} F(x) = (f_1(x), f_2(x), \ldots, f_m(x))
\]

where \(x = (x_1, x_2, \ldots, x_n)\) is the \(n\)-dimensional decision vector, and \(F = (f_1, f_2, \ldots, f_m)\) is the \(m\)-dimensional objective vector. It should be noted that the dimension of the decision vector \(n\) is greater than 100 [8], [36]. The goal of LSMOAs is to find the solutions for which all objectives are as small as possible. However, conflicts exist between multiple objectives, and one solution cannot satisfy the minimum of all objectives. Therefore, a trade-off method called Pareto dominance is introduced to compare these solutions. The set of all possible trade-off solutions is called the POS in the decision space and the Pareto-optimal front (POF) in the objective space.

Definition 1 (Decision Vector Domination): A decision vector \(x_1\) Pareto-dominates another vector \(x_2\) denoted by \(x_1 \succ x_2\), if and only if

\[
\begin{align*}
    f_i(x_1) &\leq f_i(x_2), & i = 1, \ldots, m, \\
    f_i(x_1) &< f_i(x_2), & i = 1, \ldots, m.
\end{align*}
\]

Definition 2 (Pareto-Optimal Set, POS): If a decision vector \(x^*\) satisfies

\[
\text{POS} = \{x^*| x \succ x^* \}
\]

then all \(x^*\) are called Pareto-optimal solutions, and the set of Pareto-optimal solutions is called the POS.

Definition 3 (Pareto-Optimal Front, POF): POF is the corresponding objective vector of the POS

\[
\text{POF} = \{y^*| y^* = F(x^*), x^* \in \text{POS} \}.
\]

B. Theory of the Manifold

The Karush–Kuhn–Tucker condition induces the POS of a continuous multiobjective problem to follow a continuous \((m - 1)\)-dimensional manifold in the decision space, and the theorem is given in detail in [30] and [31]. Suppose that the objectives \(f_i(x), i = 1, \ldots, m\) are continuous and differentiable at the POS \(x^*\), then \(\alpha = (\alpha_1, \ldots, \alpha_m)^T (||\alpha||_2 = 1)\) exists and satisfies

\[
\sum_{i=1}^{m} \alpha_i \nabla f_i(x^*) = 0.
\]

The points satisfying (4) are the Karush–Kuhn–Tucker points. Equation (4) has \(n + 1\) equality constraints and \(n + m\) variables \(x_1, \ldots, x_n, \alpha_1, \ldots, \alpha_m\). Thus, under certain smoothness conditions, the distribution of POS of a multiobjective problem is a continuous \((m - 1)\)-dimensional manifold. The test instances in the LSMOP benchmark [37] are continuous and differentiable and meet this basic theorem.

Using this manifold property as a guideline, some algorithms reproduce offspring via probabilistic models to approximate promising solutions’ area. Zhang et al. [38] modeled solutions as a \(k\)-piecewise low-dimensional manifold, and then sampled and selected solutions from the model to approximate the POS. Cheng et al. [12] learned the mapping from the objective space to the decision space by the Gaussian process, and then generated offspring via mapping by sampling the objective space. Compared with these manifold-learning-based methods, our proposed method has merits both in learning the manifold and generating offspring. First, neural networks exhibit strong power to handle high-dimensional features, which is beneficial for analyzing the decision space of LSMOPs [39]. Second, because GAN has the intrinsic ability to travel manifold, the proposed method can generate solutions with diversity by manipulating the low-dimensional latent variables and better escape from the local optimum, which is more efficient in solving LSMOPs.

C. Related Work

The existing LSMOAs can be roughly grouped into the following three categories [40], including decision variable grouping-based LSMOAs, decision space reduction-based LSMOAs, and novel search strategy-based LSMOAs.

Decision variable grouping-based LSMOAs divide the decision variables into several groups and then optimize each group of decision variables successively. Antonio and Coello [41] maintained several independent subpopulations. Each subpopulation is a subset of the equal-length decision variables obtained by variable grouping (e.g., random grouping, linear grouping, ordered grouping, or differential grouping). All the subpopulations work cooperatively to optimize the LSMOPs in a divide-and-conquer manner.

Zhang et al. [42] proposed a decision variable clustering-based large-scale evolutionary algorithm (LMEA). In LMEA, the decision variables are divided into two types using a clustering method. Then, for diversity-related variables and convergence-related variables, two different strategies are implemented to improve convergence and diversity.

Ma et al. [43] presented a multiobjective evolutionary algorithm based on decision variable analysis (MOEA/DVA). The LSMOP is decomposed into a number of simpler subproblems that are regarded as independent subpopulations to optimize. The disadvantage of decision variable analysis-based LSMOAs is that they may incorrectly identify the linkages between
decision variables to update solutions, which may lead to local optima [42].

However, decision variable grouping methods easily lead to excessive computational complexity due to the large-scale decision variables [44], and two related variables may be divided into different groups, which may lead to local optima [45].

Decision space reduction-based LSMOAs aim to convert large-scale problems into small-scale problems by dimensionality reduction techniques, and then conventional evolutionary algorithms solve the transformed problems. He et al. [46] introduced a generic framework called the large-scale multiobjective optimization framework (LSMOF). LSMOF reformulates the original LSMOP into a low-dimensional single-objective optimization problem with some direction vectors and weight variables, which aims at guiding the population toward the POS.

Zille et al. [47] proposed a weighted optimization framework (WOF) for solving LSMOPs. Decision variables are divided into many groups, and each group is assigned a weight variable. Then, in the same group, the weight variables are regarded as a subproblem of a subspace of the original decision space.

However, due to the information loss of the original search space caused by dimensionality reduction, the transformed problem is likely to lose the global optimal solutions.

Novel search strategy-based LSMOAs design reproduction operators and probability models for solving the LSMOPs directly. Tian et al. [8] used a competitive swarm optimizer to solve LSMOPs. The proposed LMOCSO suggests a two-stage strategy to update the positions of particles: a competitive mechanism is adopted to determine the particles to be updated, and the proposed updating strategy is used to update each particle.

GMOEA [9] used a GAN to generate offspring for solving multiobjective optimization problems. IM-MOEA [12] used the Gaussian process model to map the solutions from the objective space to the decision space and generated offspring. SparseEA [48] designed a new crossover operator and a new mutation operator for better searching the sparse optimal solutions by controlling the sparsity of the offspring.

A more comprehensive review of LSMOAs and LSMOPs can be referenced in [40].

