Integrating Conjugate Gradients into Evolutionary Algorithms for Large-Scale Continuous Multi-Objective Optimization

Ye Tian, Haowen Chen, Haiping Ma, Xingyi Zhang, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, and Yaochu Jin, *Fellow, IEEE*

***Abstract*—Large-scale multi-objective optimization problems (LSMOPs) pose challenges to existing optimizers since a set of well-converged and diverse solutions should be found in huge search spaces. While evolutionary algorithms are good at solving small-scale multi-objective optimization problems, they are criticized for the low efficiency in converging to the optimums of LSMOPs. By contrast, mathematical programming methods offer fast convergence speed on large-scale single-objective optimization problems, but they have difficulties in finding diverse solutions for LSMOPs. Currently, how to integrate evolutionary algorithms with mathematical programming methods for solving LSMOPs remains unexplored. In this paper, a hybrid algorithm is tailored for LSMOPs by coupling differential evolution and a conjugate gradient method. On the one hand, the conjugate gradients and differential evolution are used to update different decision variables of a set of solutions, where the former drives the solutions to quickly converge towards the Pareto front and the latter promotes the diversity of the solutions to cover the whole Pareto front. On the other hand, the objective decomposition strategy of evolutionary multi-objective optimization is used to differentiate the conjugate gradients of solutions, and the line search strategy of mathematical programming is used to ensure the higher quality of each offspring than its parent. In comparison with state-of-the-art evolutionary algorithms, mathematical programming methods, and hybrid algorithms, the proposed algorithm exhibits better convergence and diversity performance on a variety of benchmark and real-world LSMOPs.**

***Index Terms*—Large-scale multi-objective optimization, evolutionary computation, mathematical programming, conjugate gradient, differential evolution.**

Manuscript received –. This work was supported in part by the National Key R&D Program of China under Grant 2018AAA0100100, in part by the National Natural Science Foundation of China under Grant 61906001, Grant 62136008, and Grant U21A20512, in part by the Key Program of Natural Science Project of Educational Commission of Anhui Province under Grant KJ2020A0036, and in part by an Alexander von Humboldt Professorship for Artificial Intelligence funded by the Federal Ministry of Education and Research, Germany. *(Corresponding authors: Xingyi Zhang.)*

Y. Tian, H. Chen, and H. Ma are with the Information Materials and Intelligent Sensing Laboratory of Anhui Province, Institutes of Physical Science and Information Technology, Anhui University, and also with the Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei 230601, China (email: field910921@gmail.com; chen15931998216@163.com; hpma@ahu.edu.cn).

X. Zhang is with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Artificial Intelligence, Anhui University, Hefei 230601, China (email: xyzhanghust@gmail.com).

K. C. Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR (email: kctan@polyu.edu.hk).

Y. Jin is with the Faculty of Technology, Bielefeld University, 33619 Bielefeld, Germany (email: yaochu.jin@uni-bielefeld.de).

I. Introduction

Many optimization problems in scientific research and engineering applications have multiple objectives and a large number of decision variables, which are known as large-scale multi-objective optimization problems (LSMOPs) [1]. For example, time-varying ratio error estimation aims to find the true phase voltages by optimizing two objectives and four constraints [2], and fluence map optimization aims to find the optimal beam weights for intensity-modulated radiotherapy by optimizing several objectives [3], [4]. Such LSMOPs contain hundreds or thousands of decision variables, making them challenging to be solved by conventional optimizers. The difficulties mainly lie in the need for a set of well-converged and diverse solutions rather than a single one, disabling conventional optimizers from striking a balance between convergence and diversity [5].

More recently, a number of multi-objective evolutionary algorithms (MOEAs) have been tailored for solving LSMOPs, which tackle the huge search spaces by decision variable grouping, search space reduction, or novel search strategies [6]. The decision variable grouping based MOEAs divide the large number of decision variables into multiple groups and alternately optimize each group of variables, hence the optimal solutions can be approximated in a divide-and-conquer manner [1]. The search space reduction based MOEAs directly reduce the number of decision variables, which is achieved by problem transformation or dimensionality reduction strategies [7], [8]. The novel search strategy based MOEAs generate well-converged solutions by suggesting new search strategies, including variation operators and probability models [9], [10]. These MOEAs can obtain a set of diverse solutions in a single run, but the approximation of global optimal solutions still requires a large number of function evaluations, though they are much faster than conventional MOEAs.

Mathematical programming methods, on the other hand, can quickly converge to a single optimum with the assistance of function information. For solving nonlinear

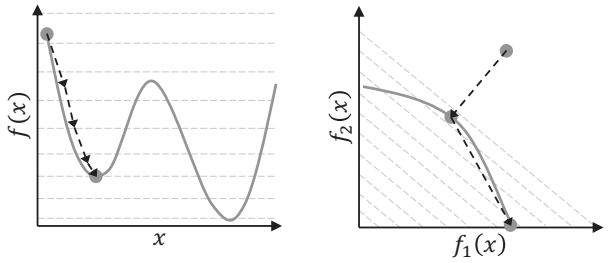


Fig. 1. A problem with nonconvex landscape (left) and a problem with nonconvex Pareto front (right).

and unconstrained optimization problems, the first-order methods (e.g., steepest descent [11] and Adam [12]) and the second-order methods (e.g., Newton's method [13] and the BFGS method [14]) can be adopted. Moreover, there exist a variety of methods tailored for specific types of problems, such as the simplex method for linear programming [15], the Lagrange multiplier method for quadratic programming [16], the Levenberg-Marquardt algorithm for least squares problems [17], the sequential quadratic programming for constrained optimization [18], and the operator splitting method for conic optimization [19]. However, most mathematical programming methods can only generate a single solution at a time, which are inefficient to generate a set of diverse solutions. More seriously, they are difficult to handle nonconvex problems, which refers to two different cases as illustrated in Fig. 1. On the one hand, for the problems with nonconvex landscapes, these methods are vulnerable to getting trapped in local optimums due to the use of gradients [20]. On the other hand, for the problems with nonconvex Pareto fronts, they can only find extreme solutions, which always have the minimum objective values regardless of the weight vector for the aggregation of multiple objectives [21].

As a consequence, evolutionary computation and mathematical programming are developed by two huge but almost completely disjoint communities, each with its own strengths and weaknesses [22]. MOEAs are suitable for solving small-scale multi-objective optimization problems owing to the population based evolutionary framework, while mathematical programming methods are effective for solving large-scale single-objective optimization problems with the assistance of gradients. By contrast, MOEAs exhibit slow convergence speed on large-scale optimization problems since they search in a black-box manner, and mathematical programming methods hold poor diversity performance on multi-objective optimization problems due to the single-point search paradigm. Thus, the integration of MOEAs and mathematical programming methods turns out to be a key issue in solving continuous LSMOPs. However, little attention has been paid to this direction and existing hybrid algorithms just attach gradient based local search to MOEAs [23], [24], whereas the deep integration of the two types of optimizers is rare. To bridge the gap between evolutionary computation and mathematical programming for better solving LSMOPs, this work proposes a new algorithm by combing multi-objective differential evolution and a conjugate gradient method, which is expected to drive a set of solutions to quickly converge towards different regions of the Pareto front. This paper contains the following contributions:

1) Based on a review of existing MOEAs, mathematical programming methods, and hybrid algorithms, this paper analyzes their limitations in solving LSMOPs. Also, an important but overlooked issue is discussed, i.e., how to obtain the gradients of continuous LSMOPs.
2) To better solve LSMOPs, this paper proposes a hybrid method for generating solutions. Firstly, it generates solutions based on conjugate gradients to quickly approximate the Pareto optimal solutions. Secondly, it uses differential evolution to tune the solutions for enhancing the population diversity. Thirdly, it adopts the objective decomposition strategy used in MOEAs to assist the conjugate gradient based search, and adopts the line search strategy used in mathematical programming to assist the differential evolution. By inheriting the good convergence performance of conjugate gradient methods and the good diversity performance of differential evolution, the population is expected to have good convergence and diversity at last.
3) Moreover, this paper proposes a multi-objective conjugate gradient and differential evolution (MOCGDE) algorithm. The proposed algorithm evolves a small population by the proposed hybrid method, and maintains a large archive to collect a set of well-converged and diverse solutions as the output. According to the experimental results on three benchmark suites and two real-world applications with up to 10 000 decision variables, the proposed algorithm shows significantly better convergence and diversity performance than 11 state-of-the-art MOEAs, mathematical programming methods, and hybrid algorithms.

