IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 8 May 2024, accepted 19 May 2024, date of publication 23 May 2024, date of current version 3 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3404407

RESEARCH ARTICLE

A Hybrid Composite Differential Evolution and Multiobjective Particle Swarm Optimization Evolutionary Algorithm and Its Application

JIN SHANG[1] **AND GUIYING LI**[2]

[1]Control Technology Institute, Wuxi Institute of Technology, Wuxi 214121, China
[2]School of Mechanical and Electrical Engineering, Heilongjiang University, Harbin 150080, China

Corresponding author: Guiying Li (liguiying@hlju.edu.cn)

This work was supported by Jiangsu Province University Science and Technology Innovation Team Project.

ABSTRACT The current multi-objective particle swarm algorithms excel in convergence speed for solving complex problems but often suffer from a loss of population diversity. Conversely, composite differential evolution algorithms maintain superior solution distribution but lag in convergence efficiency. This research introduces an improved hybrid algorithm, CoDE-MOPSO, which integrates multi-objective particle swarm optimization with composite differential evolution based on clustering technology. The clustering algorithm is used for all individual clusters to analyze the distribution constructs of populations, which determines whether the new solutions come from global or local populations at a mating restriction probability. The mating restriction probability is updated at each generation. To adapt the balance between the population solution diversities and the convergence speed of the algorithm, at each generation, the control probability is adjusted by a developed adaptive strategy according to the reproduction utility of the two mechanisms of generating new solutions over the last certain generations. This research introduces the CoDE-MOPSO algorithm, designed to transcend existing multi-objective optimization methods' limitations by optimally balancing exploration and exploitation. Our approach significantly advances evolutionary multi-objective optimization, demonstrating superior performance through lower Inverse Generational Distance and higher Hypervolume metrics, indicating enhanced efficiency in solving complex MOPs across various fields. In practical scenarios like gear reducer optimization, CoDE-MOPSO showcases remarkable effectiveness, highlighting its value in engineering applications and setting a foundation for sophisticated optimization strategies that combine speed with solution quality.

INDEX TERMS Clustering algorithm, composite differential evolution, gear reducer, multiobjective optimization, particle swarm algorithm.

I. INTRODUCTION

In practical engineering, multi-objective optimization problems (MOPs) often refer to those with several independent variables, with inequality constraints or equality constraints, and with nonlinear objective functions. The general constrained MOP [1] can be stated as follows.

$$\min \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^{\mathrm{T}}$$
$$s.t. = \begin{cases} \mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}} \in \Omega \\ g_i(\mathbf{x}) \le 0 \qquad i = 1, 2, 3, \cdots, p \\ h_j(\mathbf{x}) = 0 \qquad j = 1, 2, 3, \cdots, q \end{cases} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$ is the n-dimensional decision vector, $\Omega = \prod_{i=1}^{n}[a_i, b_i] \subseteq R^n$ represents the decision space where the variable xi is restricted to $[a_i, b_i]$, and $\mathbf{F}(\mathbf{x})$ is the objective vector with m objective function $f_i(\mathbf{x}), i = 1, \cdots m$ in the

The associate editor coordinating the review of this manuscript and approving it for publication was Sotirios Goudos.

objective space. In (1), p inequality constraints $g_i(\mathbf{x})$ and q equality constraints $h_j(\mathbf{x})$ are also considered.

In general, unlike a single objective optimization problem which has one or several isolated optimal solutions, MOP, as the objectives are conflicting, always could not obtain the solutions to optimize them simultaneously but a set of tradeoffs, called Pareto set (PS). Then all the PSs are mapped to the target space to get the target vectors, which are called Pareto front (PF). Solving MOPs is to acquire the approximation front close to and covering all the PFs by the evolutionary algorithms. Traditional methods are not good at dealing with MOPs, while Multiobjective Evolutionary Algorithms (MOEAs), free of restrictions of problem regulation characteristics, do well in them and have gained remarkable achievements [2]. Along this way, more and more MOEAs have been proposed, and among them, multi-objective particle swarm optimization (MOPSO) and composite differential evolution (CoDE) exhibit advantages in approximating PF. The former is a simple algorithm with fewer control parameters and fast convergence speed. It depends on the information sharing mechanism to make each particle adjust its velocity and position according to the experience of the global optimum and past optimum when a new solution is achieved [3], [4], [5]. However, this mechanism may extinguish the diversity, which causes the approximation front not to distribute along the whole PF as evenly as possible. The latter, the CoDE algorithm, maintains diversity by utilizing three complementary difference operators to generate new solutions, however, at the cost of low convergence speed [6]. Considering the advantages and disadvantages of the aforementioned two algorithms, some improved algorithms are proposed. Therefore, a hybrid composite differential evolution and multi-objective particle swarm optimization evolutionary algorithm (CoDE-MOPSO) is proposed in this paper and then applied to the optimum design of mechanical structure for a two-level straight gear reducer. This study presents the CoDE-MOPSO algorithm, designed to address the critical challenges in multi-objective optimization by synergistically integrating Composite Differential Evolution and Multi-Objective Particle Swarm Optimization. Our approach significantly enhances the capacity to navigate the intricate landscapes of complex optimization problems, providing a robust solution that balances the dual needs of diversity and convergence, with wide-reaching implications for both theoretical advancements and practical applications in various fields. To make the topic of this paper clear, the main contributions are listed as follows:

- Coordinate Particle Swarm Optimization Algorithm with Composite Differential Evolution Algorithm to get a new solution. To balance between the population diversity and convergence speed, CoDE-MOPSO introduces the control probability β to determine the way to generate a new solution, Particle Swarm Algorithm, or Composite Differential Evolution Algorithm and thus improves the solving ability.
- The control probability β is updated adaptively in each iteration according to the utilization of different reproduction mechanisms so as to adapt to different MOPs and different evolutionary stages.
- The novelty of our method lies in the adaptive integration of K-means [7] clustering with CoDE and MOPSO algorithms to dynamically segment the population based on the evolving solution landscape. This integration allows for a more nuanced balance between exploration and exploitation by identifying and focusing on promising regions of the search space, while maintaining diversity across the population.

Section II states a brief description of the related works, including MOPSO-based MOEAs and CoDE-based MOESs. Section III describes the framework and other parts of CoDE-MOPSO in detail. Section IV gives the experimental results of CoDE-MOPSO with four state-of-the-art popular MOEAs on the test instances. Section V discusses instances with complicated PF shapes and parameter sensitivity further. Section VI explains the application of CoDE-MOPSO algorithm in optimum design of gear reducer. Finally, the conclusion of the paper is drawn in Section VII.

II. RELATED WORK

A. COMPOSITE DIFFERENTIAL EVOLUTION ALGORITHM

The differential evolution, proposed by Storn and Price, is a population-based evolutionary algorithm to solve single-objective optimization problems [8]. DE receives great popularity because of the simplicity of being implemented and coded, the advantage of optimizing no separable objective functions, and few number of control parameters needed [9]. There are 5 widely used DE mutation operators such as rand/1, best/1, rand/2, best/2 and current-to-rand/1 [10]. Considering different reproduction operators hold different search abilities on different regions in the search space, at different stages of the search, or on different optimization problems. Zhang et al. proposed a novel framework of coordinating with multiple operators to reach a new solution, named Composite Differential Evolution Algorithm [11]. In this algorithm, two candidate pools are prepared to store mutation strategies and control parameter settings. The CoDE algorithm chooses mutation strategies and control parameter settings from candidate pools to generate new individuals. The efficiency and robustness have been proved by simulation in the existing literature. Then CoDE was studied more deeply and used in many fields [12].In order to improve the effectiveness and robustness, Zhang et al. proposed an algorithm with the ensemble of two CoDEs biased toward exploration and exploitation respectively [13]. Yang and Liu proposed a self-adaptive clustering-based differential evolution with new composite trial vector generation strategies [14].

This paper utilizes the diversity maintenance mechanism of CoDE algorithm to make up the defect that particle swarm algorithm is apt to suffer local optimum. Therefore, CoDE

uses 3 mutation strategies with control parameter pool whose size is three to enhance performance of individuals. For diversity, three mutation operators, rand/1/bin, rand/2/bin, and current-to-rand/1, are selected as per (2), (3), and (4). The ''bin'' in ''rand/1/bin'' and ''rand/2/bin'' indicates that a binomial crossover is used. During this crossover, for each gene (or parameter) in the solution vector, a random number is compared with the crossover rate (CR). If the random number is less than CR, the gene from the mutated vector is carried over to the offspring; otherwise, the gene from the target vector is retained. This approach combines information from different solutions, maintaining diversity in the population and helping the algorithm escape local optima.

$$u_{i,j,t} = \begin{cases} x_{r1,j,t} + F(x_{r2,j,t} - x_{r3,j,t}) \\ \quad rand(j) \leq CR \ or \ j = randn(i) \\ x_{i,j,t} \quad otherwise \end{cases} \tag{2}$$

$$u_{i,j,t} = \begin{cases} x_{r1,j,t} + F(x_{r2,j,t} - x_{r3,j,t}) + F(x_{r4,j,t} - x_{r5,j,t}) \\ \quad rand(j) \leq CR \ or \ j = randn(i) \\ x_{i,j,t} \quad otherwiseor \end{cases} \tag{3}$$

$$u_{i,j,t} = \begin{cases} x_{i,j,t} + F(x_{r1,j,t} - x_{i,j,t}) + F(x_{r2,j,t} - x_{r3,j,t}) \\ \quad rand(j) \leq CRorj = randn(i) \\ x_{i,j,t} \quad otherwise \end{cases} \tag{4}$$

where randn(i) generates random integer from 1 to n. In mutation, F amplifies the difference vectors. x_{r1}, x_{r2}, and x_{r3} are randomly chosen individuals from the current population and are required to be different from the current solution. Rand/1 has superior exploration capability and large perturbations, but it has a slow convergence speed. Compared with rand/1, rand/2 is more robust and explorative with additional two difference vectors when scaling factor F is kept unchanged. Rand/2 suffers from slower convergence. Operator current-to-rand/1 makes the perturbation between the current individual (based vector) and two perturbation vectors, helping the solution of population maintain a good distribution. Current-to-rand/1 is rotation-invariant and suitable for rotated problems.

The important parameters in CoDE include scale factor (F),crossover rate and population size (N) [4]. There are 3 control parameters are listed as follows:

- $[F = 1.0, CR = 0.1]$ is used to solve separate problems.
- $[F = 1.0, CR = 0.9]$ is used to maintain the diversity of the population.
- $[F = 0.8, CR = 0.2]$ is used to enhance the exploration of the algorithm.

B. MULTIOBJECTIVE OPTIMIZATION ALGORITHM

Multiobjective optimization algorithms are algorithms used to solve optimization problems involving multiple objective functions [15]. In real-world decision-making scenarios, we often need to consider multiple goals or criteria simultaneously, and these goals may have conflicting or complementary relationships. It is generally impossible to achieve the optimum for all these objectives simultaneously, therefore the purpose of multiobjective optimization is to find a set of optimal solutions that provide the best trade-offs among different objectives [16].