D. Generative Adversarial Networks

GANs are generative models used to learn the characteristics of complicated real-world data. The basic structure of a GAN contains two neural nets: a discriminator and a generator. In addition to using generator neural nets to synthesize meaningful instances from prior distributions, a discriminator is trained to distinguish fake samples synthesized by the generator and real samples from the training dataset. The generator aims to learn the distribution of training instances and generate fake instances to deceive the discriminator. The discriminator consists of a classifier that can determine whether the input sample is a generative fake sample or a real sample. The training procedure culminates with a balance between the generator and the discriminator, i.e., the discriminator cannot make a better decision that determines whether a particular sample is fake or real. Compared with other generative models, GANs provide a novel and efficient framework for learning and generating data. Therefore, GANs have recently been successfully applied to many applications [9], [49]–[51].

To learn the distribution of the training data, the generator \( G \) takes the prior distribution of a random noise variable \( z \sim P_z \) as input to generate the data distribution \( P_G(z) \) approximating to the real data distribution \( P_{data}(x) \) and tries to make the difference between distributions of the generative data and true data as small as possible. In the existing GANs, Kullback–Leibler divergence [52], Jensen–Shannon divergence [53], and Wassertein distance [51] can be used to measure the difference between two probability distributions. The discriminator determines whether the input is generative fake instances or true instances, and the output probability of discriminator \( D_G(z) \) tends to 0 for fake instances, while it tends to 1 for true instances. Finally, the discriminator cannot distinguish whether the data belong to real samples or generative samples. The discriminator is updated by the stochastic gradient ascent method, and the generator is updated by the stochastic gradient descent method. The process of training is a game between the two neural networks, and the model eventually tends to the global optimum. The objective function of the game can be expressed as

\[
\min_G \max_D V(G, D) = \mathbb{E}_{x \sim P_{data}} \log(D(x)) + \mathbb{E}_{z \sim P_z} \log(1 - D(G(z))).
\]

From the perspective of the manifold, GANs have the ability to model low-dimensional manifolds. The real-world data follow a low-dimensional manifold [54], that is, the low-dimensional data can be considered as a compact representation of high-dimensional real data. In GAN, generator \( G \) models \( z \) to generate high-dimensional data (the dimension of \( z \) is generally smaller than that of the real data). Through generator \( G \), the mapping from low-dimensional data to high-dimensional data is learned, and \( z \) can be seen embedding on the manifold [55]. When this manifold is learned, the similarity Sim between two generated samples can be approximated as the Euclidean distance between their corresponding latent variables [32]

\[
\text{Sim}(G(z_1), G(z_2)) \approx ||z_1 - z_2||.
\]

Hence, interpolating \( z_1 \) and \( z_2 \) can produce uniform and smooth samples locating on the manifold. In such a way, we can use this manifold interpolation method to improve the uniformity and diversity of the population, thus helping the population escape from the local optima and better handling LSMOPs. This is why manifold interpolation strategy can be beneficial to solve LSMOPs.

III. PROPOSED FRAMEWORK

This work proposes a GAN-driven manifold interpolation method to address LSMOPs. The details of the proposed
algorithm, GAN-LMEF, are presented in this section. Briefly, GAN-LMEF consists of three subroutines: Procedure Computing Central Solutions, Procedure Manifold Interpolation, and Procedure Selection. In Fig. 3, we give a detailed illustration of the main loop in GAN-LMEF. The main purpose of Procedure Computing Central Solutions is to map non-dominated solutions into a manifold and find some representative non-dominated solutions on this manifold, called central solutions. In Procedure Manifold Interpolation, a GAN is used to interpolate offspring solutions between these central solutions. Procedure Selection selects promising solutions from the parental population and generative offspring to guide the future search.

The main scheme of the proposed GAN-LMEF is presented in Algorithm 1. The proposed manifold interpolation strategy should only be implemented in the early stage of the search to avoid the premature of population. Therefore, we allocate the first \( e \cdot e \) evaluations for the interpolation of solutions, where parameter \( e \) controls the number of iterations of interpolation and \( e \) is the maximum number of evaluations for solving the problem. When \( e \cdot e \) evaluations are used, the remaining \((1 - e) \cdot e\) evaluations are optimized by any population-based MOA. In every iteration of the main loop, the fast non-dominated sort [56] is used to distinguish the non-dominated solution set \( \mathcal{F} \). Next, the non-dominated solution set \( \mathcal{F} \) is inputted into Procedure Computing Central Solutions in which these non-dominated solutions are mapped into the \( m - 1 \)-dimensional manifold. These mapping solutions are clustered into several clusters, and a central solution \( C_i \) for each cluster is identified. Afterward, in Procedure Manifold Interpolation, a GAN is used to interpolate offspring \( Q \) between the central solutions \( C \). Finally, Procedure Selection selects promising solutions \( P \) from the parental population and interlopes offspring for the next round.

In Sections III-A–III-C, we will introduce the three main components: Procedure Computing Central Solutions, Procedure Manifold Interpolation, and Procedure Selection, respectively.

### A. Computing Central Solutions Procedure

Central solutions are the representatives on the manifold, and interpolating between these central solutions may yield better solutions. First, to obtain the manifold, the non-dominated solutions \( \mathcal{F} \) are mapped as the manifold data \( \phi(\mathcal{F}) \) by PCA. The mapped data \( \phi(\mathcal{F}) \) are clustered into \( k \) clusters by \( k \)-means. Each cluster is denoted as \( \phi(\mathcal{F})_i, i = 1, \ldots, k \), and in each cluster, a central solution is identified to represent the cluster. The central solution \( C_i \) for each cluster \( \phi(\mathcal{F})_i \) is defined as the solution with the minimum distance from \( \phi(C_i) \) to the centroid of the cluster calculated as

\[
C_i = \arg\min_x ||\phi(x) - \phi(\mathcal{F}_i)||. \tag{7}
\]

The centroid \( \phi(\mathcal{F}_i) \) of the cluster \( \phi(\mathcal{F})_i \) is

\[
\phi(\mathcal{F}_i) = \frac{\sum_{j=1}^{\#(\mathcal{F}_i)} \phi(\mathcal{F})_{i,j}}{\#(\mathcal{F}_i)} \tag{8}
\]

where \( \phi(\mathcal{F})_{i,j} \) represents the \( j \)th data in the cluster \( \phi(\mathcal{F})_i \).

The central solutions identified and the piecewise manifolds are shown in Fig. 3(b). The details of computing the central solutions are given in Procedure Computing Central Solutions.