The remainder of this paper is structured as follows. Section II reviews existing work and discusses one relevant issue, Section III details the procedure of the proposed algorithm, Section IV analyzes the experimental results, and Section V concludes this paper.

II. Background and Related Work

This work focuses on continuous optimization problems with or without constraints, which can be mathematically defined as

$$\begin{aligned} &\text{Minimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})) \\ &\text{Subject to } \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ &\qquad h_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, p \end{aligned}, \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_D)$ denotes a decision vector containing D real variables, $\mathbf{f}(\mathbf{x})$ denotes an objective vector

containing M objective functions, $\mathbf{l}$ denotes the lower bound of decision vectors, $\mathbf{u}$ denotes the upper bound of decision vectors, and $h_i(\mathbf{x})$ denotes p inequality constraints. Note that the objective and constraint functions may be given explicitly or implicitly, which means that the gradients can be calculated directly or estimated by some strategies. The problem defined in (1) is called an LSMOP if the number of objectives $M \geq 2$ and the number of decision variables $D \geq 100$ [9], which leads to the poor convergence performance of conventional MOEAs and the poor diversity performance of conventional mathematical programming methods.

A. Evolutionary Algorithms for Solving LSMOPs

Although conventional MOEAs can tackle multiple objectives by using dominance based [25], decomposition based [21], or indicator based [26] selection strategies, they are inefficient to find well-converged solutions in the huge search spaces of LSMOPs. To address this dilemma, three categories of MOEAs have been developed to solve LSMOPs, but they still encounter difficulties in balancing between efficiency and effectiveness.

The first category of MOEAs divides the decision variables into multiple groups via different strategies, which mainly include random grouping, differential grouping, and variable analysis. The random grouping is very efficient since it randomly divides the decision variables into a predefined number of groups with equal size [1], but it may drive the solutions towards local optimums since the interactions between variables are totally ignored. The differential grouping is more effective for approximating global optimal solutions due to the detection of interactions between variables [27], but it holds quite a high computational complexity since the interaction between each two variables should be detected independently. The variable analysis serves as a supplement of differential grouping [28], which can distinguish between the variables contributing to convergence and diversity but is still time-consuming.

The second category reduces the number of decision variables via problem transformation or dimensionality reduction strategies. Instead of optimizing the decision variables, the problem transformation strategies explore the huge search space by optimizing a weight vector acted on existing solutions, and the number of optimized variables can be highly reduced since the weight vector is much shorter than decision vectors [7], [29]. But this is at the expense of effectiveness, where local optimal solutions can be efficiently found while global optimal solutions are likely to be lost in the weight vector space [30]. By contrast, the dimensionality reduction strategies aim to reduce the search space via machine learning [31], data mining [32], or other techniques [33], which consider the variable interactions and can retain the global optimal solutions in the reduced search space. However, the dimensionality reduction strategies are only suitable for specific LSMOPs like those with low intrinsic dimensions [31] or sparse optimal solutions [24].

The third category customizes new variation operators or probability models to efficiently approximate the global optimal solutions in the original search space. The customized variation operators are the enhancement of existing ones such as genetic operators [34], competitive swarm optimizer [35], and covariance matrix adaptation evolution strategy [36]. The customized probability models generate new solutions by characterizing promising solutions found so far, which is achieved by Gaussian process [10], generative adversarial networks [37], mirror partition based solving knowledge [38], and other techniques [39]. These search strategies can improve the convergence speed of existing MOEAs on LSMOPs, but a large number of function evaluations are still required to reach the global Pareto front.

B. Mathematical Programming Methods for Nonlinear and Unconstrained Optimization

On the contrary, mathematical programming methods converge much faster since the search direction is guided by gradients. In terms of nonlinear and unconstrained optimization, the mathematical programming methods can be divided into three categories. The first category updates solutions according to first-order gradients, such as steepest descent [11] and many momentum assisted methods like Nesterov's method [40], RMSProp [41], and Adam [12]. These methods have very low computational complexities, which are efficient for solving the problems with many decision variables, e.g., the training of deep neural networks [20]. The second category updates solutions according to second-order gradients, such as Newton's method [13], the quasi-Newton method [42], and the BFGS method [14]. These methods have much faster convergence speeds than the first-order methods, but they are efficient for only small-scale problems since the calculation of the inverse of the Hessian matrix or its approximation is extremely time-consuming. The third category refers to the conjugate gradient methods [43], which update solutions according to the linear combination of the current and historical gradients. The conjugate gradient methods only need to calculate the first-order gradients, and its convergence speed is between those of the first- and second-order methods, holding a good balance between efficiency and effectiveness for solving large-scale problems. Therefore, this work adopts a popular conjugate gradient method in the proposed algorithm, namely, the FRCG method [44].

Nevertheless, the above mathematical programming methods are customized for single-objective optimization, where a single gradient has to be obtained to update each solution. To handle multi-objective optimization problems, the weighted sum method aggregates multiple objectives by a predefined weight vector [45], Wierzbicki's method aggregates multiple objectives by a predefined reference point [46], and the ϵ-constraint method optimizes a single objective and regards the others as constraints [47]. To eliminate the requirement of

decision makers' preference, the multi-objective steepest descent method [48], [49] and multi-objective Newton's method [50] were suggested, which can converge to a solution that satisfies certain first-order necessary conditions of Pareto optimality without the aggregation of multiple objectives. While the above methods can only generate one solution at a time, a gradient based multi-objective optimization method incorporates a population based aggregative strategy [51], and it is further enhanced with a distance constraint technique to handle nonconvex Pareto fronts [52]. However, the obtained populations have poor diversity since the method does not adopt effective diversity preservation strategies.

C. Evolutionary Algorithms Assisted by Gradients

To obtain a set of diverse solutions in a single run, some hybrid algorithms have been suggested to integrate evolutionary algorithms with gradient based local search strategies. These hybrid algorithms can be grouped into two categories, according to whether the true gradients or pseudo-gradients are used. The first category generates solutions by variation operators and true gradients, for instance, the Gaussian mutation and RProp work cooperatively to generate solutions in [23], the solutions obtained by particle swarm optimization are updated by the quasi Newton-Raphson algorithm in [53], and the solutions generated by a sparse multi-objective evolutionary algorithm are tuned by stochastic gradient descent in [24]. To inherit both the exploration ability of genetic operators and the exploitation ability of gradient descent for deep neural network training, a gradient based simulated binary crossover operator is developed in [20], where the population can escape from local optimums and quickly converge to the global Pareto front. The second category generates solutions by variation operators and pseudo-gradients, for example, a gradient based local search is attached to the genetic algorithm in [54] and the gradient information is involved in the variation operator of particle swarm optimization in [55], both of which estimates the gradients by using finite difference. In [56], the evolutionary gradient search estimates the gradient through random mutations on the current solution, and this method is further extended to multi-objective optimization in [57].

In spite of the assistance of gradient information, the hybrid algorithms for multi-objective optimization have not yet received much attention in the community, which is mainly due to the following two reasons. Firstly, the variation operators are simply tailed by gradient based local search in existing work, which accelerates the convergence speed but is harmful to the diversity preservation in objective spaces. Secondly, most existing work focuses on optimizing a single objective, while the aggregation of the gradients of multiple objectives is rarely touched. As a consequence, the performance of evolutionary algorithms can be enhanced on large-scale single-objective optimization problems, but such problems have already been solved by mathematical programming methods. By contrast, mathematical programming methods are ineffective for LSMOPs, which also pose challenges to existing hybrid algorithms.