Li et al. [17] presents a hybrid MOO algorithm combining enhanced NSGA-II with MOPSO, utilizing logistic mapping for better initial population distribution and a dynamic selection mechanism for crossover and mutation operators to balance search capabilities. It shows improved convergence and diversity on standard tests and the TEAM 22 benchmark compared to standalone methods. The MOIMPA [18] combines quantum theory with the marine predators algorithm to improve search efficiency in multi-objective optimization. It uses a modified Schrödinger equation for particle positioning and incorporates a Pareto dominance mechanism for solution selection. Abdullah et al. [19] introduces MOFDO, extending the single-objective Fitness Dependent Optimizer for multi-objective scenarios. It incorporates situational, normative, and other knowledge types, uses an archive for Pareto solutions, and employs polynomial mutation. Tested on ZDT and CEC 2019 benchmarks, MOFDO outperforms or equals existing methods like MOPSO and NSGA-III, proving its effectiveness in diverse problems. PSOMOFS [20] introduces a novel approach to feature selection by incorporating fuzzy cost considerations, utilizing particle swarm optimization. This method excels in balancing the trade-offs between minimizing feature costs and maximizing classification accuracy, showcasing significant advancements over conventional selection techniques.

Multi-objective particle swarm optimization (MOPSO) was introduced by Coello et al. [4]. MOPSO is a population based meta-heuristic algorithm that is illuminated by the social behavior of birds clustering to find food [3]. MOPSO with different strategies has been widely used to solve MOPs in many fields [21], [22], [23].

In recent years, MOPSO has been studied more deeply and more improvements are proposed. Zheng and Liu integrated the mutation strategies into MOPSO, which enhances the ability of MOPSO to jump out of the local optimal solution in the search process [24]. In order to improve the convergence of MOPSO, Zhao et al. proposed the decomposition strategy after the basic PSO to update the particle (MOPSO/AD) [25]. A new algorithm called I-MOPSO was introduced by A. Britto et al. It utilized archive and leader's selection methods to introduce more convergence to the Pareto front and address diversity of the obtained solutions [26]. In [27] Britto and Pozo enhanced I-MOPSO algorithm, and the new algorithm is called REF-I-MOPSO. In this algorithm, the I-MOPSO algorithm is evaluated in different scenarios and increases the search convergence.

MOPSO algorithms, proposed to deal with multiple objectives as well as multiple constraints, aim to search for the best approximation of PS. MOPSO algorithm has to find the global best particle $gbest_t^i$ and the past best particle $pbest_t^i$ of

the individual x^i, and then update the information of particle by the velocity and position. We perform environmental selection based on Pareto and hyper-volume. The new population combined with an external archive in environmental selection. The external archive is updated by dominating relationships.

In each generation, the velocity and position of the particle x^i denoted as $V^i_{t,j} = \left(v^i_{t,1}, v^i_{t,2}, \cdots, v^i_{t,j}\right)$ and $X^i_{t,j} = \left(x^i_{t,1}, x^i_{t,2}, \cdots, x^i_{t,j}\right)$, respectively, are updated as follows.

$$v^i_{t+1,j} = w \times v^i_{t,j} + c_1 r_1 (pbest^i_{t,j} - x^i_{t,j}) + c_2 r_2 (gbest^i_{t,j} - x^i_{t,j}) \tag{5}$$

$$x^i_{t+1,j} = x^i_{t,j} + v^i_{t,j} \tag{6}$$

where w is inertial weight, $c1$ is cognition coefficient, $c2$ is social coefficient, r_1 and r_2 are random constant between 0 and 1, which are employed to prevent the algorithm being jumped into local convergence, is current evolutionary generation number and $j = 1, \ldots, n$ represents dimension.

III. PROPOSED METHOD

A. ALGORITHM FRAMEWORK

The main ideas of CoDE-MOPSO is to partition the population by the k-means algorithm, and then build several clusters. The global cluster (GC) is established based on the clusters. In order to balance exploration and exploitation, the mating parents are selected from the global cluster or local population with a mating restriction probability. Then to trade off diversity and convergence, the control probability β will adapt to decide how to generate a new solution, by MOPSO or by CoDE. Moreover, parameter β can update itself according to the utility of producing new individuals by different reproductive mechanisms. The pseudo code of the CoDE-MOPSO algorithm is shown in Algorithm 1.

Algorithm 1 CoDE-MOPSO

1 Initialize the population $P = \{x^1, x^2, \cdots, x^N\}$, initialize the algorithm parameters: α_1 and β_1.
2 Set an archive population $A = P$.
3 **for** t =1, $\cdots$,T **do**
4 Partition $P = EnvironmentSel(A \cup P)$ into K clusters by K-means algorithm, $\{LC^1, L, LC^k\} = K - means(P, K)$.
5 Establish global classes GC.
6 Set the mating pool Q^i for x^i as $Q^i = \begin{cases} W^i \backslash \{x^i\} & if \ \ rand < \alpha \\ GC & otherwise \end{cases}$.
7 Generate a new solution $y^i = SolGen(Q^i, x^i, \alpha, \beta)$.
8 Update the archive population, $A = A \cup \{y^i\}$.
9 $P = EnvironmentSel(A \cup P)$, update the population. Record the numbers $NR^{gsp(clu)}$ of newly generated solutions come from global or from local population at a given rate.
10 Update β_{t+1} according to the raw fitness, $\beta_{t+1} = Update1\left(RF^{code(mopso)}, GN^{code(mopso)}\right)$.
11 Update α_{t+1} according to the number of newly generated individuals, $\alpha_{t+1} = Update2\left(NR^{gsp(clu)}, t\right)$.
12 **end for**
13 Return final population P.

In algorithm 1, K represents the number of clusters. N is the population size. Population P with the size N is initialized firstly. Mating restriction probability α and control parameter β are also initialized in line 1.Then; we set an archive population $A = P$(line 2). In each generation, population is partitioned into clusters by k-means (line 4). We established GC which individuals are selected from each cluster (line 5). Next, the mating pool for x^i consists of neighbor population with a probability α and the other is composed of the global cluster with a probability $1 - \alpha$ (line 6). MOPSO or CoDE serve as the method to generate new solution with a possibility β and a possibility $1 - \beta$, and then new solution y^i is achieved and stored in the external archive A (line 7-8). The population P is updated in environmental selection and the numbers $NR^{gsp(clu)}$ of newly generated solutions come from global or from local population at a given rate (line 9). At last, mating restriction probability α and the control possibility β is updated (line 10-11). After T iterations of evolutionary optimization, the final population P is obtained. This final population is the product of the algorithm's comprehensive application of MOPSO and CoDE strategies, along with population segmentation and adaptive updating of control parameters through k-means clustering technology (line 13).

B. K-MEANS ALGORITHM

K-means algorithm, known as one of the typical clustering technologies, the main idea is to classify a given data set by clustering population iteratively so that every point is assigned to the nearest cluster centers based on Euclidean distance. This clustering is based on minimizing the Euclidean distance between the individuals and the centroids of the clusters, effectively grouping similar solutions together. The k-means process iterates by first randomly selecting cluster centers, assigning each individual to the closest cluster based on the shortest Euclidean distance, and then updating the cluster centers based on the current members. Through this iterative process, the population is organized into distinct clusters that are used in the mating selection process. The algorithm dynamically chooses mating parents from either the entire population or specific clusters, controlled by a mating restriction probability. This clustering approach allows the CoDE-MOPSO algorithm to maintain a balance between exploration and exploitation, enhancing its performance on complex multi-objective optimization problems by leveraging localized search strategies within the clustered population. The basic procedure of K-means algorithm is shown in Algorithm 2.

K-means algorithm first randomly selects the cluster centers from P and set K empty clusters (line 1). Next, it calculates the Euclidean distance between the other individuals

Algorithm 2 $K-means(P, K)$

1 Initialize cluster centers $\{c^1, \cdots, c^K\}$, set K empty clusters $\{LC^1, \cdots, LC^k\}$.
2 **while** failure to meet end condition **do**
3 **for** $i = 1$ **to** N
4 ***if*** $k = \arg\min_{j=1,\cdots,K} distance\left(\mathbf{x}^i, c^j\right)$
5 x^i to cluster LC^k, $x^i \in P$, $i = 1, 2, \ldots, N$.
6 **end if**
7 **end for**
8 **for** $k = 1$ **to** K
9 Update cluster centers by $c^k = \left|\frac{1}{LC^k}\right| \sum_{x \in LC^k} x$.
10 **end for**
11 **end while**

to the centers and classifies them to the clusters with the shortest distance (line 3-5). Then, the cluster centers are updated (line 6-8). By averaging the positions (in the solution or feature space) of all individuals assigned to each cluster, thereby determining the new geometric center (line 9). The algorithm continues to classify and update until the clustering requirement is satisfied.

The innovative application of K-means in our CoDE-MOPSO framework specifically targets the clustering of the solution population to enhance the algorithm's exploration and exploitation capabilities. The application of K-means within our algorithm involves the following custom steps:

Initialization: At the start of the clustering process, we select initial cluster centers within the solution space based on a diversity measure to ensure a wide coverage across the population.

Assignment: Solutions (or individuals in the population) are then assigned to the nearest cluster not just based on their Euclidean distance in the objective space, but also considering their distribution in the decision space to maintain diversity within clusters.

Update: After all solutions are assigned, cluster centers are updated to reflect the current state of the population. This step may also include an adaptive mechanism to adjust the number of clusters based on the population's diversity and the optimization stage.

Integration with Evolutionary Processes: Clusters are then utilized to direct the selection process, where solutions from less crowded clusters are given preference to maintain diversity. Moreover, cluster information guides the crossover and mutation processes by encouraging intra-cluster exploration and inter-cluster exploitation.

Adaptive Clustering: The number of clusters and the clustering criterion are adaptively adjusted based on the evolutionary stage to balance exploration and exploitation dynamically.

C. NEW SOLUTION GENERATION

In CoDE-MOPSO, we use β to control new solutions generated by MOPSO or CoDE. A mating restriction α was set to control the source of parents. New solutions are generated through the Solution Generation (SolGen( )) function. The SolGen( ) function facilitates a balanced approach to optimization, allowing the algorithm to navigate the trade-offs between exploring new regions of the solution space and exploiting known promising areas. By dynamically adjusting its strategy based on the algorithm's current needs and the characteristics of the solution space. The detail is shown in Algorithm 3. For all the targets in population, if $rand < \beta$, the new solutions are generated by MOPSO algorithm (line 1-4), and are repaired with its boundaries to ensure the flexibility (line 5). a_j and $b_j (j = 1, 2, \ldots, n)$ denote the lower and upper boundaries of the j-th variable. The function $rand()$ means a random number between [0, 1]; if $rand > \beta$, the new solutions are created by CoDE (line 7-11), CoDE operator consists of three ordinary DE operators, named, ''rand/1/bin'' ''rand/2/bin'' and ''current-to-rand/1''. Two parents are selected according α (line 7). CoDE control parameters are selected to perform DE operator and evaluate the gene of y^i (line 8-10). We repair the gene of y^i (line 11). Finally, trial solution is generated (line 13).