### Algorithm 1 GAN-LMEF

**Input:** \( N \) (the population size), \( e \) (the proportion of interpolated solutions), MOA (a multiobjective optimization algorithm), \( F() \) (the LSMOP), \( k \) (the number of clusters), \( e \) (the maximum evaluations)

**Output:** \( P, O, S \) (the POS)

1. Set parameter \( e \) for controlling the number of iterations of interpolation;
2. Initialize population \( P \) randomly;
3. while Used evaluations \( \leq e \cdot e \) do
4. \( P' = \) Mating-Selection\((P)\);
5. \( P = P \cup \) Variation\((P')\);
6. \( \mathcal{F} = \) Fast-Non-Dominated-Sort\((P, F())\);
7. \( C = \) Computing-Central-Solutions\((\mathcal{F}, k)\);
8. \( Q = \) GAN-Manifold-Interpolation\((C, P, k)\);
9. \( O = \) Selection\((C, Q, \sigma)\);
10. end
11. \( P, O, S = \) MOA\((P, F()), (1 - e) \cdot e\);
12. return \( P, O, S \)

### Procedure (Computing Central Solutions)

**Input:** \( \mathcal{F} \) (the non-dominated set), \( k \) (the number of clusters)

**Output:** \( C \) (the central solutions)

1. \( C = \varnothing \);
2. Use PCA with \( \mathcal{F} \) to get the manifold \( \phi(\mathcal{F}) \);
3. Cluster \( \phi(\mathcal{F}) \) into \( k \) clusters using \( k \)-means;
4. for \( i = 1, \ldots, k \) do
5. Calculate central solution \( C_i \) for each cluster \( \phi(\mathcal{F})_i \) according to (7);
6. \( C = C \cup C_i \);
7. end
8. return \( C \)
clusters; 2) interpolating in the cluster; and 3) perturbation

three interpolation strategies are used: 1) interpolating between

After that, the well-trained generator:

and diversity and are beneficial to the evolutionary process.

These solutions can improve uniformity

gaps of this manifold. These solutions can improve uniformity

Fig. 4. Flowchart of the manifold interpolation model. Step 1. Train the GAN with non-dominated solutions \( F \) and noise variables \( z \). Step 2. Save the well-trained generator: \( G \). Step 3. Apply interpolation strategies to \( z \) and input these interpolating noise variables into the generator: \( G \). Step 4. Generate interpolating solutions \( Q \).

**Procedure (GAN Manifold Interpolation)**

| Input: C (the central solutions), \( F \) (the population), \( F(\cdot) \) (the LSMOP), \( k \) (the number of clusters) |
| Output: \( Q \) (the interpolation solutions) |
| Q = ∅; |
| \( F = \text{Fast-Non-Dominated-Sort}(P, F(\cdot)); \) |
| 3. Sample noise variables \( \{z\} \) from noise prior \( U(-1, 1); \) |
| 4. Train a GAN with \( F \) and \( \{z\} \), save its generator: \( G \); |
| 5. for \( i = 1, \ldots, k \) do |
| 6. Extract the corresponding variable \( z_{C_i} \) from \( \{z\} \) where \( G(z_{C_i}) = \hat{C}_i \); |
| 7. end |
| 8. //Interpolating between clusters; |
| 9. for \( i = 1, \ldots, k \) do |
| 10. for \( j = 1, \ldots, k \) and \( j \neq i \) do |
| 11. Interpolate the noise variable \( z_{ij} \) according to (9); |
| 12. \( Q = Q \cup G(z_{ij}); \) |
| 13. end |
| 14. end |
| 15. //Interpolating in the cluster; |
| 16. for \( i = 1, \ldots, k \) do |
| 17. for \( x_a, x_b \) in the \( k \)th cluster do |
| 18. Interpolate the noise variable \( z_{ab} \) according to (10); |
| 19. \( Q = Q \cup G(z_{ab}); \) |
| 20. end |
| 21. end |
| 22. //Perturbation interpolation; |
| 23. Add disturbance to noise variables \( \{z\} \) to generate variable \( \{z'\}; \) |
| 24. \( Q = Q \cup G(\hat{z}'); \) |
| 25. return \( Q \); |

**B. Manifold Interpolation Procedure**

The manifold interpolation will reproduce a number of offspring solutions that are distributed on the manifold and fill gaps of this manifold. These solutions can improve uniformity and diversity and are beneficial to the evolutionary process.

Before interpolating, a standard GAN is trained with non-dominated solutions \( F \) and latent variables \( \{z\} \sim U(-1, 1) \) to learn the manifold characteristics of solutions. After that, the well-trained generator: \( G \) and latent variables \( \{z\} \) are saved in the archive for later interpolation. In our work, three interpolation strategies are used: 1) interpolating between clusters; 2) interpolating in the cluster; and 3) perturbation interpolation.

1) **Interpolating Between Clusters**: This method is used to fill the gap between two clusters. Noise variables \( z_{C_i} \) are extracted from the archive, where \( z_{C_i} \) is the noise variable that generates the generative fake central solution \( \hat{C}_i \) (\( G(z_{C_i}) = \hat{C}_i \)). For every two noise variables \( z_{C_i} \) and \( z_{C_j} \) of fake central solutions, interpolating noise variables between them can be described as [57]

\[
z_{ij} = (1 - \alpha)z_{C_i} + \alpha z_{C_j}, \quad i, j = 1, \ldots, k, i \neq j \tag{9}
\]

where \( \alpha = l/n, l = 0, 1, \ldots, n \), and \( n \) represents the dimension of the decision variable. Because \( z_{ij} \) are gradual between \( z_{C_i} \) and \( z_{C_j} \), the generative solutions \( G(z_{ij}) \) generated by \( z_{ij} \) will also be gradual between the two clusters. For \( k \) clusters, the GAN will interpolate solutions between different central solutions, so it will interpolate \( k \times (k - 1)/2 \) times.

2) **Interpolating in the Cluster**: This method is used to fill gaps between individuals from the same cluster. The interpolating noise variables are described as

\[
z_{ab} = \frac{z_a + z_b}{2} \tag{10}
\]

where \( z_a \) and \( z_b \) are the noise variables that generate fake solutions \( x_a \) and \( x_b \), respectively, and \( x_a \) and \( x_b \) are from the same cluster. Similarly, these noise variables \( z_{ab} \) are inputted into the generator: \( G \). This interpolation method imitates the semantic operation for noise variables [58].

3) **Perturbation Interpolation**: This method is used to apply disturbance on a noise variable to explore the neighbor space of a corresponding solution. Perturbation interpolation adds Gaussian noise to a random dimension of a noise variable. These noise variables are also inputted into the generator: \( G \).

The details of the manifold interpolation are given in Procedure Manifold Interpolation, and an illustration of the manifold interpolation is shown in Fig. 4.

**C. Procedure of Selection**

In the selection procedure, high-quality solutions are identified from interpolated solutions’ set \( Q \) and used to update the population.
 due to having the two minimum MDs.

There is a huge number of generative solutions in \( Q \), and it is a waste of evaluations to calculate their objectives and determine whether a solution is high quality. To settle this issue, a manifold selection mechanism is proposed to select solutions that have the minimum distance to the manifold. The illustration of the manifold selection mechanism is shown in Fig. 5.