Therefore, this work proposes an effective hybrid algorithm for solving LMOPs by deeply integrating evolutionary computation with mathematical programming. Before the description of the proposed algorithm, the way to calculate the gradients of continuous LSMOPs is discussed in the following.

D. Calculation of Gradients

To use mathematical programming methods for solving LSMOPs, the first issue is to calculate the gradients of given solutions. For the continuous objective functions whose first partial derivatives are available, the Jacobian matrix can be directly calculated by

$$J_{\mathbf{f}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_D} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{x})}{\partial x_1} & \frac{\partial f_M(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_M(\mathbf{x})}{\partial x_D} \end{bmatrix}. \tag{2}$$

By defining a weight vector $\mathbf{w} = (w_1, \ldots, w_M)$, the gradient $\mathbf{g}$ can be obtained to update a solution $\mathbf{x}$, i.e., $\mathbf{g} = \mathbf{w}J_{\mathbf{f}}(\mathbf{x})$.

While for the objective functions whose first partial derivatives are unavailable, the gradients can still be approximated according to the definition of first partial derivative. For example, some work approximates the gradient of a solution $\mathbf{x}$ by [56], [57]

$$f_i'(\mathbf{x}) \approx \frac{f_i(\mathbf{x}+\epsilon) - f_i(\mathbf{x})}{\epsilon},\ i = 1, \ldots, M, \tag{3}$$

where ϵ is set to a tiny value. It should be noted that, however, such an approximation is questionable, since the first partial derivatives $\frac{\partial f_i(\mathbf{x})}{\partial x_j}$ are not obtained and the Jacobian matrix is still unknown. By calculating a scalar $f_i'(\mathbf{x})$ rather than a vector $\left(\frac{\partial f_1(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial f_1(\mathbf{x})}{\partial x_D}\right)$, all the decision variables can only be updated with the same direction and the same step size, which is obviously ineffective in reaching the global optimums.

On the contrary, performing forward finite difference [58] on each variable is more promising to approximate the Jacobian matrix:

$$\frac{\partial f_i(\mathbf{x})}{\partial x_j} \approx \frac{f_i(\mathbf{x}_{j+\epsilon}) - f_i(\mathbf{x})}{\epsilon},\ i = 1, \ldots, M,\ j = 1, \ldots, D, \tag{4}$$

where $\mathbf{x}_{j+\epsilon} = (x_1, \ldots, x_{j-1}, x_j + \epsilon, x_{j+1}, \ldots, x_D)$. This way, the first partial derivatives of objective functions can be properly approximated by setting ϵ to a tiny value. Considering the range of the j-th dimension of the search space, this work sets ϵ to $10^{-6}(u_j - l_j)$ in the experiments, where u_j and l_j denote the upper and lower bounds of the j-th decision variable, respectively.

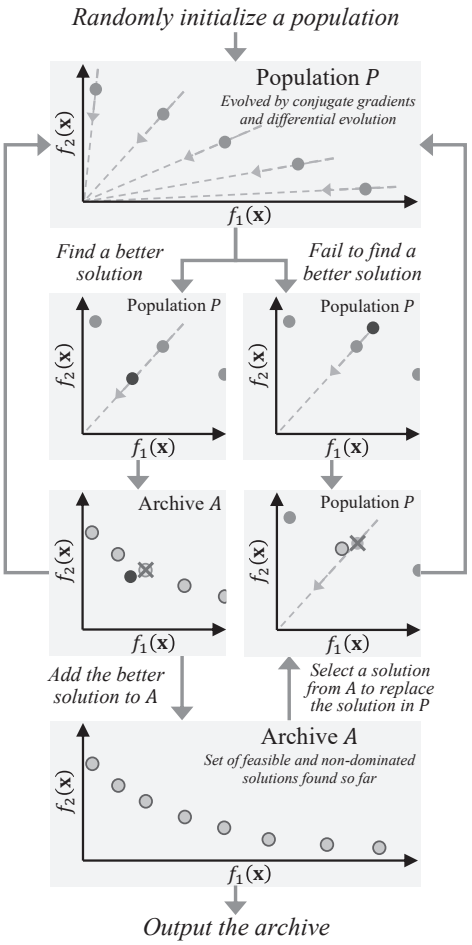


Fig. 2. General procedure of the proposed MOCGDE.

To consider both black- and white-box scenarios in the experiments, we regard the benchmark LSMOPs as black-box problems and approximate the Jacobian matrix by (4), and regard the real-world LSMOPs as white-box problems and calculate the Jacobian matrix by hand. In short, the proposed algorithm is a metaheuristic since it can solve black-box problems by approximating the gradients, and is also a heuristic since it can solve white-box problems by using the partial derivatives directly. By contrast, the proposed algorithm is different from matheuristics [59], since the proposed algorithm is a problem-independent method based on mathematical programming methods and metaheuristics for continuous optimization, while matheuristics are problem-dependent methods mostly based on mathematical programming models and ad hoc heuristics for combinatorial optimization.

III. The Proposed Algorithm

A. Procedure of the Proposed MOCGDE

The general procedure of the proposed multi-objective conjugate gradient and differential evolution (MOCGDE) algorithm is illustrated in Fig. 2. As can be observed, the proposed algorithm maintains two solution sets during the optimization process, including a population P evolved by conjugate gradients and differential evolution, and an archive A storing the feasible and non-dominated solutions found so far. The population has a small size to save computational resources, since each solution in the population is updated by conjugate gradients and differential evolution independently. By contrast, the archive has a large size to store sufficient promising solutions as the output, and also provide alternatives to replace prematurely converged solutions in the population.

Algorithm 1: Procedure of MOCGDE

Input: D (number of decision variables), N_P (size of population), N_A (size of archive)
Output: A (final archive)

1 $P \leftarrow$ Generate N_P random solutions;
2 $W \leftarrow$ Generate N_P uniformly distributed weight vectors;
3 $A \leftarrow UpdateArchive(P, N_A)$;
4 $K \leftarrow 1 \times D$ vector of zeros; `//Counters for restart`
5 $G \leftarrow \emptyset$; `//Set of gradients`
6 $S \leftarrow \emptyset$; `//Set of search directions`
7 **while** *termination criterion is not fulfilled* **do**
8 **for** $i = 1, \ldots, |P|$ **do**
9 $[\mathbf{g}, diff] \leftarrow CalGradient(P[i], W[i])$;
10 **if** $\mathrm{mod}(K[i], D) == 0$ **then**
11 $\mathbf{s} \leftarrow -\mathbf{g}$;
12 **else**
13 $\mathbf{s} \leftarrow -\mathbf{g} + \frac{\mathbf{g}\mathbf{g}^T}{G[i]G[i]^T} S[i]$;
14 $K[i] \leftarrow K[i] + 1$;
15 $G[i] \leftarrow \mathbf{g}$;
16 $S[i] \leftarrow \mathbf{s}$;
17 $[\mathbf{x}, success] \leftarrow GenSolution(P[i], A, \mathbf{s}, diff)$;
18 **if** $success$ **then**
19 $P[i] \leftarrow \mathbf{x}$;
20 $A \leftarrow UpdateArchive(A \cup \{\mathbf{x}\}, N_A)$;
21 **else**
22 $P[i] \leftarrow$ Randomly select a solution from A;
23 $K[i] \leftarrow 0$;
24 **return** A;

As described in Algorithm 1, the proposed algorithm starts with the random generation of a population P (Line 1), then the same number of uniformly distributed weight vectors are generated by Das and Dennis's method [60] to differentiate the convergence directions of the solutions in P (Line 2). Of course, the weight vectors can also be generated by other sampling methods such as the Taguchi method [61] and uniform design [60]. Besides, an archive A and the other variables are also initialized (Lines 3–6). In each iteration, each solution in the population is updated by conjugate gradients and differential evolution in turn, and the newly generated solution is used to update the population and the archive. More specifically, the gradient of a solution is first calculated (Line 9), and the next search direction $\mathbf{s}$ is calculated (Lines 10–13) according to the following formula suggested in FRCG [44]:

$$\mathbf{s} = \begin{cases} -\mathbf{g}, & \text{if } \mathrm{mod}(k, D) = 0 \\ -\mathbf{g} + \frac{\mathbf{g}\mathbf{g}^T}{\mathbf{g}_0\mathbf{g}_0^T}\mathbf{s}_0, & \text{otherwise} \end{cases}, \tag{5}$$