D. ENVIRONMENTAL SELECTION

After generating new solution, $EnvironmentSel(A \cup P)$ will select the best ones from $A \cup P$ to enter next evolution procedure. We execute environmental selection method based on hypervolume and Pareto dominance [28]. How to conduct $EnvironmentSel(A \cup P)$ in CoDE-MOPSO is depicted in Algorithm 4. Firstly, We set the reference points for calculating hyper volume (line 1). Then add the trial solution y^i to archive population A to form P' (line 2). After calculating the number of dominating points dp^i for each individual in P' (line 3-4), the one with the largest dp or the one with the smallest HV_i is deleted according the calculating results (line5-11). Finally, P is updated and as the next generation population (line 12).

E. UPDATE OF CONTROL AND MATING RESTRICTION PROBABILITY

In the CoDE-MOPSO algorithm, there are two ways to generate a new solution, one is MOPSO and the other is crossover mutation operator of CoDE. The former may accelerate the convergence but lose the diversity of the solutions. The latter could maintain the diversity but suffer from slow convergence. To make full use of the advantage of both methods, control possibility β has to be designed elaborately, because it directly decides the proportion of these two methods to generate a new solution. In the real optimization procedure, a special MOP often needs special value of β and even for the same MOP, different stages in the evolutionary process always need different values of β, so it's of high importance to set a proper value for β. Therefore, it's so interesting that β in the proposed CoDE-MOPSO can update itself adaptively in the evolution procedure. The procedure of updating control possibility is shown in Algorithm 5. Here, we set

Algorithm 3 Hybrid Solution Update Mechanism

1 if $rand() < \beta$
2 Find out $pbest_t^i$ and $gbest_t^i$ of particle x^i;
Choose $gbest_t^i$ from $Q^i = \begin{cases} W^i \backslash \{x^i\} & if \ \ rand < \alpha \\ GC & otherwise \end{cases}$.
3 update the velocity of particles: $v_j^i = w \times v_j^i + c_1 r_1 (pbest_j^i - x_j^i) + c_2 r_2 (gbest_j^i - x_j^i), j = 1, \ldots, n$.
4 update the position of particles: $x_j^i = x_j^i + v_j^i, j = 1, \ldots, n$.
5 Repair the trial gene of current solution $y_j^i = \begin{cases} a_j & x_j^i < a_j \\ b_j & x_j^i > b_j \quad j = 1, 2, \ldots, n \\ x_j^i & \text{otherwise} \end{cases}$.
6 else
7 Select two parents p^1 and p^2 from the mating pool randomly $(p^1, p^2) = \begin{cases} x | x \in W^i & if \ \ rand < \alpha \\ x | x \in GC & otherwise \end{cases}$.
8 Utilize the following three trial vector generating strategies, "rand/1", "rand/2" and "current-to-rand", and choose a control parameter from candidate pool, which combined with it to generate three solutions are y^{i1}, y^{i2}, y^{i3}.
9 $y^i = (y^{i1}, y^{i2}, y^{i3})$

$$y_j^{i1} = \begin{cases} x_j^i + F(p_j^1 - p_j^2) \\ \quad rand \le CRorj = randn(i) \\ x_j^i \quad otherwise \end{cases}$$

$$y_j^{i2} = \begin{cases} x_j^i + F(p_j^1 - p_j^2) + F(p_j^3 - p_j^4) \\ \quad rand \le CRorj = randn(i) \\ x_j^i \quad otherwise \end{cases}$$

$$y_j^{i3} = \begin{cases} x_j^i + F(p_j^1 - x_j^i) + F(p_j^2 - p_j^3) \\ \quad rand \le CRorj = randn(i) \\ x_j^i \quad otherwise \end{cases}$$

10 Evaluate three new solutions $y^i = (y^{i1}, y^{i2}, y^{i3})$, choosing a best $\mathbf{y}^i$ according to non-dominated sort and degree of congestion.
11 Repair the gene of current solution $y_j^i = \begin{cases} a_j & x_j^i < a_j \\ b_j & x_j^i > b_j \quad j = \\ x_j^i & otherwise \end{cases}$ $1, 2, \ldots, n$.
12 **end if**
13 Return new solution $y^i = (y_1^i, \cdots, y_n^i)$.

Algorithm 4 *EnvironmentSel*$(A \cup P)$

Set the reference points.
Add the trial solution y^i to archive population A to form $P' = A \cup \{y^i\}$.
for $i = 1$ **to** $N + 1$ **do**
  Calculate number of dominating points dp^i to each individual $x^i, x^i \in P'$.
end for
if $\exists dp \neq 0$ **then**
  Delete the individual with the largest dp from P'.
end if
for $i = 1$ **to** $N + 1$ **do**
  Calculate hyper volume HV_i of each individual $x^i, x^i \in P'$.
end for
Delete the individual with smallest HV_i from P'.
$P = P'$.

$\varepsilon = 10^{-10}$ to guarantee the validity of the calculation. $GN_k^{MOPSO(CODE)}$ is the number of new solutions generated,

Algorithm 5 *Update*1 $\left(RF^{code(mopso)}, GN^{code(mopso)}\right)$

1 Find $GN_k^{MOPSO(CODE)}$ the number of new solution generated by MOPSO (CoDE).
2 Calculate $RF_k^{MOPSO(CODE)}$ for all the individuals generated by MOPSO (CoDE) in the generation k.
3 Calculate the reproduction utility by MOPSO (CoDE)

$$u_t^{MOPSO(CODE)} = \begin{cases} \sum_{k=t-HL+1}^{t} RF_k^{MOPSO(CODE)} \\ \quad / \sum_{k=t-HL+1}^{t} GN_k^{MOPSO(CODE)} & t > HL \\ \sum_{k=1}^{t} RF_k^{MOPSO(CODE)} \\ \quad / \sum_{k=1}^{t} GN_k^{MOPSO(CODE)} & t < HL \end{cases}.$$

4 Calculate control possibility β by $\beta_{t+1} = (u_t^{MOPSO} + \varepsilon)/(u_t^{MOPSO} + u_t^{CODE} + \varepsilon)$.

Algorithm 6 *Update*2 $\left(RF^{code(mopso)}, GN^{code(mopso)}\right)$

1 Calculate the total number of generating solutions using clusters (populations) in the past generation HL.

$$NR^{clu(gsp)} = \begin{cases} \sum_{k=t-HL+1}^{t} NT_k^{clu(gsp)} & t < HL \\ \sum_{k=1}^{t} NT_k^{clu(gsp)} & t > HL \end{cases}$$

2 Calculate the total number of individuals survived from environmental selection.

$$SNR^{clu(gsp)} = \begin{cases} \sum_{k=t-HL+1}^{t} NS_k^{clu(gsp)} & t < HL \\ \sum_{k=1}^{t} NS_k^{clu(gsp)} & t > HL \end{cases}$$

3 Calculate the utility $ut_t^{clu(gsp)} = SNR^{clu(gsp)}/(NR^{clu(gsp)} + \varepsilon)$.
4 Calculate mating restriction probability $\alpha_{t+1} = (ut_t^{clu} + \varepsilon)/(ut_t^{clu} + u_t^{gsp} + \varepsilon)$.

and $RF_k^{MOPSO(CODE)}$ represents the sum of fitness values of all the individuals generated by MOPSO(CoDE) the generation k.

Furthermore, population in CoDE-MOPSO is divided into several clusters by k-means algorithm. The mating parents are chosen from the whole population for exploration or selected from the clusters for exploitation. In order to balance exploration and exploitation, the mating restriction probability is used to control the source of the mating parents. In each generation, the mating restriction probability is updated. The detail is dictated in Algorithm 6.

IV. RESULTS AND DISCUSSION

A. DESIGN OF EXPERIMENTS

This subsection outlines the comprehensive experimental setup designed to evaluate the effectiveness and efficiency of the CoDE-MOPSO algorithm. Our objective is to demon-

strate the algorithm's capabilities in addressing complex multi-objective optimization problems and to compare its performance with that of other state-of-the-art algorithms.

In the following experiments, performance of CoDE-MOPSO will be tested by six benchmark instances GLT1-GLT6 [29] with complex PF shapes, and two quality metrics named inverse generational distance (IGD) [30] and hypervolume (HV) [31] are employed. Both IGD and HV are able to measure convergence and diversity of approximation PF meanwhile. Obviously, small value of IGD and large value of HV mean the approximation PF has fast convergence and good diversity. Statistical tests, such as the Wilcoxon rank-sum test, will be conducted to determine the statistical significance of the observed differences in performance between CoDE-MOPSO and other algorithms.

In the test instances, metric value of HV is calculated with the reference points as r=(2,2)T in GLT1, r=(2,11)T in GLT2, r=(2,2)T in GLT3, r=(2,3)T in GLT4, r=(2,2,2)T in GLT5-GLT6.

CoDE-MOPSO will be compared with four other typical multi-objective multi-objective algorithms: NSGA-II, SMS-EMOA, RM-MEDA [30] and TMOEA/D [32], [33]. To ensure the comparison fairness, these algorithms will choose the corresponding optimum parameters and be implemented by MATLAB in the same computer. All the parameter settings are listed in Table 1.

B. COMPARATIVE EXPERIMENTS

To get a convincing statistics conclusion, the averages and standard deviations of the IGD and HV metric values are obtained from 33 independent runs on each test instance. In Table 2, the means of IGD (HV) metric values on each test question are sorted in an ascending (descending) order and filled in the square brackets, and the average values (average rank) of IGD and HV obtained by each algorithm to the GLT1-GLT6 questions are also listed. Wilcoxon's rank sum test at a 5% significance level is employed between CoDE-MOPSO and comparison algorithms. Symbols "†", "§", and "≈" denote that the performance of CoDE-MOPSO is better than, worse than, and similar to that of the comparison algorithm according to Wilcoxon's rank sum test, respectively.

C. EXPERIMENTAL RESULTS FOR SEARCH QUALITY

In Table 2, the statistical results of HV and IGD are calculated from 33 independent runs of GLT tests for NSGA-II, SMS-EMOA, RM-MED, TMOEA/D, and CoDE-MOPSO algorithm. It can be seen that CoDE-MOPS gets 6 optimum and 6 suboptimum metric values under two quality metrics for GLT1-GLT6. According to Wilcoxon's rank sum test, compared to NSGA-II, SMS-EMOA, RM-MEDA, and TMOEA/D with 12 times comparison, CoDE-MOSPO reaches 12, 12, 12, and 6 significantly better mean metric values, respectively. Besides, the mean rank value shows that in solving GLT tests, the resorting order of the performance is CoDE-MOSPO, TMOEA/D, SMS-EMOA, RM-MEDA, and NSGA-II. To sum up, CoDE-MOPSO has the optimal solution performance.