First, PCA is used with \( C \) and \( Q \) to obtain the mapping manifold data \( \phi(C) \) and \( \phi(Q) \), respectively. Next, the manifold distance (MD) from each mapping solution \( x \in \phi(Q) \) to the two nearest mapping central solutions \( \phi(C_i) \) and \( \phi(C_j) \) is computed

\[
MD(x) = ||x - \phi(C_i)||_2 + ||x - \phi(C_j)||_2.
\]

Then, top-\( \sigma \cdot |P| \) solutions that have the minimum MD are selected from \( Q \) and denoted as \( Q' \). The solutions in \( Q' \) are promising and can improve convergence or diversity because they are close to the existing piecewise manifolds or lie in the gap between piecewise manifolds. Finally, only solutions in \( Q' \) are evaluated and solutions \( x_i \in P \) are replaced with \( x_j \in Q' \), where \( x_j > x_i \). The new population \( P \) is regarded as a parental population for the next round.

Fig. 6 illustrates the selection process. The details of the selection are given in Procedure Selection.

D. Complexity Analysis

In this section, we present the time complexity analysis for each component of GAN-LMEF and give an overall time complexity.

For the Procedure Computing Central Solutions, the \( k \)-means algorithm requires \( O(k \cdot n \cdot N) \), and calculating the central solutions for each cluster requires \( O(N) \), where \( k \) is the number of clusters and \( n \) and \( N \) are the dimension of the decision variables and the size of the population, respectively.

For the Procedure Manifold Interpolating, the generator of the GAN contains a three-layer fully connected neural network [59], and the discriminator contains a three-layer fully connected neural network in which a sigmoid unit is used as the output layer. It takes \( O(\sum_{l=1}^{d} n_l \cdot N_{l-1} \cdot N^2_{l-2}) \) to execute the GAN once [60], where \( d \) is the depth of the neural networks, and \( n_l \) is the number of neurons in the \( l \)th layer. Interpolation of solutions between central solutions of clusters requires execution of the GAN \( k \ast (k-1)/2 \) times, and the total time complexity of executing the GAN in our work is \( O(k^2 \cdot (\sum_{l=1}^{d} n_l \cdot n_{l-1} \cdot N_{l-2}^2)) \). The time complexity of the Procedure GAN Manifold Interpolating is \( O(k \cdot n \cdot N) + O(N) + O(k^2 \cdot (\sum_{l=1}^{d} n_l \cdot n_{l-1} \cdot N_{l-2}^2)) = O(k^2 \cdot (\sum_{l=1}^{d} n_l \cdot n_{l-1} \cdot N_{l-2}^2)) \).

The time complexity of the Procedure Selection is \( O(N \cdot k) \).

Because \( k \) is a user-defined constant, the overall time complexity of GAN-LMEF is simplified as \( O(\sum_{l=1}^{d} n_l \cdot n_{l-1} \cdot N_{l-2}^2) \).

IV. EXPERIMENTS

A. Algorithms in Comparison and Test Problems

The proposed GAN-LMEF is compared against several popular algorithms including WOF [47], LSMOF [46], GMOEA [9], LMOCSO [8], LMEA [42], and RM-MEDA [38]. The parameters of the WOF, GMOEA, LMOCSO, RM-MEDA, and LMOCSO algorithms are set to defaults according to [61]. For a fair comparison, the training parameters and the structure of the GAN in GMOEA are the same as those in ours. The basic solvers of the proposed GAN-LMEF, WOF [47], and LSMOF [46] are unified as NSGA2 [62], MOEADDE [63], and CMOPSO [64], respectively, and we rename them according to the basic solver, e.g., GAN-NSGA2, WOF-NSGA2, and LSMOF-NSGA2.

The widely used large-scale multiobjective test problem LSMOP1-LSMOP9 [37] is used in our experiments. For all competitive comparisons, the number of objectives is set to 2 and the number of decision variables is set from 500 to 2000. For the ablation study, the number of objectives varies from 2 to 6, and the number of decision variables is fixed to 500.
B. Performance Indicators

In this study, the following metrics are used to evaluate the performance of the compared algorithms.
1) **Inverted Generational Distance (IGD)** [65]: The IGD is a metric for quantifying the convergence of the solutions obtained by a multiobjective optimization algorithm. When the IGD value is small, the convergence of the solution is improved.
2) **Schott’s Spacing (SP)** [66]: It measures the uniformity of the solutions found by the algorithm. The smaller the SP value, the more uniform the distribution of the solution obtained by the algorithm.

C. Parameter Settings

1) **Population Size**: The population size $N$ is set to 100 for all test instances.
2) **Termination Condition**: The number of maximum evaluations $e$ is set to 100,000 for all compared LSMOAs.
3) **The Structure of the GAN Is as Follows**: The number of nodes of the three-layer fully connected neural network in the generator is set to $m - 1$, $[n \neq 0.5]$, and $n$, respectively. The number of nodes of the fully connected neural network in the discriminator is set to $n$, $[n \neq 0.5]$, and 1, respectively.
4) **Parameters of GAN-LMEF**: The number of clusters $k$ is set to 3. The proportion of the introduced interpolated solutions $\sigma$ is set to 0.4. $\epsilon$ is set to 0.1. The learning rate of the GAN is set to 0.001, the epoch is set to 200, and the GAN is trained by full batch learning.

D. Performance on LSMOP Problems

The statistical results of the average IGD and SP values over ten runs can be found in Tables I and S-I (supplementary material), respectively. In the tables, (+), (=), and (−) indicate that GAN-NSGA2 is statistically significantly better, indifferent, or significantly worse than the compared algorithms, respectively, using the Wilcoxon test [67] (0.05 significance level).

As can be observed from Table I, it is obvious that the proposed GAN-NSGA2 exhibits better performance than the five compared algorithms for convergence. GAN-NSGA2 achieved 17 of the 27 best results, LSMOF-NSGA2 achieved 6 of the best results, GMOEA achieved 2 of the best results, and WOF-NSGA2 achieved 2 of the best results for IGD values. Specifically, GAN-NSGA2 performs better than NSGA2 and MOEADDE on LSMOP1, LSMOP2, LSMOP4, LSMOP5, and LSMOP9 under all decision variable settings but is slightly worse than LSMOF-NSGA2 on the LSMOP3 and LSMOP6 problems. GMOEA can obtain a set of well-converged solutions for the LSMOP7 problems with 1000 and 2000 decision variables. WOF-NSGA2 achieves better convergence over POE on LSMOP7 and LSMOP8 with 500 decision variables. It should be noted that the authors of LMOCSO and LMEA set larger evaluations for their comparisons. Therefore, for a fair comparison, we conducted additional experiments to compare LMOCSO and LMEA with our proposed algorithms with different basic solvers, where the number of evaluations is $n^* = 15000$ and $n$ is the number of decision variables. As can be seen from Table S-II, the proposed GAN-LMEF with basic solvers NSGA2 and MOEADDE (GAN-NSGA2 and GAN-MOEADDE) is still very competitive. The statistical results of SP values are presented in Table S-I. GAN-NSGA2 performs the best on 16 out of 27 test instances, followed by RM-MEDA with six best results, LMOCSO with four best results, and WOF-NSGA2 with one best result.