Algorithm 2: $UpdateArchive(A, N_A)$

Input: A (archive), N_A (size of archive)
Output: A (updated archive)
1 $A \leftarrow$ Remove the dominated solutions from A;
2 **while** $|A| > N_A$ **do**
3 $\quad [\mathbf{x}, \mathbf{y}] \leftarrow \text{argmin}_{\mathbf{x},\mathbf{y} \in A,\ \mathbf{x} \neq \mathbf{y}} \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\|$;
4 $\quad dis_{\mathbf{x}} \leftarrow \min_{\mathbf{z} \in A \setminus \{\mathbf{x},\mathbf{y}\}} \|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{z})\|$;
5 $\quad dis_{\mathbf{y}} \leftarrow \min_{\mathbf{z} \in A \setminus \{\mathbf{x},\mathbf{y}\}} \|\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{z})\|$;
6 $\quad$ **if** $dis_{\mathbf{x}} < dis_{\mathbf{y}}$ **then**
7 $\quad\quad A \leftarrow A \setminus \{\mathbf{x}\}$;
8 $\quad$ **else**
9 $\quad\quad A \leftarrow A \setminus \{\mathbf{y}\}$;
10 **return** A;

Algorithm 3: $CalGradient(\mathbf{x}, \mathbf{w})$

Input: $\mathbf{x}$ (current solution), $\mathbf{w}$ (weight vector)
Output: $\mathbf{g}$ (gradient), $diff$ (variables with different signs of partial derivatives on all objectives)
1 **if** $\exists i \in \{1, \ldots, p\} : h_i(\mathbf{x}) > 0$ **then**
2 $\quad J_{\mathbf{h}}(\mathbf{x}) \leftarrow$ Calculate the Jacobian matrix of constraints directly or by finite difference;
3 $\quad \mathbf{w} \leftarrow 1 \times |\mathbf{w}|$ vector of ones;
4 $\quad \mathbf{g} \leftarrow \mathbf{w} J_{\mathbf{h}}(\mathbf{x})$;
5 $\quad diff \leftarrow \emptyset$;
6 **else**
7 $\quad J_{\mathbf{f}}(\mathbf{x}) \leftarrow$ Calculate the Jacobian matrix of objectives directly or by finite difference;
8 $\quad \mathbf{g} \leftarrow \mathbf{w} J_{\mathbf{f}}(\mathbf{x})$;
9 $\quad diff \leftarrow$ Set of variables with different signs of partial derivatives in $J_{\mathbf{f}}(\mathbf{x})$;
10 **return** $\mathbf{g}, diff$;

where $\mathbf{g}$ denotes the gradient of the solution to be updated. $\mathbf{g}_0$ and $\mathbf{s}_0$ denote the recent gradient and search direction of the solution, respectively, which are stored in the sets G and S. While the solution may spirally converge to the optimum under the guidance of gradients, the search direction is periodically reverted to the negative of the gradient to ensure the conjugacy [44]. This is achieved by a counter k, where $\mathbf{s}$ is set to $-\mathbf{g}$ if k is a multiple of the number of decision variables D, and k is increased by 1 at the end of each iteration.

Afterwards, the obtained search direction together with differential evolution is used to update the solution (Line 17). If a better solution is found, the solution is used to replace its ancestor in the population (Line 19) and update the archive (Line 20). While if no better solution is found, a solution randomly selected from the archive is used to replace the ancestor in the population (Line 22) and restart the conjugate gradient iterations (Line 23). The procedure of updating the archive is detailed in Algorithm 2, where all the dominated solutions are first removed from the archive (Line 1), and then the solution having the shortest Euclidean distance to the others is removed one by one (Lines 3–9), until the archive does not exceed its maximum size (Line 2).

In the next subsections, the two core components of the proposed algorithm are elaborated, including the calculation of gradients and the generation of solutions.

B. Calculating Gradients in the Proposed MOCGDE

Algorithm 3 presents the procedure of calculating gradients in the proposed MOCGDE. Since the algorithm is tailored for both unconstrained and constrained optimization, the Jacobian matrix of the constraints is calculated if the current solution is infeasible (i.e., at least one constraint function value is larger than zero) (Line 2), and the Jacobian matrix of the objectives is calculated if the current solution is feasible (Line 7). The Jacobian matrix can be calculated directly if the partial derivatives are available, and has to be estimated by finite difference if the partial derivatives are unavailable.

It is worth noting that the weight vectors for aggregating the constraints and the objectives are different. Since all the constraints must be satisfied, the weight vector for aggregating the constraints is set to a vector of ones (Lines 3–4), which equates the weights of all the constraints. By contrast, since the solutions should make diverse trade-offs between the objectives, the weight vector for aggregating the objectives is predefined (Line 8), where each solution is associated with a distinct weight vector as well as convergence direction. In short, the use of gradients aims to enhance the population convergence, while the uniformly distributed weight vectors are responsible for preserving the population diversity. Besides, since the first partial derivatives of a decision variable may have different signs on all the objectives, which means that the update of this decision variable will decrease some objectives but increase the others, such decision variables are stored in the variable $diff$ (Line 9) and will not be updated to ensure the solution can converge along the predefined direction.

C. Generating Solutions in the Proposed MOCGDE

Algorithm 4 presents the procedure of generating solutions in the proposed MOCGDE. To determine a suitable step size, a simple line search strategy is used to exponentially reduce the step size m from 1 until a better solution is found. To be specific, a new solution $\mathbf{y}$ is obtained by updating an existing solution $\mathbf{x}$ according to the following formula

$$y_i = x_i + (1 - d_i) \times 0.5^m \times s_i + d_i \times 0.5^m \times (x'_i - x''_i), \quad (6)$$

where $\mathbf{s}$ is the search direction used in FRCG, $\mathbf{x}', \mathbf{x}''$ are two solutions randomly selected from the archive used in differential evolution, $d_i = 1$ indicates that the i-th variable is in $diff$ and $d_i = 0$ otherwise. Since $diff$ stores the decision variables with different signs of first partial derivatives, the decision variables out of $diff$ are updated by conjugate gradients to enhance the population convergence (Line 7), while the decision variables in $diff$ are tuned by differential evolution to enhance the population diversity (Line 9).

After a new solution is obtained, the line search terminates if the new solution is better than the original one

Algorithm 4: $GenSolution(\mathbf{x}, A, \mathbf{s}, diff)$

Input: $\mathbf{x}$ (current solution), A (archive), $\mathbf{s}$ (search direction), $diff$ (variables with different signs of partial derivatives on all objectives)
Output: $\mathbf{y}$ (new solution), $success$ (whether a better solution is found)

```
1  success ← false;
2  for m = 0, ..., 9 do
3      y ← 1 × |x| vector of zeros;
4      [x', x''] ← Randomly select two solutions from A;
5      for i = 1, ..., |x| do
6          if i ∉ diff then
7              y_i ← x_i + 0.5^m × s_i;
8          else
9              y_i ← x_i + 0.5^m × (x'_i − x''_i);
10     CV_x ← Σ_{i=1}^{p} max{h_i(x), 0};
11     CV_y ← Σ_{i=1}^{p} max{h_i(y), 0};
12     if CV_y < CV_x || CV_y == CV_x && f(y) ≺ f(x) then
13         success ← true;
14         break;
15 return y, success;
```