D. EXPERIMENTAL RESULTS FOR SEARCH EFFICIENCY

The evolution curves of IGD mean value derived from 33 times independent runs on GLT test for NSGA-II, SMS-EMOA, RM-MED, TMOEA/D, and CoDE-MOPSO algorithm are depicted in Figure 1. For GLT 3 and GLT 5-GLT 6, CoDE-MOPSO receives minimum mean IGD metric values with the smallest numbers of evolution. For GLT 2, CoDE-MOPSO gets a similar solution effect as well as TMOEA/D. For GLT 1 and GLT 4, CoDE-MOPSO archives the second lowest mean IGD values. Anyhow, compared with the other four algorithms, CoDE-MOPSO exhibits better convergence and diversity in the evolutionary procedure on GLT tests.

E. EXPERIMENTAL RESULTS FOR APPROXIMATED FRONTS OF GLTS

In order to show the superiority of CoDE-MOPSO, the two best algorithms in Table 2 are selected to run 33 times on GLT test, both the approximation fronts of CoDE-MOPSO algorithm and TMOEA/D algorithm obtained from runs (See Figure 2 (a) and Figure 2 (b))and the representative approximation fronts corresponding to average IGD metric value (See Figure 2 (c) and Figure 2 (d)) are shown in Figure 2. From Figure 2 (a) and Figure 2 (b), we can observe that for GLT 1–GLT 3 and GLT 5 – GLT-6, the approximated front obtained by CoDE-MOPSO can steadily converge to PFs and evenly cover them. But for the GLT 4 – GLT 6 test, the approximated front obtained by TMOEA/D could neither converge to PFs completely nor cover the whole PFs. From Figure 2 (c) and Figure 2 (d), it can be seen that CoDE-MOPSO could successfully converge to PFs and cover them on GLT1-GLT6 test except on GLT 3 test, while the representative approximated front obtained by TMOEA/D for GLT 3–GLT 6, shows a poor uniformity of distribution and at the same time, some of the representative approximated fronts for GLT 5–GLT could not converge to PFs. The visible results indicate CoDE-MOPSO is prior to TMOEA/D on GLT tests.

F. COMPUTATIONAL COMPLEXITY ANALYSIS

The computational complexity of the CoDE-MOPSO algorithm primarily stems from three components: the Composite Differential Evolution operations, the Multi-Objective Particle Swarm Optimization procedures, and the K-means clustering process integrated into the framework.

The complexity of CoDE is influenced by the differential mutation and crossover operations. For a population of size N, with D dimensions, the complexity of generating new solutions using CoDE is $O(N \cdot D)$, considering each individual requires D operations for mutation and crossover.

MOPSO involves calculating velocities and updating positions for each particle, which also operates with a complexity of $O(N \cdot D)$. Additionally, the complexity of updating personal

TABLE 1. Common Parameters to solve GLT Test Suite and Parameters of NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, CoDE-MOPSO.

Algorithm	Parameter Settings
public parameters	Population size N=100 (105) for two (three) objective mops; Variable dimension n=10; maximum number of generations T=300.
NSGA-II	DE parameters $F=0.5, CR=1$; Mutation parameters $p_m=1/\mathrm{n}, \eta_m=20$.
SMS-EMOA	DE parameters $F=0.5, CR=1$; Mutation parameters $p_m=1/\mathrm{n}, \eta_m=20$.
RM-MEDA	Number of clusters in local PCA is $K=5$; Maximal number of iteration in local PCA is $M_n=50$. Extension rate of sampling is $\eta=0.25$.
TMOEA/D	Neighbor size $\mathrm{NS}=30$; Number of generation in first search stage $T1=T/10$; Number of generation in the 2nd search stage $T2=aT$. α= {0.01, 0.02… 0.1, 0.1, 0.1, 0.15}. DE control parameters $F=0.3, CR=1$
CoDE-MOPO	Velocity update parameters w=0.6;c1=2;c2=2; Maximum number of clusters $K=5$;History length $HL=20$; DE control parameters : $F=1.0, CR=0.1$, $F=1.0, CR=0.9$, $F=0.8, CR=0.2$ Mating restriction probability $\alpha=0.7$; Control probability $\beta_0=0.5$; Mutation operators $p_m=1/\mathrm{n}, \eta_m=20$.

TABLE 2. Statistical results (Mean(Std.Dev.)[rank]0 of IGD and HV values of approximated fronts obtain by NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D and CoDE-MOPSO, respectively, over 33 independent runs on the GLT test suite.

Instance	NSGA-II	SMS-EMOA	RM-MEDA	TMOEA/D	CoDE-MOPSO
			IGD		
GLT1	3.145e-01$^{\dagger}_{2.39e-01}$[5]	5.723e-02$^{\dagger}_{2.71e-02}$[4]	2.756e-02$^{\dagger}_{1.78e-02}$[3]	6.714e-03$^{\approx}_{5.30e-04}$[1]	9.900e-03$_{03}^{6.50e-}$[2]
GLT2	2.834e+00$^{\dagger}_{9.87e+01}$[5]	8.910e-02$^{\dagger}_{4.13e-02}$[4]	7.233e-02$^{\dagger}_{9.26e-02}$[3]	5.684e-02$^{\S}_{4.58e-02}$[1]	5.780e-02$_{03}^{2.70e-}$[2]
GLT3	2.495e-01$^{\dagger}_{1.00e-01}$[5]	5.175e-02$^{\dagger}_{1.75e-02}$[3]	3.270e-02$^{\dagger}_{1.46e-02}$[2]	8.341e-02$^{\dagger}_{9.75e-02}$[4]	2.440e-02$_{02}^{1.77e-}$[1]
GLT4	8.123e-01$^{\dagger}_{1.96e-01}$[5]	1.356e-01$^{\dagger}_{9.05e-02}$[4]	6.879e-02$^{\dagger}_{5.40e-02}$[3]	2.385e-02$^{\S}_{3.75e-02}$[1]	4.010e-02$_{02}^{6.01e-}$[2]
GLT5	1.830e-01$^{\dagger}_{6.07e-02}$[5]	3.991e-02$^{\dagger}_{1.56e-03}$[2]	7.895e-02$^{\dagger}_{4.41e-03}$[4]	6.241e-02$^{\dagger}_{2.23e-03}$[3]	2.950e-02$_{04}^{3.69e-}$[1]
GLT6	1.385e-01$^{\dagger}_{5.21e-02}$[5]	3.162e-02$^{\dagger}_{3.07e-03}$[2]	6.118e-02$^{\dagger}_{4.06e-03}$[3]	6.344e-02$^{\dagger}_{3.76e-02}$[4]	2.200e-02$_{04}^{5.42e-}$[1]
			HV		
GLT1	2.736e+00$^{\dagger}_{4.51e-01}$[5]	3.205e+00$^{\dagger}_{5.31e-01}$[4]	3.280e+00$^{\dagger}_{3.90e-02}$[3]	3.367e+00$^{\S}_{1.26e-03}$[1]	3.336e+00$_{02}^{2.27e-}$[2]
GLT2	1.100e+01$^{\dagger}_{3.73e+00}$[5]	1.961e+01$^{\dagger}_{1.77e-01}$[3]	1.957e+01$^{\dagger}_{2.52e-01}$[4]	1.972e+01$^{\S}_{1.79e-02}$[1]	1.971e+01$_{03}^{7.40e-}$[2]
GLT3	3.820e+00$^{\dagger}_{3.19e-01}$[5]	3.938e+00$^{\dagger}_{3.47e-03}$[3]	3.942e+00$^{\dagger}_{2.60e-03}$[2]	3.924e+00$^{\dagger}_{3.26e-02}$[4]	3.943e+00$_{03}^{3.30e-}$[1]
GLT4	2.697e+00$^{\dagger}_{8.19e-01}$[5]	4.805e+00$^{\dagger}_{2.10e-01}$[4]	4.927e+00$^{\dagger}_{6.90e-02}$[3]	4.966e+00$^{\S}_{8.62e-02}$[1]	4.930e+00$_{01}^{1.43e-}$[2]
GLT5	7.811e+00$^{\S}_{8.12e-02}$[5]	7.963e+00$^{\S}_{6.69e-04}$[2]	7.929e+00$^{\S}_{3.70e-03}$[4]	7.946e+00$^{\S}_{2.00e-03}$[3]	7.969e+00$_{04}^{1.73e-}$[1]
GLT6	7.805e+00$^{\dagger}_{6.76e-02}$[5]	7.955e+00$^{\dagger}_{2.57e-03}$[2]	7.930e+00$^{\dagger}_{3.45e-03}$[4]	7.934e+00$^{\dagger}_{2.25e-02}$[3]	7.962e+00$_{04}^{4.06e-}$[1]
Mean rank	5.000	3.083	3.167	2.250	1.500
†/§/≈	12/0/0	12/0/0	12/0/0	6/5/1	

and global bests depends on the comparison operations, which are $O(N)$ in the worst case.

The complexity of K-means clustering is $O(N \cdot K \cdot D \cdot I)$, where K is the number of clusters and I is the number of iterations until convergence. While K and I are typically much smaller than N, they introduce an additional computational burden.

In aggregate, the overall computational complexity of the CoDE-MOPSO algorithm can be approximated as $O(N \cdot D \cdot (1 + I \cdot K))$, acknowledging that the most significant factors are the population size, dimensionality of the problem, and the K-means iterations.

G. COMPARISON OF CODE AND MOPSO

we have added comparisons with algorithms related to the Composite Differential Evolution (CoDE) and Multi-Objective Particle Swarm Optimization (MOPSO) in our study. In the Table 3, we provided statistical results (Mean and Standard Deviation) of Inverse Generational Distance (IGD) and Hypervolume (HV) values for CoDE, MOPSO, and our proposed CoDE-MOPSO algorithm across various instances of the GLT test suite.

The CoDE-MOPSO algorithm outperforms individual CoDE and MOPSO algorithms across various test instances in terms of IGD and HV metrics. Specifically, CoDE-MOPSO consistently achieved lower IGD values, indicating superior precision in approximating the Pareto front, and higher HV values, demonstrating greater coverage of the true Pareto front. For example, in the GLT1 test, while CoDE and MOPSO scored 2.170 and 3.231 respectively, CoDE-MOPSO achieved a superior value of 3.336.