The above experimental results suggest that GAN-NSGA2 can obtain a set of solutions with good convergence and diversity for most test problems. However, GAN-NSGA2 is worse for IGD values on LSMOP3 and LSMOP6. This may be attributed to the fitness landscapes of LSMOP3 being the adoption of Rosenbrock’s function and Rastrigin’s function. LSMOP6 mixes Rosenbrock’s function and Ackley’s function [37]. The mix of separable and nonseparable functions leads to a complicated fitness landscape and increases the difficulty of learning characteristics, thus degenerating the performance of the interpolated solutions.

E. Ablation Study

In this section, we perform ablation experiments to demonstrate the effectiveness of applying the proposed framework to NSGA2, MOEADDE, and CMOPSO. In addition, we embed basic solvers into two popular frameworks for solving LSMOP, WOF, and LSMOF. Note that NSGA2, MOEADDE, and CMOPSO are not suitable for solving LSMOPs. The statistics of IGD values can be found in Table II.

As seen from Table II, GAN-NSGA2 (GAN-MOEADDE and GAN-CMOPSO) achieves significantly better IGD values than WOF-NSGA2 (WOF-MOEADDE and WOF-CMOPSO) and LSMOF-NSGA2 (LSMOF-MOEADDE and LSMOF-CMOPSO). Specifically, GAN-NSGA2 performs significantly better than WOF-NSGA2 in 17 out of 27 test functions and LSMOF-NSGA2 in 18 out of 27 test functions. GAN-NSGA2 has better IGD values than WOF-MOEADDE in 23 out of 27 test functions and LSMOF-MOEADDE in 19 out of 27 instances. GAN-CMOPSO obtains better IGD values than WOF-CMOPSO in 15 out of 27 instances and LSMOF-CMOPSO in 15 out of 27 functions. These pairwise comparisons have demonstrated the ability of our proposed GAN-LMEF framework in improving the performance of the existing MOAs on LSMOPs.

To verify the effectiveness of interpolation for latent noise variables $\epsilon$, we develop Ip-LMEF that directly uses the piecewise linear interpolation method to interpolate solutions between the decision variables of central solutions. Also, WOF$_{\sigma}$-LMEF is designed, which generates solutions via GAN but without interpolating noise variables. The experimental results are shown in Tables III, S-III, and S-IV.

As seen from Table III, GAN-NSGA2 achieves six of the nine best IGD values, while Ip-NSGA2 achieves three of the best results. For the optimizer MOEADDE, GAN-MOEADDE achieves six of the best results. For the basic solver CMOPSO, GAN-CMOPSO performs significantly better than Ip-CMOPSO, WOF$_{\sigma}$-CMOPSO, and CMOPSO in six out of nine instances. We also draw convergence profiles of
TABLE I
MEAN AND VARIANCE VALUES OF IGD METRIC OBTAINED BY THE COMPARED ALGORITHMS OVER LSMOP PROBLEMS

| Problem | Dec. | GAN-NSGA2 | WOP-NSGA2 | LAMF-NSGA2 | GAMA | LAMO-CE | LARBA | EMO-UCDA |
|---------|------|-----------|-----------|------------|------|---------|-------|----------|
| LSO-PP | 500  | 2.45±3.54(7.30) | 1.81±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.75(0.46) | 1.80±1.72(0.46) |
|         | 1000 | 2.26±3.04(7.30) | 1.81±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.75(0.46) | 1.80±1.72(0.46) |
| LSO-PP | 500  | 2.45±3.54(7.30) | 1.81±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.75(0.46) | 1.80±1.72(0.46) |
|         | 1000 | 2.26±3.04(7.30) | 1.81±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.75(0.46) | 1.80±1.72(0.46) |
| LSO-PP | 500  | 2.45±3.54(7.30) | 1.81±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.75(0.46) | 1.80±1.72(0.46) |
|         | 1000 | 2.26±3.04(7.30) | 1.81±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.75(0.46) | 1.80±1.72(0.46) |

TABLE II
MEAN AND VARIANCE VALUES OF IGD METRIC OBTAINED BY THE COMPARED ALGORITHMS OVER LSMOP PROBLEMS

| Problem | Dec. | GAN-NSGA2 | WOP-NSGA2 | LAMF-NSGA2 | GAMA | LAMO-CE | LARBA | EMO-UCDA |
|---------|------|-----------|-----------|------------|------|---------|-------|----------|
| LSO-PP | 500  | 3.46±3.54(7.30) | 2.81±1.79(0.46) | 4.16±3.04(4.4) | 3.64±1.10(4.9) | 3.06±0.94(0.46) | 3.15±1.96(0.46) | 3.18±1.72(0.46) |
|         | 1000 | 5.27±3.04(7.30) | 6.28±1.79(0.46) | 7.46±3.04(4.4) | 6.36±1.10(4.9) | 5.06±0.94(0.46) | 5.15±1.96(0.46) | 5.18±1.72(0.46) |
| LSO-PP | 500  | 3.46±3.54(7.30) | 2.81±1.79(0.46) | 4.16±3.04(4.4) | 3.64±1.10(4.9) | 3.06±0.94(0.46) | 3.15±1.96(0.46) | 3.18±1.72(0.46) |
|         | 1000 | 5.27±3.04(7.30) | 6.28±1.79(0.46) | 7.46±3.04(4.4) | 6.36±1.10(4.9) | 5.06±0.94(0.46) | 5.15±1.96(0.46) | 5.18±1.72(0.46) |

TABLE III
MEAN AND VARIANCE VALUES OF IGD METRIC OBTAINED BY THE COMPARED ALGORITHMS OVER LSMOP PROBLEMS WHEN THE NUMBER OF DECISION VARIABLES IS 500

| Problem | Dec. | GAN-NSGA2 | WOP-NSGA2 | LAMF-NSGA2 | GAMA | LAMO-CE | LARBA | EMO-UCDA |
|---------|------|-----------|-----------|------------|------|---------|-------|----------|
| LSO-PP | 500  | 1.85§±3.54(7.30) | 1.32±1.79(0.46) | 1.73±3.04(4.4) | 1.46±1.10(4.9) | 1.86±0.94(0.46) | 1.65±1.96(0.46) | 1.59±1.72(0.46) |
|         | 1000 | 2.08±3.04(7.30) | 2.00±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.72(0.46) |
| LSO-PP | 500  | 1.85§±3.54(7.30) | 1.32±1.79(0.46) | 1.73±3.04(4.4) | 1.46±1.10(4.9) | 1.86±0.94(0.46) | 1.65±1.96(0.46) | 1.59±1.72(0.46) |
|         | 1000 | 2.08±3.04(7.30) | 2.00±1.79(0.46) | 3.16±3.04(4.4) | 1.58±1.10(4.9) | 2.06±0.94(0.46) | 1.80±1.96(0.46) | 1.59±1.72(0.46) |