(Lines 12–14). Formally, solution $\mathbf{y}$ is better than solution $\mathbf{x}$ if $\mathbf{y}$ has a smaller constraint violation than $\mathbf{x}$:

$$\sum_{i=1}^{p} \max\{h_i(\mathbf{y}), 0\} < \sum_{i=1}^{p} \max\{h_i(\mathbf{x}), 0\}, \tag{7}$$

or they have the same constraint violation and $\mathbf{y}$ dominates $\mathbf{x}$:

$$\begin{cases} \forall i \in \{1, \ldots, D\} : f_i(\mathbf{y}) \leq f_i(\mathbf{x}) \\ \exists i \in \{1, \ldots, D\} : f_i(\mathbf{y}) < f_i(\mathbf{x}) \end{cases}. \tag{8}$$

D. Computational Complexity of the Proposed MOCGDE

According to Algorithm 1, the computational complexity of MOCGDE is mainly determined by four operations at each iteration, including the calculation of gradients, the calculation of search direction, the update of solution, and the update of archive. For the calculation of gradients, the time complexity is $O(MD)$ for calculating the Jacobian matrix through (2). For the calculation of search direction, the time complexity is $O(D)$ according to (5). For the update of solution, the time complexity is $O(D)$ according to (6). For the update of archive, the time complexity is $O(MN_A^2)$ for calculating the distances between each two solutions in the objective space. As a result, the computational complexity of updating one solution in MOCGDE is $O(M(D+N_A^2))$, when an archive with size N_A is used to solve a problem having M objectives and D variables.

E. Discussions

To summarize, the proposed MOCGDE suggests a deep combination of mathematical programming and evolutionary computation. More specifically, the conjugate gradient method and differential evolution are used to update solutions collaboratively, the uniform weight vectors suggested in decomposition based MOEAs are used to differentiate the search directions used in the conjugate gradient method, and the line search strategy suggested in mathematical programming is used to ensure that the population evolved by evolutionary algorithms can always generate better offspring. In comparison with existing hybrid algorithms, the proposed algorithm has the following advantages:

- *Framework*: Most existing algorithms evolve a single population, where a small population cannot generate many diverse solutions and a large population wastes many computational resources. By contrast, the proposed algorithm drives a small population towards the global optimums, and maintains a large archive to store sufficient diverse solutions.
- *Gradients*: Most existing algorithms consider the gradients of a single objective, which do not explicitly define weight vectors to aggregate multiple objectives and differentiate the search directions of multiple solutions. By contrast, the proposed algorithm uses objective decomposition strategy to associate each solution with a distinct weight vector, and thus the solutions can spread along the whole Pareto front. In addition, the proposed algorithm also considers the gradients of constraints when needed, which is suitable for both unconstrained and constrained optimization.
- *Operators*: Most existing algorithms directly attach gradient based local search to the variation operators of evolutionary algorithms, where the population diversity may deteriorate due to the difference between the signs of first partial derivatives on all the objectives. By contrast, the proposed algorithm only uses conjugate gradients to update the decision variables having the same sign of first partial derivatives on all the objectives, and thus each solution can converge along its predefined direction. Moreover, the proposed algorithm uses differential evolution to tune the remaining decision variables, which can further enhance the population diversity.

As a consequence, the proposed MOCGDE inherits both the fast convergence speed of mathematical programming methods and the good diversity performance of evolutionary algorithms, and is also very efficient since it holds the same computational complexity as conjugate gradient methods and differential evolution. In the next section, the performance of the proposed MOCGDE is verified by comparing it with state-of-the-art algorithms on benchmark and real-world LSMOPs.

IV. Experimental Studies

A. Comparative Algorithms

Eleven representative algorithms are involved in the experiments, including three MOEAs, three mathematical programming methods, and five hybrid algorithms. The three MOEAs are NSGA-II [25], LSMOF [29], and

LMOCSO [9], where NSGA-II is one of the most classical multi-objective evolutionary algorithms, LSMOF is a problem transformation based large-scale MOEA, and LMOCSO is a new variation operator based large-scale MOEA. The three mathematical programming methods include FRCG [44], Izui's algorithm [51], and MOSD [49], where FRCG is a popular conjugate gradient method, Izui's algorithm is a population based multi-objective linear programming method, and MOSD is a multi-objective steepest descent algorithm. The five hybrid algorithms are NSGA-II+gradient, NSGA-II+gSBX [20], MO-EGS [57], GPSO [53], and SAPSO [55], where NSGA-II+gradient denotes that each solution generated by NSGA-II is tuned by a line search based steepest descent, NSGA-II+gSBX denotes the NSGA-II with gradient based simulated binary crossover, MO-EGS is a multi-objective evolutionary gradient search algorithm, GPSO is a combination of particle swarm optimization and the quasi Newton-Raphson algorithm, and SAPSO integrates gradient information into the variation operator of particle swarm optimization.

Table I lists the detailed settings of all the compared algorithms, which mostly follow those suggested in their original papers. Since the calculations of objectives and Jacobian matrices are with different computational complexities, the CPU time rather than the number of function evaluations is adopted as the termination criterion for fairness. It should be noted that FRCG, MOSD, MO-EGS, GPSO, and SAPSO can only generate a single solution at a time, hence they are executed many times with the assistance of a set of uniformly distributed weight vectors [60] for the aggregation of multiple objectives, and the total CPU time is also divided into the same number of stages. Besides, since most of the compared algorithms are not tailored for constrained optimization, these algorithms use the same constraint handling strategies as the proposed MOCGDE, i.e., the constraints are given higher priority than the objectives when calculating gradients and comparing solutions.

TABLE I
DETAILED SETTINGS OF THE COMPARED ALGORITHMS

General Settings	
Population or archive size	50 and 45 for 2- and 3-objective problems
Weight vectors	Used in FRCG, MOSD, MO-EGS, GPSO, and SAPSO
	The same size to the population
	Generated by Das and Dennis's method [60]
Termination criterion (CPU time)	40s and 900s for 1 000- and 10 000-variable ZDT and DTLZ problems
	100s and 3 000s for 1 000- and 10 000-variable IMF problems
	100s, 300s, and 500s for 1 200-, 6 000-, and 9 900-variable TREE problems
	700s for FMO problems
Variation Operators	
Simulated binary crossover [62]	Used in NSGA-II, LSMOF, NSGA-II+gradient, and NSGA-II+gSBX
	Crossover probability: 1
	Distribution index: 20
Polynomial mutation [63]	Used in NSGA-II, LSMOF, NSGA-II+gradient, and NSGA-II+gSBX
	Mutation probability: $1/D$
	Distribution index: 20
Competitive swarm optimizer [9]	Used in LMOCSO
Particle swarm optimization [64]	Used in GPSO and SAPSO
	Inertia weight: 0.4
Algorithm-Specific Settings	
NSGA-II [25]	—
LSMOF [29]	Generation of weight optimization: 10
	Population size of transferred problem: 30
LMOCSO [9]	—
FRCG [44]	Parameter β in backtracking line search: 0.6
	Parameter σ in backtracking line search: 0.4
Izui's algorithm [51]	—
MOSD [49]	Step size: 0.1
NSGA-II+gradient	—
NSGA-II+gSBX [20]	—
MO-EGS [57]	Number of test candidates: 500
GPSO [53]	Population size of single run: 20
	Accuracy of local search: 10^{-3}
	Step size of local search: 1
	Number of iterations of local search: 1 000
SAPSO [55]	Population size of single run: 20
	Parameter c_1 in variation operator: 2
	Parameter c_2 in variation operator: 10^{-2}
	Lower bound of diversity: 10^{-6}
	Upper bound of diversity: 0.25
	Number of local evaluations: 3
MOCGDE	Small population size: 10

B. Test Problems

Three multi-objective test suites are adopted to test the performance of the compared algorithms, including ZDT [65], DTLZ [66], and IMF [10], which contain 22 scalable benchmark problems in total. These problems contain two or three objectives without constraint, and the number of decision variables is set to 1 000 and 10 000. Two real-world problems are also adopted, including time-varying ratio error estimation (TREE) [2] and fluence map optimization (FMO) [3], [4]. The TREE problems contain two objectives with three or four constraints, aiming to minimize the total ratio error and ratio error variation and satisfy the constraints of topology, series, and phase. The FMO problems are very important in intensity-modulated radiotherapy [3], which contain two objectives without constraint and aim to satisfy the requirements of maximum and minimum doses. The number of decision variables of the real-world problems varies from 1 000 to 9 900, depending on the five datasets involved in the TREE problems and the three datasets involved in the FMO problems.