Furthermore, the integration of CoDE's diversity preservation with MOPSO's fast convergence in the CoDE-MOPSO algorithm has proven effective, providing a robust solution that performs consistently across different scenarios (GLT1

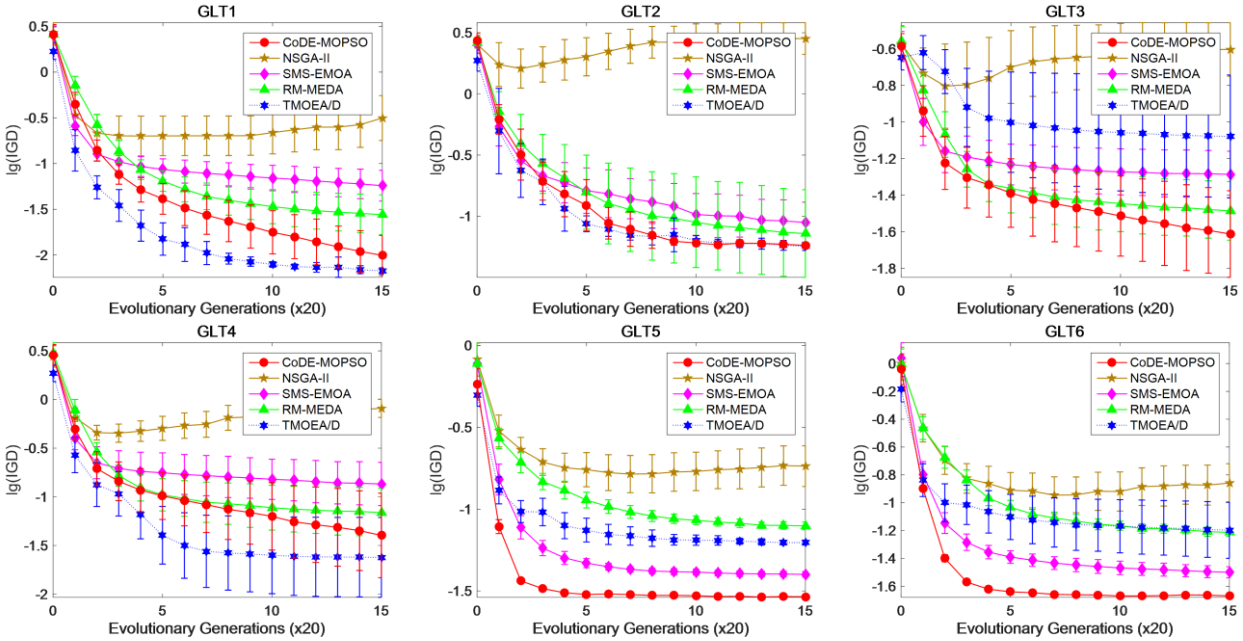


FIGURE 1. Evolution of the mean IGD metric value.

TABLE 3. Statistical results (Mean(Std.Dev.) of IGD and HV values of approximated fronts obtain by CoDE, MOPSO and CoDE-MOPSO, respectively, over 33 independent runs on the GLT test suite.

Instance	CoDE	MOPSO	CoDE-MOPSO
		IGD	
GLT1	5.478e-01$_{3.28e\text{-}01}$	2.367e-02$_{5.50e\text{-}02}$	9.900e-03$_{6.50e\text{-}03}$
GLT2	3.962e+00$_{9.37e\text{-}01}$	5.980e-02$_{2.70e\text{-}01}$	5.780e-02$_{2.70e\text{-}03}$
GLT3	2.750e-01$_{1.66e\text{-}01}$	3.540e-02$_{1.77e\text{-}02}$	2.440e-02$_{1.77e\text{-}02}$
GLT4	8.839e-01$_{2.30e\text{-}01}$	4.310e-02$_{7.01e\text{-}02}$	4.010e-02$_{6.01e\text{-}02}$
GLT5	2.351e-01$_{5.97e\text{-}02}$	3.852e-02$_{4.79e\text{-}04}$	2.950e-02$_{3.69e\text{-}04}$
GLT6	1.782e-01$_{5.80e\text{-}02}$	2.520e-02$_{3.86e\text{-}04}$	2.200e-02$_{5.42e\text{-}04}$
		HV	
GLT1	2.170e+00$_{4.82e\text{-}01}$	3.231e+00$_{7.37e\text{-}02}$	3.336e+00$_{2.27e\text{-}02}$
GLT2	1.025e+01$_{4.06e+00}$	1.962e+01$_{5.30e\text{-}02}$	1.971e+01$_{7.40e\text{-}03}$
GLT3	3.505e+00$_{2.89e\text{-}01}$	3.861e+00$_{3.47e\text{-}03}$	3.943e+00$_{3.30e\text{-}03}$
GLT4	2.258e+00$_{8.62e\text{-}01}$	4.952e+00$_{1.26e\text{-}01}$	4.930e+00$_{1.43e\text{-}01}$
GLT5	7.152e+00$_{8.55e\text{-}02}$	7.958e+00$_{1.63e\text{-}04}$	7.969e+00$_{1.73e\text{-}04}$
GLT6	7.140e+00$_{7.15e\text{-}02}$	7.959e+00$_{2.16e\text{-}04}$	7.962e+00$_{4.06e\text{-}04}$

to GLT6). This synergy enhances the algorithm's ability to balance exploration and exploitation, which is critical for navigating complex Pareto fronts in multi-objective optimization. These findings confirm the efficacy of our approach, addressing the limitations of existing methods.

H. PERFORMANCE ON WFG TEST SUITE

In order to verify the performance of CoDE-MOPSO further, WFG test instances [34], [35] are chosen, because they have complex Pareto front shapes and complicated Pareto set characteristics. By similar statistic calculations as in Section IV, NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, and CoDE-MOPSO will run each WFG test 33 times. The optimal parameter settings are shown in Table 4. The HV metric value is obtained with the reference point (3, 5) T for all instances. The comparison results are listed in Table 5. It can be found that compared with each of the other algorithms for 18 times, NSGA-II, SMS-EMOA, RM-MEDA, and TMOEA/D, CoDE-MOPSO obtained 18, 9, 15 and 15 significant superior, 0, 8, 3 and 2 significant substandard, and 0, 1, 0 and 1 non-differential mean index value. According to Wilcoxon's rank sum test, CoDE-MOPSO performs similarly to SMS-EMOA on WFG tests, while on GLT tests, it shows an advantage over SMS-EMOA.

I. EFFECTIVENESS OF CLUSTER AND CODE

The effectiveness of cluster and composite difference operators in CoDE-MOPSO is demonstrated by the comparison with CoDE-MOPSO-NoC and DEPSO based on GLT tests. In fact, CoDE-MOPSO-NoC is a simplified CoDE-MOPSO

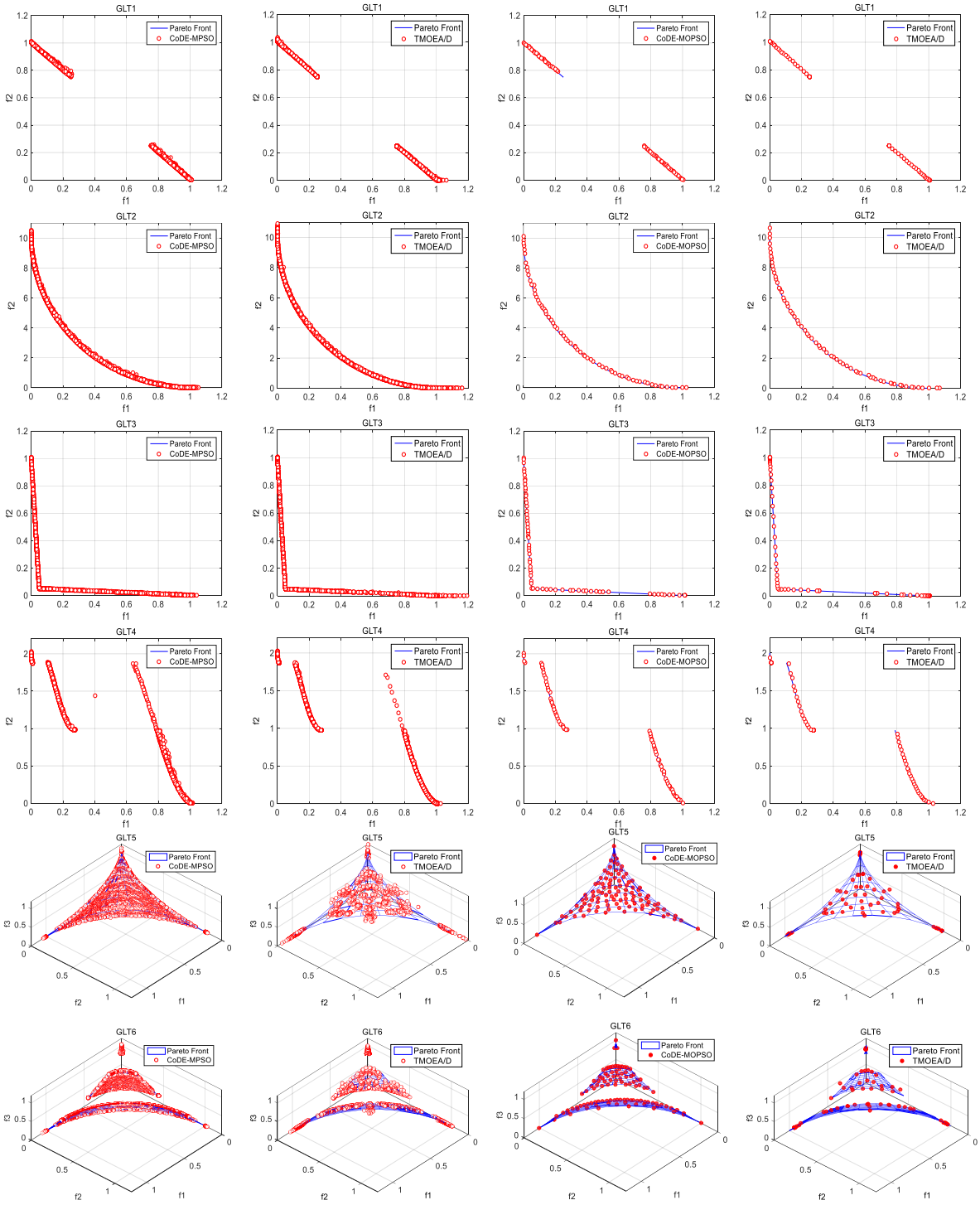


FIGURE 2. Approximated fronts obtained by CoDE-MOPSO and TMOEA/D. (a) All AFs of CoDE-MOPSO; (b) All AFs of TMOEA/D; (c) Representative AFs of CoDE-MOPSO; (d) Representative AFs of TMOEA/D.

without cluster technique, that is, it randomly selects a parent individual in the whole population to recombine. DEPSO is another simplified CoDE-MOPSO with the CoDE operator replaced by DE operator. Here, the control parameter setting of DE operator is chosen as F=0.3, CR=0.8 and other parameter settings for the three algorithms are the same as in Section IV. Each of the three algorithms executes 33 times GLT tests and then the statistic results are shown in Table 6.

TABLE 4. Common Parameter to solve WFG Test Suite and Parameters of NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, CoDE-MOPSO.