IGD values obtained by GAN-NSGA2 and NSGA2 in Fig. 7. The convergence rate of GAN-NSGA2 is faster than that of NSGA2. As seen in the experimental results, incorporation of GAN into NSGA2 can greatly improve the quality and convergence speed of solutions in solving LSMOPs. These results confirm the effectiveness of our proposed interpolation method. The GAN learns the mapping relationship from the latent low-dimensional manifold space to the high-dimensional solutions, and interpolation of the latent variables z can generate high-quality solutions lying on the manifold.

F. Performance of Different Interpolation Strategies
To investigate the quality of solutions interpolated via different interpolation strategies, we record the number of different interpolated non-dominated solutions selected by the Procedure Selection during evolutionary optimization.

As shown in Fig. 8, interpolating between central solutions can produce much better individuals than the perturbation interpolation method and interpolating in the cluster method. This indicates that the gaps between clusters are always huge in solving LSMOPs, and the population lacks diversity and convergence during the entire evolutionary optimization.
Fig. 7. Convergence profiles of IGD values obtained by GAN-NSGA2 and NSGA2 on LSMOP problems with 500 decision variables.

Fig. 8. Number of selected non-dominated solutions which are interpolated by different strategies on LSMOP problems with 500 decision variables.

Therefore, interpolating between central solutions can produce more promising solutions.

G. Sensitivity Study

In GAN-LMEF, the parameters $k$ and $\epsilon$ decide the number of interpolated solutions and affect the performance. In this sensitivity study, the influence of $k$ and $\epsilon$ on the convergence of GAN-NSGA2 is investigated. Table IV shows IGD values obtained by different choices of $k$ and $\epsilon$ values for GAN-NSGA2 on the LSMOP benchmark. When $k$ increases from 2 to 3, IGD values improve in 18 out of 27 instances. When $\epsilon$ varies from 0.1 to 0.5, half of the instances in terms of IGD values become small, while half of the instances increase. Based on the experimental results, a larger $k$ value can generate more piecewise manifolds, thus interpolating many more solutions in the gaps. While the choice of parameter $\epsilon$ depends on the specific problem, it needs to be set carefully.

In addition, the influence of the proportion of interpolated solutions $\sigma$ and the first evaluations for interpolating $\cdot \cdot \epsilon$ are investigated, and the plots are shown in Fig. 9. We can observe that interpolating at the early evolution stage achieves better IGD values in most cases. This is because the proposed manifold interpolation model has a powerful ability of exploration. Besides, the parameter $\sigma$ is not significantly sensitive to the experimental results.

Mode collapse is a well-recognized and non-trivial problem in the realm of GAN, which means the generated samples are identical or very similar. To illustrate the quality of generated solutions, the dynamic behavior of the proposed...
Fig. 9. Influence of the proportion of interpolated solutions $\sigma$ and the first evaluations for interpolating $\epsilon$.

### TABLE IV

| Problem | $\epsilon$       | $\sigma$       | $\text{IGD}$ | $\text{var}$ |
|---------|------------------|----------------|--------------|--------------|
| LSMP1   | 0.1              | 0.4            | 0.06         | 0.67         |
| LSMP2   | 0.2              | 0.6            | 0.74         | 0.86         |
| LSMP3   | 0.3              | 0.8            | 0.74         | 0.86         |
| LSMP4   | 0.4              | 1.0            | 0.74         | 0.86         |
| LSMP5   | 0.5              | 1.2            | 0.74         | 0.86         |

Because POS can be represented in an $m-1$-dimensional manifold, the number of objectives $m$ can determine the complexity of the manifold, thus affecting the performance of interpolation. Therefore, we conduct the experiment concerning $m$, and the experimental results concerning the IGD metric are shown in Table V. GAN-NSGA2 achieves better IGD values in 12 out of 27 instances. However, we can immediately find that GAN-NSGA2 has more difficulty obtaining well-converged solutions on more objective functions because the manifold becomes more complicated and more difficult to learn.

### H. Discussion on Runtime

It is worth noting that the model-based evolutionary algorithms need extra computational resources to train the model. Therefore, it is necessary to discuss the runtime of the proposed GAN-LMEF. Fig. 12 presents the relationship between the number of decision variables and the wall-clock time of executing Lines 4–9 in Algorithm 1 once. The wall-clock time increases linearly with the number of decision variables. When the parameter $\epsilon$ is set to 0.1, $\sigma$ is 0.4, the population size $N$ is 100, and the total evaluations $e$ is set to 100,000, and there will be $(e \times \epsilon / (1 + \sigma) \times N) \approx 70$ times of process of executing Lines 4-9 in Algorithm 1. According to Fig. 12, we can see that the good performance of GAN-LMEF comes with a small price. We suggest to carefully tune these parameters according to the specific problem to meet the trade-off between the computation time and solution quality.

### I. Why GAN-LMEF Works?

During the past few decades, there are many crossover operators proposed. Among these operators, the simulated binary crossover exhibits excellent performance and has been widely used. However, as the fitness landscape becomes more complicated and contains more local optima, the offspring generated by simulated binary crossover is more difficult...
to escape from the local optimum. Therefore, we design a model-based approach to partially replace the genetic operators to reproduce offspring. We note that POS is distributed on the low-dimensional manifold, and the GAN can be used to learn the manifold. When the GAN is well-trained, GAN can automatically learn the manifold of data, and the similarity between two generated samples can be approximated as the Euclidean distance between their corresponding latent variables as shown in (6). Therefore, interpolating $z_1$ and $z_2$ can generate uniform samples locating on the manifold. In such a

---

**TABLE V**

| Problem   | GAN-SSGA | WAF-SSGA | LSMOF-SSGA | CGA | LibGA | LibMOEA | LibMARTA | KMI-MDA |
|-----------|----------|----------|------------|-----|-------|---------|----------|---------|
| LSMOP1    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP2    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP3    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP4    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP5    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP6    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP7    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP8    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |
| LSMOP9    | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) | 1.00(1.000) |

---

Fig. 10. SP values of the original population, generated population, and population being selected by Procedure Selection on LSMOP problems with 2000 decision variables.

Fig. 11. Number of non-dominated solutions being trained when the dimension of problems is 2000.