In terms of mathematical programming methods, the Jacobian matrix is approximated by finite difference on each benchmark problem and calculated directly on each real-world problem. The inverted generational distance (IGD) [67] and hypervolume (HV) [68] are adopted as the performance indicators. For the populations obtained on benchmark problems, approximately 10 000 reference points are sampled by the methods suggested in [60] for IGD calculation. For the populations obtained on real-world problems, the solutions in all the populations for the same test instance are collected, and a reference point is set to the maximum objective values of the solutions on all dimensions for HV calculation. The experiments are performed for 30 independent runs, and the mean

TABLE II
IGD VALUES OBTAINED BY TWELVE ALGORITHMS ON ZDT AND DTLZ TEST SUITES

Problem (M, D)	NSGA-II	LSMOF	LMOCSO	FRCG	Izui's algorithm	MOSD	NSGA-II+ gradient	NSGA-II+ gSBX	MO-EGS	GPSO	SAPSO	MOCGDE
ZDT1	5.6637e-1	9.9424e-3	2.1318e+0	3.1898e-1	1.2327e+0	2.6338e+0	2.0851e+0	2.0556e+0	2.2685e+0	2.3511e-1	2.5302e+0	7.5508e-3
(2,1000)	(8.74e-2) −	(4.19e-4) −	(2.01e-1) −	(9.89e-2) −	(1.84e-1) −	(2.03e-2) −	(1.18e-1) −	(1.08e-1) −	(1.02e-1) −	(9.80e-2) −	(5.57e-2) −	(8.04e-5)
ZDT1	2.0267e+0	1.0069e-2	2.7842e+0	2.6646e+0	1.1731e+0	2.9045e+0	2.7995e+0	2.6146e+0	2.7047e+0	2.1754e+0	2.8482e+0	8.0435e-3
(2,10000)	(3.85e-2) −	(3.78e-4) −	(2.32e-2) −	(4.60e-1) −	(1.38e-1) −	(2.97e-2) −	(1.41e-2) −	(9.34e-2) −	(1.52e-2) −	(9.88e-1) −	(1.43e-2) −	(5.27e-4)
ZDT2	8.1034e-1	9.8913e-3	3.1488e+0	1.1360e+0	2.2737e+0	4.0901e+0	3.5233e+0	3.3725e+0	3.9070e+0	4.5658e-1	4.2654e+0	7.6381e-3
(2,1000)	(8.38e-2) −	(4.60e-4) −	(1.48e-1) −	(1.16e-1) −	(2.85e-1) −	(5.00e-2) −	(1.47e-1) −	(1.19e-1) −	(5.87e-2) −	(1.32e-1) −	(1.13e-1) −	(7.67e-5)
ZDT2	3.6064e+0	1.0476e-2	3.7016e+0	4.4804e+0	2.0827e+0	4.6193e+0	4.6103e+0	4.2438e+0	4.4373e+0	6.0949e-1	4.6395e+0	7.7498e-3
(2,10000)	(5.30e-2) −	(7.57e-4) −	(2.27e-1) −	(1.03e-1) −	(8.76e-2) −	(1.85e-2) −	(1.52e-2) −	(9.90e-2) −	(1.08e-2) −	(1.17e-16) −	(1.53e-2) −	(9.62e-5)
ZDT3	3.6440e-1	1.8567e-1	1.6639e+0	9.2358e-2	9.8576e-1	2.2105e+0	2.3023e+0	1.6430e+0	1.9113e+0	1.7496e+0	2.4705e+0	3.7836e-2
(2,1000)	(1.03e-1) −	(9.70e-2) −	(1.24e-1) −	(3.00e-2) −	(1.25e-1) −	(1.33e-1) −	(6.01e-2) −	(6.15e-2) −	(9.73e-2) −	(2.48e-1) −	(2.17e-1) −	(8.56e-2)
ZDT3	1.4953e+0	4.0317e-1	1.8707e+0	3.9005e-1	7.4045e-1	2.3483e+0	2.7768e+0	2.0834e+0	2.1625e+0	2.3067e+0	2.4369e+0	2.8915e-2
(2,10000)	(2.58e-2) −	(1.61e-1) −	(1.53e-1) −	(2.08e-1) −	(1.89e-2) −	(3.62e-2) −	(1.87e-1) −	(8.80e-2) −	(2.03e-2) −	(1.29e-2) −	(1.34e-1) −	(3.42e-2)
ZDT4	3.3308e+3	6.6082e+0	2.0813e+4	8.2175e+3	2.4818e+4	1.5727e+4	7.1683e+3	1.1975e+4	1.6412e+4	1.2755e+4	1.1222e+3	4.6546e+3
(2,1000)	(8.20e+2) ≈	(1.08e+1) +	(1.66e+3) −	(2.42e+2) −	(3.83e-12) −	(1.62e+3) −	(3.81e+3) ≈	(4.50e+2) −	(2.19e+2) −	(9.10e+2) −	(6.22e+1) +	(3.67e+3)
ZDT4	1.0087e+5	5.1586e+1	1.8482e+5	8.2703e+4	2.0777e+5	1.7282e+5	9.4186e+4	1.2327e+5	1.7454e+5	1.3836e+5	1.3476e+4	6.4416e+4
(2,10000)	(1.49e+4) −	(1.31e+2) +	(5.91e+4) −	(4.62e+2) −	(1.13e+5) −	(1.27e+3) −	(2.62e+4) −	(2.60e+3) −	(4.37e+2) −	(6.70e+3) −	(2.56e+2) +	(2.74e+4)
ZDT6	4.8456e+0	5.3309e-1	7.2895e+0	6.4301e-1	6.6209e+0	7.6365e+0	7.5490e+0	7.3556e+0	7.0886e+0	6.0954e-1	7.6660e+0	2.3628e-1
(2,1000)	(5.15e-1) −	(9.90e-2) −	(1.44e-1) −	(3.88e-2) −	(1.97e-1) −	(2.91e-2) −	(2.22e-2) −	(4.26e-2) −	(7.17e-2) −	(8.94e-2) −	(2.04e-2) −	(1.14e-1)
ZDT6	7.4259e+0	9.4675e-1	7.5330e+0	3.5280e+0	6.4634e+0	7.8740e+0	7.8629e+0	7.7257e+0	7.6858e+0	5.4056e-1	7.8820e+0	2.4182e-1
(2,10000)	(3.65e-2) −	(2.50e-1) −	(1.12e-1) −	(3.32e+0) −	(7.23e-2) −	(9.59e-3) −	(1.05e-2) −	(5.00e-2) −	(1.38e-2) −	(1.76e-1) −	(6.25e-3) −	(1.64e-1)
DTLZ1	3.7988e+3	5.6073e+0	7.3815e+3	4.9581e+3	2.0264e+4	9.2265e+3	1.2205e+4	1.2652e+4	8.8768e+3	9.9829e+3	3.1072e+4	1.3559e+3
(2,1000)	(3.94e+2) −	(5.43e+0) +	(1.70e+3) −	(2.07e+2) −	(7.59e+2) −	(3.41e+3) −	(8.82e+2) −	(1.72e+3) −	(3.02e+1) −	(5.26e+3) −	(1.55e+3) −	(6.68e+2)
DTLZ1	1.9020e+5	1.0742e+2	6.7214e+4	4.9490e+4	1.9548e+5	2.1457e+5	1.2493e+5	1.1991e+5	8.9079e+4	1.2487e+5	3.0633e+5	6.5565e+4
(2,10000)	(3.51e+4) −	(1.19e+2) +	(2.10e+4) ≈	(1.41e+3) +	(1.25e+4) −	(2.32e+4) −	(1.77e+1) −	(5.61e+3) −	(3.98e+2) −	(7.26e+0) −	(2.19e+4) −	(2.40e+4)
DTLZ2	2.1762e-1	1.0435e-2	4.0718e+0	4.7916e-1	1.2753e+1	4.9523e+1	2.2528e-1	3.2679e+1	7.6976e+1	3.4242e-1	2.6925e+1	9.8765e-3