Algorithm	Parameter setting
public parameters	n=30, T=450, N=100
NSGA-II	F=0.3, CR=0.2
SMS-EMOA	F=0.5, CR=0.2
RM-MEDA	K=5
TMOEA/D	F=0.5, CR=0.2
CoDE-MOPO	F=0.3, CR=0.2, $\alpha_0 = 0.3, \beta_0 = 0.7$
	K=4, HL=5
	[F=1.0, CR=0.1], [F=1.0, CR=0.9], [F=0.8, CR=0.2]

TABLE 5. Statistical results (Mean(Std.Dev.)[rank] of IGD and HV values of approximated fronts obtain by NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, CoDE-MOPSO, respectively, over 33 independent runs on the WFG test suite.

Instance	NSGA-II	SMS-EMOA	RM-MEDA	TMOEA/D	CoDE-MOPSO
			IGD		
WFG1	1.453e+00$^{\dagger}_{6.33e-02}$[4]	1.618e+00$^{\dagger}_{2.68e-02}$[5]	1.170e+00$^{\dagger}_{7.89e-03}$[3]	9.280e-01$^{\S}_{1.66e-02}$[1]	1.390e+00$_{02}^{6.50e-}$[2]
WFG2	1.455e-02$^{\dagger}_{7.04e-04}$[3]	1.143e-02$^{\approx}_{6.94e-04}$[1]	3.176e-02$^{\dagger}_{5.39e-03}$[5]	2.862e-02$^{\dagger}_{3.22e-03}$[4]	1.280e-02$_{04}^{7.70e-}$[2]
WFG3	1.451e-01$^{\dagger}_{7.30e-04}$[4]	1.144e-01$^{\S}_{2.19e-04}$[1]	1.594e-01$^{\dagger}_{4.97e-03}$[5]	1.247e-02$^{\dagger}_{6.41e-04}$[3]	1.174e-02$_{05}^{8.77e-}$[2]
WFG4	5.865e-02$^{\dagger}_{5.29e-03}$[4]	1.096e-02$^{\S}_{3.39e-03}$[1]	9.356e-02$^{\dagger}_{2.88e-03}$[5]	4.863e-02$^{\dagger}_{1.30e-02}$[3]	2.986e-02$_{03}^{6.01e-}$[2]
WFG5	6.915e-02$^{\S}_{3.62e-04}$[2]	6.654e-02$^{\S}_{8.78e-05}$[1]	1.149e-01$^{\dagger}_{1.81e-02}$[5]	8.045e-02$^{\dagger}_{1.84e-02}$[4]	6.950e-02$_{04}^{3.69e-}$[3]
WFG6	3.838e-01$^{\dagger}_{1.15e-02}$[5]	3.411e-01$^{\dagger}_{1.69e-02}$[3]	3.270e-01$^{\dagger}_{4.15e-03}$[2]	3.510e-01$^{\dagger}_{1.82e-02}$[4]	3.246e-01$_{03}^{6.01e-}$[1]
WFG7	3.002e-02$^{\dagger}_{1.94e-03}$[3]	1.390e-02$^{\dagger}_{1.79e-04}$[2]	4.147e-02$^{\dagger}_{6.55e-03}$[4]	6.389e-02$^{\dagger}_{7.86e-02}$[5]	1.286e-02$_{04}^{1.91e-}$[1]
WFG8	6.130e-02$^{\dagger}_{8.14e-03}$[4]	3.590e-02$^{\S}_{6.44e-03}$[1]	1.693e-01$^{\dagger}_{1.38e-03}$[5]	4.823e-02$^{\approx}_{3.54e-02}$[2]	4.986e-02$_{03}^{6.01e-}$[3]
WFG9	2.717e-01$^{\dagger}_{1.06e-02}$[5]	2.679e-01$^{\dagger}_{1.62e-02}$[3]	2.077e-01$^{\S}_{2.36e-02}$[1]	2.687e-01$^{\dagger}_{2.46e-02}$[4]	2.530e-01$_{02}^{4.42e-}$[2]
			HV		
WFG1	4.362e+00$^{\dagger}_{2.45e-01}$[4]	4.043e+00$^{\dagger}_{2.26e-01}$[5]	5.610e+00$^{\S}_{4.19e-02}$[2]	6.747e+00$^{\S}_{8.53e-02}$[1]	4.836e+00$_{01}^{2.27e-}$[3]
WFG2	1.141e+01$^{\dagger}_{3.88e-03}$[4]	1.146e+01$^{\S}_{4.14e-04}$[1]	1.126e+01$^{\dagger}_{4.03e-02}$[5]	1.143e+01$^{\dagger}_{3.79e-03}$[3]	1.144e+01$_{03}^{2.40e-}$[2]
WFG3	1.088e+01$^{\dagger}_{6.46e-03}$[4]	1.096e+01$^{\S}_{3.73e-04}$[1]	1.078e+01$^{\dagger}_{3.96e-02}$[5]	1.093e+01$^{\S}_{3.85e-03}$[3]	1.094e+01$_{03}^{3.30e-}$[2]
WFG4	8.343e+00$^{\S}_{3.40e-02}$[4]	8.670e+00$^{\S}_{2.44e-03}$[1]	8.114e+00$^{\dagger}_{2.03e-02}$[5]	8.352e+00$^{\S}_{1.43e-01}$[3]	8.430e+00$_{02}^{4.43e-}$[2]
WFG5	8.160e+00$^{\dagger}_{6.37e-02}$[2]	8.144e+00$^{\dagger}_{5.02e-02}$[3]	7.931e+00$^{\dagger}_{7.30e-02}$[4]	7.736e+00$^{\dagger}_{3.55e-01}$[5]	8.179e+00$_{02}^{5.73e-}$[1]
WFG6	6.320e+00$^{\dagger}_{5.99e-02}$[3]	6.303e+00$^{\dagger}_{8.63e-02}$[4]	6.377e+00$^{\dagger}_{2.28e-02}$[2]	6.259e+00$^{\dagger}_{6.74e-02}$[5]	6.409e+00$_{02}^{4.28e-}$[1]
WFG7	8.578e+00$^{\dagger}_{8.25e-03}$[3]	8.662e+00$^{\dagger}_{5.89e-02}$[2]	8.456e+00$^{\dagger}_{4.33e-02}$[4]	7.857e+00$^{\dagger}_{8.79e-01}$[5]	8.670e+00$_{03}^{2.73e-}$[1]
WFG8	8.336e+00$^{\dagger}_{4.29e-02}$[3]	8.511e+00$^{\S}_{5.99e-02}$[1]	7.638e+00$^{\dagger}_{8.10e-02}$[5]	8.275e+00$^{\dagger}_{2.69e-01}$[4]	8.379e+00$_{02}^{4.76e-}$[2]
WFG9	6.100e+00$^{\dagger}_{5.69e-02}$[4]	6.122e+00$^{\S}_{8.02e-01}$[3]	6.411e+00$^{\S}_{2.16e-02}$[1]	6.026e+00$^{\dagger}_{1.27e-01}$[5]	6.162e+00$_{01}^{1.86e-}$[2]
Mean rank	3.222	2.167	3.778	3.556	1.889
†/§/≈	18/0/0	9/8/1	15/3/0	15/2/1	

TABLE 6. Statistical results (Mean(Std.Dev.)[rank] of IGD and HV values of approximated fronts obtain by DEPSO, CoDE-MOPSO-NoC, CoDE-MOPSO, respectively, over 33 independent runs on the GLT test suite.

Instance	DEPSO	CoDE-MOPSO-NoC	CoDE-MOPSO
		IGD	
GLT1	1.068e-02$^{\dagger}_{3.41e-02}$[2]	3.467e-02$^{\dagger}_{1.15e-02}$[3]	9.900e-03$_{03}^{6.50e-}$[1]
GLT2	5.967e-02$^{\dagger}_{1.07e-01}$[2]	1.200e-01$^{\dagger}_{2.24e-02}$[3]	5.780e-02$_{03}^{2.70e-}$[1]
GLT3	3.565e-02$^{\dagger}_{1.68e-02}$[3]	2.488e-02$^{\dagger}_{8.59e-03}$[2]	2.440e-02$_{02}^{1.77e-}$[1]
GLT4	4.644e-02$^{\dagger}_{7.83e-02}$[2]	8.913e-02$^{\dagger}_{4.47e-02}$[3]	4.010e-02$_{02}^{6.01e-}$[1]
GLT5	3.986e-02$^{\dagger}_{7.47e-04}$[3]	3.576e-02$^{\dagger}_{9.78e-04}$[2]	2.950e-02$_{04}^{3.69e-}$[1]
GLT6	2.958e-02$^{\dagger}_{3.79e-04}$[3]	2.680e-02$^{\dagger}_{1.03e-03}$[2]	2.200e-02$_{04}^{5.42e-}$[1]
		HV	
GLT1	3.212e+00$^{\dagger}_{7.61e-02}$[2]	3.207e+00$^{\dagger}_{3.67e-02}$[3]	3.336e+00$_{02}^{2.27e-}$[1]
GLT2	1.962e+01$^{\dagger}_{2.18e-01}$[2]	1.927e+01$^{\dagger}_{5.84e-02}$[3]	1.971e+01$_{03}^{7.40e-}$[1]
GLT3	3.942e+01$^{\dagger}_{3.11e-03}$[1]	3.941e+01$^{\dagger}_{1.53e-03}$[3]	3.943e+00$_{03}^{3.30e-}$[2]
GLT4	4.998e+00$^{\dagger}_{1.12e-01}$[1]	4.893e+00$^{\dagger}_{2.37e-02}$[3]	4.930e+00$_{01}^{1.43e-}$[2]
GLT5	7.967e+00$^{\S}_{1.92e-04}$[2]	7.962e+00$^{\dagger}_{5.87e-04}$[3]	7.969e+00$_{04}^{1.73e-}$[1]
GLT6	7.958e+00$^{\S}_{2.73e-04}$[2]	7.955e+00$^{\dagger}_{9.19e-04}$[3]	7.962e+00$_{04}^{4.06e-}$[1]
Mean rank	3.222	2.167	3.778
†/§/≈	18/0/0	9/8/1	15/3/0

In Table 6, it can be found that CoDE-MOPSO gained 10 best results of 12 compared with DEPSO and CoDE-MOPSO-NoC. Such results highlight two advantages of CoDE-MOPSO algorithm. On the one hand, CoDE-MOPSO algorithm utilizes a clustering technique to assemble similar individuals in one cluster and select parents from the whole population or the same cluster with a certain mating restriction probability, which speeds up CoDE-MOPSO's searching

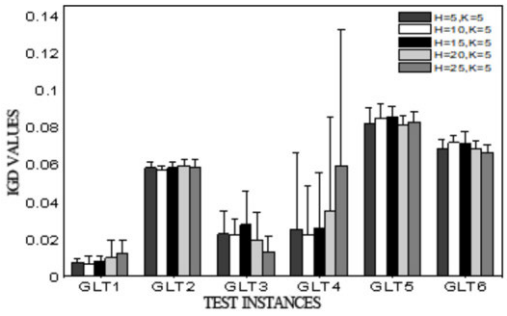


FIGURE 3. History length (HL).