---

**Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.**
way, individuals with uniformity and diversity are introduced, thus helping the population escape from the local optima and better handling LSMOPs.

V. CONCLUSION

This article has presented a GAN-based evolutionary search framework for solving LSMOPs, called GAN-LMEF. The aim of GAN-LMEF is to interpolate solutions on the manifold via a GAN to maintain the effective manifold. Based on the generative population, higher quality offspring are reproduced, thereby improving the evolutionary performance.

In the proposed GAN-LMEF, a GAN is used to learn various characteristics from the non-dominated solutions and generate a number of solutions by three different interpolation strategies. Then, a manifold selection mechanism selects promising solutions from the generative solutions for the next round. We integrated the proposed framework with NSGA2, MOEA/D, and CMOPO and evaluated their performances. The experimental results have demonstrated that the proposed GAN-LMEF is more competitive than several state-of-the-art LSMOAs.

Several possible directions may be taken for future work. For large-scale problems with complicated search spaces, such as constraints and disconnected Pareto-optimal regions, the proposed framework still has much room for improvement. In addition, a rich body of transfer techniques such as evolutionary transfer optimization and advanced machine learning can inspire further innovations in solving real-world applications with a large number of decision variables.

REFERENCES

[1] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, “A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications,” IEEE Trans. Evol. Comput., vol. 17, no. 3, pp. 321–344, Jun. 2013.
[2] Y. Zhou, G. G. Yen, and Z. Yi, “Evolutionary shallow deep neural networks at block levels,” IEEE Trans. Neural Netw. Learn. Syst., early access, Feb. 26, 2021, doi: 10.1109/TNNLS.2021.3059529.
[3] Z. Stanko, T. Nishikawa, and S. R. Paulinski, “Large-scale multi-objective optimization for the management of seawater intrusion, Santa Barbara, CA,” in Proc. AGU Fall Meeting Abstr., 2015, pp. H31G–1507.
[4] Z. Zhao, M. Jiang, S. Guo, Z. Wang, F. Chao, and K. C. Tan, “Improving deep learning based optical character recognition via neural architecture search,” in Proc. IEEE Congr. Evol. Comput. (CEC), Jul. 2020, pp. 1–7.
[5] J. Cheney, “The application of optimisation methods to the design of large scale telecommunication networks,” in Proc. IEE Colloq. Large-Scale Hierarchical Syst., Mar. 1988, pp. 2–1–2.
[6] H. Wang, L. Jiao, R. Shang, S. He, and F. Liu, “A memetic optimization strategy based on dimension reduction in decision space,” Evol. Comput., vol. 23, no. 1, pp. 69–100, Mar. 2015.
[7] S. Yang, Y. Tian, C. He, X. Zhang, K. C. Tan, and Y. Jin, “A gradient-guided evolutionary approach to training deep neural networks,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 4, 2021, doi: 10.1109/TNNLS.2021.3061630.
[8] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, “Efficient large-scale multi-objective optimization based on a competitive swarm optimizer,” IEEE Trans. Cybern., vol. 50, no. 8, pp. 3696–3718, Aug. 2020.
[9] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, “Evolutionary multiobjective optimization driven by generative adversarial networks (GANs),” IEEE Trans. Cybern., vol. 51, no. 6, pp. 3129–3142, Jun. 2021.
[10] K. Deb and R. B. Agrawal, “Simulated binary crossover for continuous search space,” Complex Syst., vol. 9, no. 2, pp. 115–148, 1995.
[11] N. Hitomi and D. Selma, “A classification and comparison of credit assignment strategies in multiobjective adaptive operator selection,” IEEE Trans. Evol. Comput., vol. 21, no. 2, pp. 294–314, Apr. 2017.
[12] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, “A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling,” IEEE Trans. Evol. Comput., vol. 19, no. 6, pp. 838–856, Dec. 2015.
[13] H. Chen, R. Cheng, J. Wen, H. Li, and J. Weng, “Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations,” Inf. Sci., vol. 509, pp. 457–469, Jan. 2020.
[14] R. Liu, R. Ren, J. Liu, and J. Liu, “A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems,” Appl. Soft Comput., vol. 89, Apr. 2020, Art. no. 106120.
[15] Y. Tian, C. Lu, X. Zhang, F. Cheng, and Y. Jin, “A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems,” IEEE Trans. Cybern., early access, Dec. 30, 2021, doi: 10.1109/TCYB.2020.3041325.
[16] Y. Tian, C. Lu, X. Zhang, K. C. Tan, and Y. Jin, “Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks,” IEEE Trans. Cybern., vol. 51, no. 6, pp. 3115–3128, Jun. 2021.
[17] X.-F. Liu et al., “Neural network-based information transfer for dynamic optimization,” IEEE Trans. Neural Netw. Learn. Syst., vol. 31, no. 5, pp. 1557–1570, May 2020.
[18] C. Zhao and D. Guo, “Particle swarm optimization algorithm with self-organizing mapping for Nash equilibrium strategy in application of multiobjective optimization,” IEEE Trans. Neural Netw. Learn. Syst., early access, Nov. 4, 2020, doi: 10.1109/TNNLS.2020.3027293.
[19] B. Wang, B. Xue, and M. Zhang, “Surrogate-assisted particle swarm optimization for evolving variable-length transferable blocks for image classification,” IEEE Trans. Neural Netw. Learn. Syst., early access, Feb. 8, 2021, doi: 10.1109/TNNLS.2021.3054400.
[20] H. Wang and Y. Jin, “A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems,” IEEE Trans. Cybern., vol. 50, no. 2, pp. 536–549, Feb. 2020.
[21] H. Tang, K. C. Tan, and Z. Yi, “A columnar competitive model for solving combinatorial optimization problems,” IEEE Trans. Neural Netw. Learn. Syst., vol. 15, no. 6, pp. 1568–1574, Nov. 2004.
[22] K. C. Tan, H. Tang, and S. S. Ge, “On parameter settings of Hopfield networks applied to traveling salesman problems,” IEEE Trans. Circuits Syst. I: Reg. Papers, vol. 52, no. 5, pp. 994–1002, May 2005.
[23] F. Cai et al., “Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks,” Nature Electron., vol. 3, pp. 1–10, Jul. 2020.
[24] J. J. Q. Yu, W. Yu, and J. Gu, “Online vehicle routing with neural combinatorial optimization and deep reinforcement learning,” IEEE Trans. Intell. Transp. Syst., vol. 20, no. 10, pp. 3806–3817, Oct. 2019.
[25] T. Barrett, W. Clements, J. Foerster, and A. Lvovsky, “Exploratory combinatorial optimization with reinforcement learning,” in Proc. AAAI Conf. Artif. Intel., Apr. 2020, vol. 34, no. 4, pp. 3243–3250.
[26] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, “Reinforcement learning for combinatorial optimization: A survey,” Comput. Oper. Res., vol. 134, Oct. 2021, Art. no. 105400.
[27] T. Guo, C. Han, S. Tang, and M. Ding, “Solving combinatorial problems with machine learning methods,” in Nonlinear Combinatorial Optimization. Cham, Switzerland: Springer, 2019, pp. 207–229.
[28] B. Wilder, E. Ewing, B. Dilkina, and M. Tambe, “End to end learning and optimization on graphs,” in Advances in Neural Information Processing Systems 32, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 4674–4685. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/8bd39eae38511daad6152e84545e504d-Abstract.html and https://dlp参数引用/GAN-LMEF19.jpg
H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, “A framework for multi-objective optimization,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 8, Aug. 2023.