(2,1000)	(1.99e-2) −	(2.95e-4) ≈	(4.73e-1) −	(1.84e-1) −	(5.63e+0) −	(4.62e+0) −	(1.20e-1) −	(4.78e+0) −	(1.43e+0) −	(2.01e-14) −	(8.84e+0) −	(9.45e-4)
DTLZ2	4.4721e+2	2.7091e-2	5.0478e+1	5.4392e+1	1.9489e+2	8.0547e+2	3.1680e-1	6.1874e+2	8.1454e+2	6.0668e-1	6.8788e+2	2.4951e-2
(2,10000)	(1.19e+1) −	(2.35e-2) ≈	(5.67e+0) −	(6.57e+1) −	(9.14e+0) −	(2.97e+0) −	(5.99e-2) −	(3.33e+1) −	(2.42e+0) −	(1.91e-1) −	(1.93e+1) −	(9.40e-3)
DTLZ3	9.9840e+3	1.0921e+1	2.0966e+4	1.8348e+4	4.9136e+4	2.9055e+4	2.4962e+4	3.5092e+4	2.4911e+4	1.9962e+4	8.3494e+4	3.3015e+3
(2,1000)	(1.03e+3) −	(1.30e+1) +	(5.71e+3) −	(1.35e+4) −	(7.06e+3) −	(1.11e+4) −	(2.69e+0) −	(4.16e+3) −	(1.08e+1) −	(1.05e+4) −	(6.40e+3) −	(1.24e+3)
DTLZ3	5.1175e+5	3.2402e+2	1.6028e+5	1.6179e+5	5.3206e+5	6.1801e+5	2.4984e+5	3.2938e+5	2.4986e+5	2.4975e+5	8.3735e+5	5.4209e+4
(2,10000)	(1.08e+5) −	(4.23e+2) +	(4.92e+4) −	(8.83e+4) −	(4.77e+4) −	(7.90e+4) −	(1.44e+1) −	(1.43e+4) −	(1.87e+1) −	(1.22e+2) −	(9.38e+4) −	(3.47e+4)
DTLZ4	7.8412e-1	5.2321e-1	1.2267e+1	7.0328e-1	3.1764e+1	6.0628e+1	5.0228e-1	4.3319e+1	7.7259e+1	7.4209e-1	7.4176e+1	4.2235e-1
(2,1000)	(2.63e-1) −	(3.54e-1) ≈	(6.98e+0) −	(1.23e-1) −	(1.83e+1) −	(1.91e+1) −	(2.06e-1) ≈	(1.36e+1) −	(1.10e+0) −	(8.97e-7) −	(9.59e+0) −	(1.69e-1)
DTLZ4	4.8683e+2	6.0058e-1	2.7314e+2	7.4209e-1	1.8237e+1	7.1392e+2	4.6209e-1	3.5793e+2	8.0913e+2	7.0956e-1	5.6554e+2	3.4898e-1
(2,10000)	(6.96e+1) −	(3.11e-1) −	(6.74e+1) −	(1.27e-11) −	(3.72e+1) −	(4.76e+1) −	(1.93e-1) ≈	(5.86e+1) −	(4.94e+0) −	(1.03e-1) −	(1.71e+2) −	(1.73e-1)
DTLZ5	3.1729e-1	1.0878e-2	4.3023e+0	5.8920e-1	1.1419e+1	4.9382e+1	8.0330e-2	3.1904e+1	7.5208e+1	3.4242e-1	3.0954e+1	1.0215e-2
(2,1000)	(1.02e-1) −	(3.81e-4) ≈	(4.94e-1) −	(1.78e-1) −	(4.98e+0) −	(3.48e+0) −	(9.70e-2) −	(4.21e+0) −	(1.93e+0) −	(7.85e-13) −	(1.46e+1) −	(1.04e-3)
DTLZ5	4.4325e+2	1.2747e-2	5.5592e+1	3.4543e+1	1.9391e+2	7.8374e+2	3.4090e-1	6.1941e+2	8.1774e+2	4.6010e-1	6.8848e+2	1.1000e-2
(2,10000)	(1.89e+1) −	(3.46e-3) ≈	(7.42e+0) −	(7.24e+1) −	(8.90e+0) −	(8.50e+0) −	(4.80e-3) −	(3.37e+1) −	(2.83e+0) −	(1.41e-1) −	(2.73e+1) −	(1.11e-3)
DTLZ6	6.3535e+2	1.1391e+2	3.1397e+2	1.0000e+0	8.3147e+2	8.9463e+2	4.1233e+1	7.4530e+2	4.3209e+2	7.0921e+1	5.7244e+2	4.1444e-1
(2,1000)	(7.98e+0) −	(9.82e+0) −	(3.10e+1) −	(1.17e-16) −	(1.91e+0) −	(9.83e-1) −	(2.62e+1) −	(6.27e+1) −	(4.11e+0) −	(6.11e+1) −	(1.84e+1) −	(1.19e-4)
DTLZ6	8.5178e+3	3.3109e+3	3.9434e+3	7.0774e+3	8.8872e+3	9.0474e+3	6.8601e+3	8.9293e+3	4.7925e+3	5.7574e+3	8.2072e+3	4.1443e-1
(2,10000)	(2.33e+1) −	(3.08e+2) −	(6.27e+2) −	(3.73e+3) −	(4.86e+0) −	(6.37e+0) −	(4.97e+2) −	(5.14e+1) −	(1.21e+1) −	(5.00e+2) −	(2.70e+1) −	(2.40e-4)
DTLZ7	1.0665e+0	4.0072e-1	5.2279e+0	2.6633e-1	3.3289e+0	7.7334e+0	4.4660e+0	5.5639e+0	6.1115e+0	3.1983e-1	7.6237e+0	9.4681e-3
(2,1000)	(1.13e-1) −	(1.37e-1) −	(5.27e-1) −	(1.18e-1) −	(6.05e-1) −	(1.17e-1) −	(4.94e-1) −	(3.83e-1) −	(8.17e-2) −	(1.16e-1) −	(3.11e-1) −	(2.79e-4)
DTLZ7	5.7861e+0	4.0174e-1	6.4588e+0	5.3949e-1	3.1057e+0	7.9608e+0	7.8697e+0	7.1688e+0	7.3912e+0	6.8268e-1	7.9696e+0	1.0086e-2
(2,10000)	(8.86e-2) −	(1.37e-1) −	(3.53e-1) −	(1.96e-1) −	(1.64e-1) −	(3.76e-2) −	(9.40e-2) −	(2.05e-1) −	(7.62e-2) −	(1.69e-1) −	(4.10e-2) −	(9.22e-4)
$+/-/\approx$	0/23/1	6/13/5	0/23/1	1/23/0	0/24/0	0/24/0	0/21/3	0/24/0	0/24/0	0/24/0	2/22/0	

and standard deviation of the indicator values are reported. Moreover, the Friedman test with Bonferroni correction at a significance level of 0.05 is used for statistical test, where '+', '−', and '≈' mean that the performance of an algorithm is statistically better, worse, and similar to the proposed MOCGDE, respectively. All the experiments are conducted on PlatEMO [69].

C. Comparisons on Benchmark Problems

Table II presents the IGD values obtained by the compared algorithms on ZDT and DTLZ test suites. Generally, the proposed MOCGDE gains the best results on 18 out of 24 test instances, which is followed by LSMOF gaining the best results on the remaining 6 test instances. Therefore, for general continuous LSMOPs, the proposed MOCGDE shows superiority over existing optimizers based on evolutionary computation or/and mathematical programming. Furthermore, for the IMF problems with complex linkages between variables, MOCGDE also exhibits better performance than existing algorithms, which obtains the best results on 13 out of 20 test instances as shown in Table III.