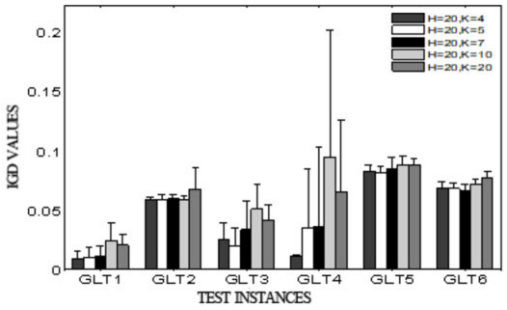


FIGURE 4. Analysis of clustering number (K).

performance. On the other hand, CoDE-MOPSO utilizes three complementary DE operators and control parameters, making it outperform DEPSO algorithm on GLT tests. In all, CoDE-MOPSO has a certain advantage of converging and distributing along PF for GLT test instances than DEPSO and CoDE-MOPSO-NoC.

J. SENSITIVITY ANALYSIS ON PARAMETERS

History length HL determines value of control possibility β which directly decides the proportion of MOPSO and CoDE operator and further influences the performance of CoDE-MOPSO. In order to analyze the influence of HL, we set $HL = 5, 10, 15, 20, 25$ in CoDE-MOPSO and respectively with all the other parameters same as Section IV. By 33 times independent run on GLT instances, the means and standard deviations of IGD metric values obtained from CoDE-MOPSO with different HL are depicted in Figure 3. We can see that different HL value will cause great fluctuation of mean IGD metric value on GLTs except on GLT 2. On GL1, GL3 and GLT4 – GLT6 test, smallest mean IGD metric values are achieved when $HL = 20$, so this value is a good choice in the experiments.

In the CoDE-MOPSO algorithm, K-means method is employed to explore the distribution information of each individual in the population. In order to explore the influence of the largest cluster number, CoDE-MOPSO algorithms with different K will be executed on GLT tests. Other parameters in CoDE-MOPSO algorithm are the same as in Section IV. Each

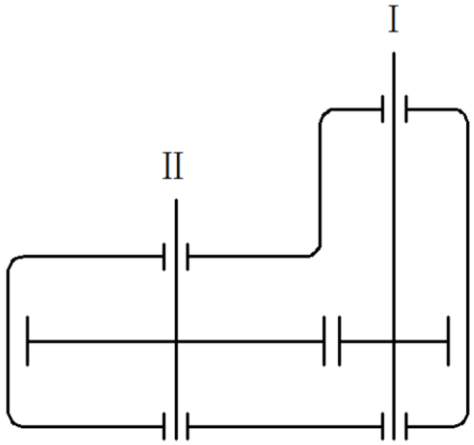


FIGURE 5. Schematic diagram of structure of gear reducer.

TABLE 7. Common Parameter to solve WFG Test Suite and Parameters of NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, CoDE-MOPSO.

Algorithm	Parameter setting
NSGA-II	$P_c = 0.9, P_m = 1/7$
SMS-EMOA	$P_c = 0.9, P_m = 1/7$
RM-MEDA	$F = 0.5, CR = 1$
TMOEA/D	$K = 5$
CoDE-MOPO	$H = 10, K = 20$

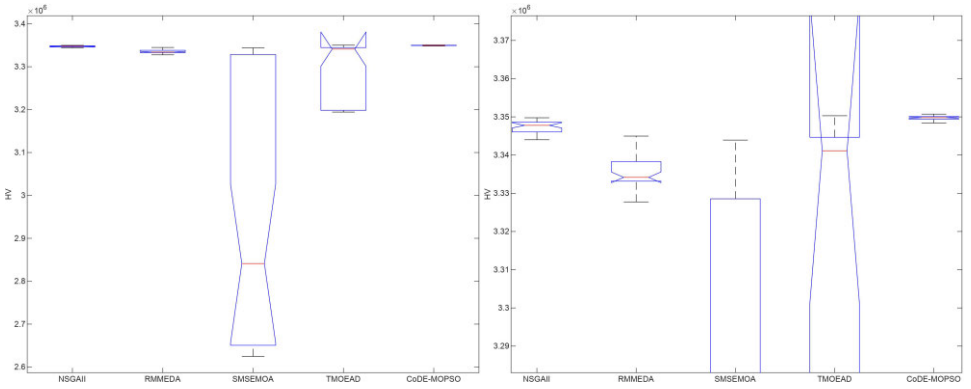
algorithm with a different value of K should be independently calculated 33 times for each test. The obtained mean and standard deviations of the IGD metric values are shown in Figure 4.

From Figure 4, it shows that for GLT1 and GLT3–GLT4 tests, the mean IGD values obtained by CoDE-MOPSO algorithms with different K values exhibit significant difference, while for the other tests, the mean IGD values are almost same. When $K = 5$, IGD metric values are all minima for GLT 1, GLT 3 and GLT 5, and for the other tests, mean IGD values are comparatively smaller. Therefore, $K = 5$ will be selected in the experiments.

V. APPLICATION OF CoDE-MOPSO ALGORITHM IN OPTIMUM DESIGN OF REDUCER

A. MATHEMATICAL MODEL

In order to test the effectiveness of CoDE-MOPSO algorithm in solving real problems, optimizing the problem for the volume of certain two-grader cylinder gear reducer and the stress of Axis is considered in the section. The simplified model of the gear reducer [36], [37] is shown in Figure 5. MOP is designed to minimize the volume f_{volume} of gear reducer and the stress f_{stress} of Axis II. The mathematical



(a) (b)

FIGURE 6. HV indicator box diagram. (a) Original photo; (b) Partial enlarged photo.

model is described as follows.

$$\begin{cases} \min f_{volume} = 0.748x_1x_2^2(10x_3^2/3 + 14.933x_3 - 43.093) \\ -1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) \\ \quad +0.785(x_4x_6^2 + x_5x_7^2) \\ \min f_{stress} = \sqrt{(745x_4/x_2x_3)^2 + 1.69 \times 10^7}/0.1x_6^3 \end{cases} \tag{7}$$

$$\begin{cases} g_1 : \dfrac{1}{x_1x_2^2x_3} - \dfrac{1}{27} \le 0 \quad g_2 : \dfrac{1}{x_1x_2^2x_3^2} - \dfrac{1}{397.5} \le 0 \\ g_3 : \dfrac{x_4^3}{x_2x_3x_6^4} - \dfrac{1}{1.93} \le 0 \quad g_4 : \dfrac{x_5^3}{x_2x_3x_7^4} - \dfrac{1}{1.93} \le 0 \\ g_5 : x_2x_3 - 40 \le 0 \quad g_6 : x_1/x_2 - 12 \le 0 \\ g_7 : 5 - x_1/x_2 \le 0 \quad g_8 : 1.9 - x_4 + 1.5x_6 \le 0 \\ g_9 : 1.9 - x_5 + 1.1x_7 \le 0 \quad g_{10} : f_{stress} \le 1300 \\ g_{12,13} : 2.6 \le x_1 \le 3.6 \quad g_{14,15} : 0.7 \le x_2 \le 0.8 \\ g_{16,17} : 17 \le x_3 \le 28 \quad g_{18,19} : 7.3 \le x_4 \le 8.3 \\ g_{20,21} : 7.3 \le x_5 \le 8.3 \quad g_{22,23} : 2.9 \le x_6 \le 3.9 \\ g_{24,25} : 5.0 \le x_1 \le 5.5 \\ g_{11} : \sqrt{(745x_4/x_2x_3)^2 + 1.575 \times 10^8}/0.1x_7^3 \le 1100 \end{cases} \tag{8}$$

where x_1 is gear width, x_2 is gear module, x_3 is pinion teeth number, x_4 is distance between two bearing of AxisI, x_5 is distance between two bearing of AxisII, x_6 is diameter of shaft of Axis I, x_7 is diameter of shaft of Axis II, g_1 is bending stress constraint of teeth, g_2 is contact stress constraint of teeth, g_3 and g_4 are deformation constraints of Axes, g_5, g_6 and g_7 are spatial dimension-based limitation and empirical constraint. g_8 and g_9 are requirements of design shaft by experience, g_{10} and g_{11} are the Axis stress constraints, g_{12} to g_{25} are maxima and minima values of 7 variables.

B. ALGORITHM PARAMETER SETTING

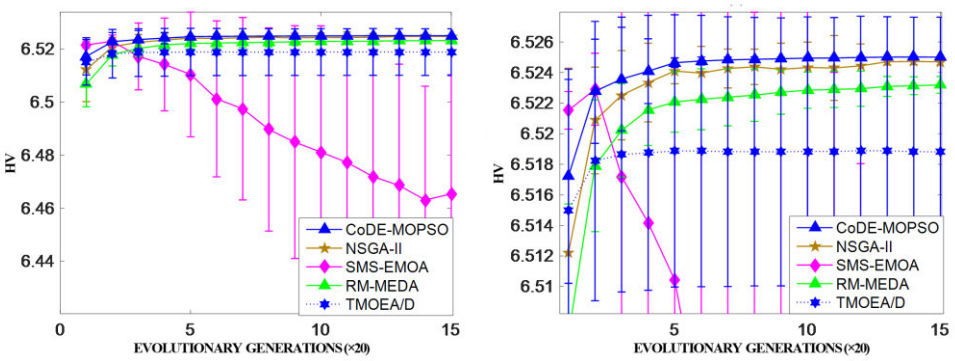
NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, CEDA, and CoDE-MOPSO are used to solve gear reducer optimal design model. Parameter settings are listed in Table 7. Other parameters are the same as in Section IV. Each algorithm calculates the model 33 times and takes the Hyper-volume HV metric value as the criterion to measure the obtained approximate front. Here, reference value r=[6600,1600]T is chosen to solve the HV value.

TABLE 8. Special solution of the optimization design model of gear reducer by CoDE-MOPSO.

x_1	x_2	x_3	x_4	x_5	x_6	x_7	f_{volume}	f_{stress}
3.500	0.7	17	7.37	7.43	3.17	5.0	2832.3	1294.8
3.504	0.7	20	7.84	7.42	3.9	5.0	5836.9	694.7
3.502	0.7	17	7.55	7.44	3.7	5.0	3599.4	696.6

C. ALGORITHM RESULTS AND ANALYSIS

The box diagrams of HV indicator values obtained by NSGA-II, SMS-EMOA, RM-MEDA, TMOEA/D, and CoDE-MOPSO, respectively, over 33 independent runs on the model of gear reducer are shown in Figure 6 (on the left is the original photo, on the right is a partially enlarged photo). It can be seen that CoDE-MOPSO obtains the largest median HV index value and the smallest quartile range. In this way, it can be proved that in solving gear reducer optimally designed model CoDE-MOPSO shows a better diversity and convergence. The revolutionary curves of mean HV value with NSGA-I, SMS-EMOA, RM-MEDA, TMOEA/D, and CoDE-MOPSO respectively, over 33 independent runs on the model of gear reducer are plotted in Figure 7 (the left is the original photo, and the right is a partially enlarged photo). It can be observed that CoDE-MOPSO obtains the highest HV index value in the smallest number of revolutionary generations, that is, CoDE-MOPSO gains the fastest convergence speed and maintains population diversity in the revolutionary procedure. Ranking these five algorithms in order of solving the gear reducer optimal is CoDE-MOPSO, NSGA-II, SMS-EMOA, RM-MEDA, and TMOEA/D. Table 8 shows the two solutions of the optimization model of gear reducer when



(a) (b)

FIGURE 7. Evolution of mean IGD indicator values. (a) Evolution of HV; (b) HV partial enlarged photo.

f_{volume} is minimum and when f_{volume} is minimum, respectively, and a trade-off solution.