Y. Tian, X. Zhang, C. Wang, and Y. Jin, “An evolutionary algorithm for multi-objective optimization,” IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 8, Aug. 2023.

H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and composition,” in Proc. 32nd AAAI Conf. Artif. Intell. (AAAI), 30th Innov. Appl. Artif. Intell. (IAAI), 8th AAAI Symp. Educ. Adv. Artif. Intell. (EAAI), S. A. McIraith and K. Q. Weinberger, Eds. New Orleans, LA, USA: AAAI Press, Feb. 2018, pp. 34–41. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17286 and https://dblp.org/rec/conf/aaai/DongHY18.bib.

F. Chao et al., “Generative adversarial nets in robotic Chinese calligraphy,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2018, pp. 1104–1110.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein GANs,” in Advances in Neural Information Processing Systems 30. Red Hook, NY, USA: Curran Associates, 2017, pp. 5767–5777.

M. N. Do and M. Vetterli, “Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance,” IEEE Trans. Image Process., vol. 11, no. 2, pp. 146–158, Feb. 2002.

J. Lin, “Divergence measures based on the Shannon entropy,” IEEE Trans. Inf. Theory, vol. 37, no. 1, pp. 145–151, Jan. 1991.

A. Khan and P. Maji, “Multi-manifold optimization for multi-view subspace clustering,” IEEE Trans. Neural Netw. Learn. Syst., early access, Feb. 19, 2021, doi:10.1109/TNNLS.2021.3054789.

M. Edraki and G.-J. Qi, “Generalized loss-sensitive adversarial learning with manifold margins,” in Computer Vision—ECCV 2018 (Lecture Notes in Computer Science), vol. 11209, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Springer, Sep. 2018, pp. 90–104. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3143031.3143042.

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.

C. Wang, C. Xu, X. Yao, and D. Tao, “Evolutionary generative adversarial networks,” IEEE Trans. Evol. Comput., vol. 23, no. 6, pp. 921–934, Dec. 2019.

A. Radford and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” Comput. Sci., vol. 13, no. 2, pp. 284–302, 2015.

T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long-term short-term memory, fully connected deep neural networks,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Apr. 2015, pp. 4580–4584.

K. He and J. Sun, “Convolutional neural networks at constrained time cost,” in Proc. CVPR, Jun. 2015, pp. 5353–5360.

Y. Tian, R. Cheng, X. Zhang, and Y. Jin, “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum],” IEEE Comput. Intell. Mag., vol. 12, no. 4, pp. 73–87, Nov. 2017.

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II,” in Parallel Problem Solving From Nature PPSN VI, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, Eds. Berlin, Germany: Springer, 2000, pp. 849–858.

H. Li and Q. Zhang, “Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II,” IEEE Trans. Evol. Comput., vol. 13, no. 2, pp. 284–302, Apr. 2009.

X. Zhang, X. Zheng, R. Cheng, J. Qiu, and Y. Jin, “A competitive mechanism based multi-objective particle swarm optimizer with fast convergence,” Inf. Sci., vol. 427, pp. 63–76, Feb. 2018.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. D. Fonseca, “Performance assessment of multiobjective optimizers: An analysis and review,” IEEE Trans. Evol. Comput., vol. 7, no. 2, pp. 117–132, Apr. 2003.

G. Imsayilov and H. R. Tопчоглу, “Dynamic multi-objective workflow scheduling for cloud computing based on evolutionary algorithms,” in Proc. IEEE/ACM Int. Conf. Utility Cloud Comput. Companion (UCC Companion), Dec. 2018, pp. 103–108.

J. Derrac, S. García, D. Molina, and F. Herrera, “A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms,” Swarm Evol. Comput., vol. 1, no. 1, pp. 3–18, Mar. 2011.
Haokai Hong (Graduate Student Member, IEEE) received the bachelor’s degree in computer science from Xiamen University, Xiamen, China, in 2020, where he is currently pursuing the master’s degree with the School of Informatics.

His research interests include computational intelligence and machine learning.

Kai Ye is currently pursuing the bachelor’s degree with the Department of Artificial Intelligence, Xiamen University, Xiamen, China.

He is currently an intern at the Department of Artificial Intelligence, Xiamen University. His current research interests include neural architecture search and transfer learning.

Guang-En Zhang received the bachelor’s degree from Beijing University of Chemical Technology, Beijing, China, in 2018. He is currently pursuing the master’s degree with the Department of Artificial Intelligence, Xiamen University, Xiamen, China.

He is currently a member with the Key Laboratory of Digital Protection and Intelligent Processing of Intangible Cultural Heritage of Fujian and Taiwan, Ministry of Culture and Tourism, Xiamen. His current research interest includes neural architecture search.

Min Jiang (Senior Member, IEEE) received the bachelor’s and Ph.D. degrees in computer science from Wuhan University, Wuhan, China, in 2001 and 2007, respectively.

He was a Post-Doctoral Researcher with the Department of Mathematics, Xiamen University, Xiamen, China. He is currently a Professor with the Department of Artificial Intelligence, Xiamen University. His main research interests include machine learning, computational intelligence, and robotics. He has a special interest in dynamic multiobjective optimization, transfer learning, software development, and the basic theories of robotics.

Dr. Jiang received the Outstanding Reviewer Award from IEEE TRANSACTIONS ON CYBERNETICS in 2016. He is the Chair of IEEE Computational Intelligence Society (CIS) Xiamen Chapter. He has been serving as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS.

Kay Chen Tan (Fellow, IEEE) received the B.Eng. (Hons.) and Ph.D. degrees from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently the Chair Professor (computational intelligence) of the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He has published more than 300 refereed articles and seven books.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He has served as the Editor-in-Chief for IEEE Computational Intelligence Magazine from 2010 to 2013 and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020. He serves as an editorial board member for more than ten journals. He is also an IEEE Distinguished Lecturer Program (DLP) Speaker and the Chief Coeditor of Machine Learning: Foundations, Methodologies, and Applications (Springer Book Series).