VI. CONCLUSION

A hybrid composite differential evolution and multi-objective particle swarm optimization evolutionary algorithm (CoDE-MOPSO) based on cluster technique is proposed in this paper. At first, K-means cluster technique, from part class or global class with a certain possibility chooses the parent population to generate new solution. Then, according to the utility of different reproduction mechanisms in the past generations, the control probability β is adjusted adaptively in each generation. Finally, β decides which algorithm will be selected to generate new solution, CoDE or MOPSO.

In order to verify the performance of CoDE-MOPSO algorithm, we choose other four typical multi-objective evolutionary algorithms to make a comparison in solving GLT test and WFG test. The results show that the proposed algorithm exhibits fast convergence speed and good diversity. The effectiveness of cluster operator and composite differential evolution operator is demonstrated by the analysis experiments. Flexibility analysis of control parameter indicates CoDE-MOPSO algorithm is sensitive to some parameter settings.

CoDE-MOPSO algorithm and other four multi-objective evolutionary algorithms are also used to optimize the designed model of a two-grader straight tooth column gear reducer. The analysis of HV box plot and the evolutionary curve of mean HV over 33 independent runs indicate the proposed CoDE-MOPSO algorithm has an advantage in this problem. Accordingly, it deserves to be applied in engineering.

Despite demonstrating promising performance across various benchmark functions, we recognize certain limitations inherent to our approach. Specifically, the performance of CoDE-MOPSO is sensitive to parameter settings, such as differential weights, crossover probabilities, and parameters related to the particle swarm's social and cognitive behaviors, which can significantly influence the algorithm's convergence speed and solution quality. Additionally, the algorithm may encounter challenges in efficiently handling high-dimensional optimization problems due to the increased complexity of searching within high-dimensional spaces. Moreover, computational efficiency becomes a concern for particularly complex or large-scale problems, limiting the algorithm's applicability in real-time or resource-constrained scenarios.

To address these limitations and pave the way for future research, we propose several technical solutions. First, exploring adaptive parameter adjustment mechanisms could reduce the impact of parameter selection, dynamically tuning parameters based on feedback during the algorithm's execution. Integrating techniques for high-dimensional optimization, such as sparse representation and dimensionality reduction, could enhance the algorithm's ability to navigate high-dimensional spaces more effectively. Leveraging parallel computing frameworks could accelerate the algorithm's execution, particularly for large-scale problems, and investigating more efficient data structures and algorithm implementations could further improve computational efficiency. Additionally, combining theoretical research with empirical studies to deepen the understanding of the algorithm's convergence and stability, as well as exploring its applicability across diverse fields such as bioinformatics, sustainable energy management, and smart manufacturing, could reveal new challenges and directions for algorithm improvement. Through these technical advancements and a broader application scope, we aim to continuously refine and innovate within the field of multi-objective optimization, contributing robust tools for solving complex optimization challenges.

REFERENCES

[1] H. Han, W. Lu, and J. Qiao, "An adaptive multiobjective particle swarm optimization based on multiple adaptive methods," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2754–2767, Sep. 2017.

[2] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 32–49, Mar. 2011.

[3] C. A. Coello Coello and M. Reyes-Sierra, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *Int. J. Comput. Intell. Res.*, vol. 2, no. 3, pp. 287–308, 2006.
[4] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.
[5] Y. Cui, X. Meng, and J. Qiao, "A multi-objective particle swarm optimization algorithm based on two-archive mechanism," *Appl. Soft Comput.*, vol. 119, Apr. 2022, Art. no. 108532.
[6] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.
[7] Y. Hou, Y. Wu, and H. Han, "Multistate-constrained multiobjective differential evolution algorithm with variable neighborhood strategy," *IEEE Trans. Cybern.*, vol. 53, no. 7, pp. 1–14, 2022.
[8] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, pp. 341–359, Dec. 1997.
[9] L. Li, L. Chang, T. Gu, W. Sheng, and W. Wang, "On the norm of dominant difference for many-objective particle swarm optimization," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 2055–2067, Apr. 2021.
[10] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, "Differential evolution: A recent review based on state-of-the-art works," *Alexandria Eng. J.*, vol. 61, no. 5, pp. 3831–3872, May 2022.
[11] W. Zhang, C. Li, M. Gen, W. Yang, Z. Zhang, and G. Zhang, "Multiobjective particle swarm optimization with direction search and differential evolution for distributed flow-shop scheduling problem," *Math. Biosciences Eng.*, vol. 19, no. 9, pp. 8833–8865, 2022.
[12] B. Wu, W. Hu, J. Hu, and G. G. Yen, "Adaptive multiobjective particle swarm optimization based on evolutionary state estimation," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3738–3751, Jul. 2021.
[13] X. Zhang, X. Zhang, S. L. Ho, and W. N. Fu, "Designing loudspeaker by ensemble of composite differential evolution ingredients," *IEEE Trans. Magn.*, vol. 50, no. 11, pp. 1–4, Nov. 2014.
[14] X. Yang and G. Liu, "Self-adaptive clustering-based differential evolution with new composite trial vector generation strategies," in *Proc. 2nd Int. Congr. Comput. Appl. Comput. Sci.*, vol. 1. Berlin, Germany: Springer, 2012, pp. 261–267.
[15] X. Li, Y. Song, J. Gao, B. Zhang, L. Gui, W. Yuan, Z. Li, and S. Han, "Multi-objective optimization method for reactor shielding design based on SMS-EMOA," *Ann. Nucl. Energy*, vol. 194, Dec. 2023, Art. no. 110097.
[16] M. Xiao, L. Chen, H. Feng, Z. Peng, and Q. Long, "Smart city public transportation route planning based on multi-objective optimization: A review," *Arch. Comput. Methods Eng.*, pp. 1–25, Mar. 2024.
[17] Y. Li, Z. Xie, S. Yang, and Z. Ren, "A hybrid algorithm based on NSGA-II and MOPSO for multi-objective designs of electromagnetic devices," *IEEE Trans. Magn.*, vol. 59, no. 5, pp. 1–4, May 2023.
[18] M. H. Hassan, F. Daqaq, A. Selim, J. L. Domínguez-García, and S. Kamel, "MOIMPA: Multi-objective improved marine predators algorithm for solving multi-objective optimization problems," *Soft Comput.*, vol. 27, no. 21, pp. 15719–15740, Nov. 2023.
[19] J. M. Abdullah, T. A. Rashid, B. B. Maaroof, and S. Mirjalili, "Multi-objective fitness-dependent optimizer algorithm," *Neural Comput. Appl.*, vol. 35, no. 16, pp. 11969–11987, Jun. 2023.
[20] Y. Hu, Y. Zhang, and D. Gong, "Multiobjective particle swarm optimization for feature selection with fuzzy cost," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 874–888, Feb. 2021.
[21] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proc. IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intelligence)*, Jun. 2008, pp. 2419–2426.
[22] O. Schutze, A. Lara, and C. A. C. Coello, "On the influence of the number of objectives on the hardness of a multiobjective optimization problem," *IEEE Trans. Evol. Comput.*, vol. 15, no. 4, pp. 444–455, Aug. 2011.
[23] K. Deb and H. Jain, "Handling many-objective problems using an improved NSGA-II procedure," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.
[24] X. Zheng and H. Liu, "A hybrid vertical mutation and self-adaptation based MOPSO," *Comput. Math. Appl.*, vol. 57, nos. 11–12, pp. 2030–2038, Jun. 2009.
[25] W. Zhao, Z. Luan, and C. Wang, "Parameter optimization design of vehicle E-HHPS system based on an improved MOPSO algorithm," *Adv. Eng. Softw.*, vol. 123, pp. 51–61, Sep. 2018.
[26] A. Britto and A. Pozo, "I-MOPSO: A suitable PSO algorithm for many-objective optimization," in *Proc. Brazilian Symp. Neural Netw.*, Oct. 2012, pp. 166–171.
[27] A. Britto and A. Pozo, "Using reference points to update the archive of MOPSO algorithms in many-objective optimization," *Neurocomputing*, vol. 127, pp. 78–87, Mar. 2014.
[28] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007.
[29] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, Sep. 2007.
[30] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.
[31] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, 1999.
[32] H.-L. Liu, F.-q. Gu, and Y.-m. Cheung, "T-MOEA/D: MOEA/D with objective transform in multi-objective problems," in *Proc. Int. Conf. Inf. Sci. Manage. Eng.*, vol. 2, Aug. 2010, pp. 282–285.
[33] M. Wang, J. Guo, Y. Wang, M. Yu, and J. Guo, "Multimodal autism spectrum disorder diagnosis method based on DeepGCN," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 31, pp. 3664–3674, 2023.
[34] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, Guanajuato, Mexico: Springer, 2005, pp. 280–295.
[35] M. Wang, Z. Ma, Y. Wang, J. Liu, and J. Guo, "A multi-view convolutional neural network method combining attention mechanism for diagnosing autism spectrum disorder," *PLoS ONE*, vol. 18, no. 12, Dec. 2023, Art. no. e0295621.
[36] A. Farhang-Mehr and S. Azarm, "Entropy-based multi-objective genetic algorithm for design optimization," *Structural Multidisciplinary Optim.*, vol. 24, no. 5, pp. 351–361, Nov. 2002.
[37] S. Azarm, A. Tits, and M. Fan, "Tradeoff-driven optimization-based design of mechanical systems," in *Proc. 4th Symp. Multidisciplinary Anal. Optim.*, Sep. 1992, p. 4758.

JIN SHANG was born in Harbin, Heilongjiang, China, in 1974. He received the Ph.D. degree in control engineering from Harbin University of Science and Technology, in 2012. From 2014 to 2023, he was an Associate Professor with Wuxi Institute of Technology. His research interests include intelligent optimal and the study of multi-objective optimization algorithms.

GUIYING LI was born in Jingle, Shanxi, China, in 1974. She received the Ph.D. degree from Northeast Forestry University, China. She is currently with the School of Mechanical and Electrical Engineering, Heilongjiang University. Her research interests include intelligent optimal and the study of multi-objective optimization algorithms.