Fig. 3 depicts the populations with median IGD values obtained by the compared algorithms on 1 000-variable ZDT1. For the three evolutionary algorithms NSGA-II, LSMOF, and LMOCSO, only LSMOF can reach the Pareto front since it has a fast convergence speed due to the problem transformation strategy. For the three mathematical programming methods FRCG, Izui's algorithm, and MOSD, only FRCG can find a few optimal solutions since it is much more efficient than the first-order methods used in Izui's algorithm and MOSD. Nevertheless, FRCG exhibits poor diversity performance since it is not tailored for multi-objective optimization. For the five hybrid algorithms NSGAII-II+gradient, NSGA-II+gSBX, MO-EGS, GPSO, and SAPSO, only GPSO can find a few optimal solutions since it adopts the quasi-Newton method to update the solutions. By contrast, NSGA-

TABLE III
IGD VALUES OBTAINED BY TWELVE ALGORITHMS ON IMF TEST SUITE

Problem (M, D)	NSGA-II	LSMOF	LMOCSO	FRCG	Izui's algorithm	MOSD	NSGA-II+ gradient	NSGA-II+ gSBX	MO-EGS	GPSO	SAPSO	MOCGDE
IMF1	3.5447e-1	4.4226e-1	4.1260e+0	8.0308e-1	5.7891e-2	1.7671e+0	2.7944e-1	2.8635e+0	4.9928e+0	4.1420e-1	1.5927e+1	1.8411e-2
(2,1000)	(4.94e-2) −	(4.68e-2) −	(1.58e+0) −	(5.43e-4) −	(6.63e-2) ≈	(7.05e-2) −	(1.70e-2) −	(1.04e+0) −	(2.76e-1) −	(4.87e-2) −	(1.48e+0) −	(1.53e-3)
IMF1	6.3732e+0	5.7186e-1	8.1592e+0	1.1763e+1	4.6940e-1	1.6753e+1	2.2815e+0	4.8814e+0	1.2973e+1	7.6692e+0	1.6268e+1	3.7511e-1
(2,10000)	(3.71e-1) −	(5.20e-3) −	(1.78e-1) −	(2.87e+0) −	(4.21e-1) ≈	(1.78e-1) −	(2.41e-1) −	(1.10e+0) −	(2.47e-1) −	(3.37e-1) −	(1.12e+0) −	(3.63e-1)
IMF2	6.1574e-1	6.0948e-1	6.4985e+0	2.2893e+0	5.6221e-2	1.4042e+0	3.5764e-1	3.9689e+0	6.9860e+0	2.8189e-1	2.0702e+1	1.8403e-2
(2,1000)	(7.72e-2) −	(1.38e-6) −	(2.06e+0) −	(2.64e-1) −	(1.59e-2) −	(4.53e-2) −	(3.83e-2) −	(2.24e+0) −	(2.00e-1) −	(3.63e-2) −	(2.70e+0) −	(1.69e-3)
IMF2	9.8769e+0	6.0910e-1	1.1181e+1	1.6857e+1	4.9270e-1	2.0639e+1	4.5216e+0	8.4740e+0	1.6614e+1	7.1158e+0	1.9752e+1	3.4679e-1
(2,10000)	(3.01e-1) −	(3.61e-4) −	(2.04e-1) −	(4.17e+0) −	(4.64e-2) −	(1.63e-1) −	(2.68e+0) −	(1.49e+0) −	(1.57e-1) −	(1.77e+0) −	(1.06e+0) −	(6.89e-2)
IMF3	1.6315e-1	1.7799e-2	4.3651e+0	3.8657e-1	3.7705e-1	2.0208e+0	3.8549e-1	5.4022e+0	1.4072e+1	4.7195e-1	1.4031e+1	1.0087e-2
(2,1000)	(5.65e-2) −	(1.28e-3) −	(2.38e+0) −	(1.70e-1) −	(2.93e-1) −	(8.74e-1) −	(6.20e-2) −	(1.60e+0) −	(9.73e-1) −	(5.66e-2) −	(1.31e+0) −	(2.21e-3)
IMF3	1.0569e+1	2.4436e-2	1.0701e+1	6.5754e-1	4.8243e-1	1.9380e+1	3.9073e+0	8.6523e+0	1.9576e+1	8.4236e+0	1.7872e+1	2.5024e-1
(2,10000)	(5.79e-1) −	(1.86e-3) +	(2.07e-1) −	(1.99e-1) −	(2.18e-2) −	(6.95e-1) −	(1.66e+0) −	(1.23e+0) −	(2.19e-1) −	(2.09e+0) −	(1.36e+0) −	(1.41e-1)
IMF4	7.8493e+1	5.4152e-1	2.3872e+2	9.1074e+1	1.4422e+0	NaN	6.1227e-1	4.8113e+1	2.1679e+3	6.5888e-1	1.2685e+3	3.5777e-1
(3,1000)	(3.25e+1) −	(1.33e-4) −	(1.35e+2) −	(5.10e+1) −	(1.63e+0) −	(NaN)	(4.51e-2) −	(2.15e+1) −	(5.63e+1) −	(4.67e-2) −	(2.06e+2) −	(5.94e-4)
IMF4	1.3004e+4	5.4450e-1	1.4722e+4	8.7078e+3	4.2756e+2	NaN	1.2317e+0	4.2713e+3	2.2814e+4	6.7919e-1	1.9534e+4	3.5745e-1
(3,10000)	(3.15e+2) −	(6.47e-3) −	(1.59e+3) −	(2.24e+3) −	(1.20e+2) −	(NaN)	(4.50e-1) −	(2.44e+3) −	(4.00e+2) −	(3.78e-3) −	(1.60e+3) −	(7.97e-5)
IMF5	1.5048e-1	5.5638e-2	1.9570e-1	6.3315e-2	3.2963e+0	2.6513e-1	3.4446e-1	1.6249e-1	3.5116e-1	2.2064e-1	5.7916e-1	3.9196e-2
(2,1000)	(1.85e-2) −	(7.74e-3) −	(1.98e-3) −	(7.26e-3) −	(3.41e+0) −	(4.63e-3) −	(6.14e-2) −	(1.43e-2) −	(5.78e-3) −	(2.45e-3) −	(1.74e-1) −	(1.90e-2)
IMF5	3.3298e-1	9.8343e-2	2.1961e-1	2.5742e-1	1.8274e-1	3.4238e-1	4.5421e-1	2.4590e-1	3.7398e-1	2.6234e-1	3.8273e-1	8.3121e-2
(2,10000)	(2.32e-3) −	(3.67e-3) ≈	(8.34e-3) −	(2.37e-2) −	(3.40e-2) −	(2.56e-3) −	(1.36e-1) −	(1.43e-2) −	(1.16e-2) −	(1.71e-3) −	(3.26e-2) −	(2.02e-2)
IMF6	3.2579e-1	1.1209e-1	2.8789e-1	1.4603e-1	1.2327e-1	5.0795e-1	4.5851e-1	2.4923e-1	5.9720e-1	3.2492e-1	1.1064e+0	1.0463e-1
(2,1000)	(6.59e-2) −	(8.17e-3) ≈	(1.07e-2) −	(2.11e-2) −	(1.27e-2) −	(1.29e-2) −	(3.01e-2) −	(2.36e-2) −	(1.86e-2) −	(7.39e-3) −	(2.07e-1) −	(1.07e-2)
IMF6	6.1122e-1	1.3577e-1	3.0905e-1	3.4964e-1	2.9513e-1	6.1158e-1	6.7743e-1	3.8391e-1	6.3641e-1	4.2249e-1	6.8732e-1	2.6528e-1
(2,10000)	(5.13e-3) −	(2.22e-3) +	(1.15e-2) −	(3.98e-2) −	(8.09e-3) −	(6.78e-3) −	(7.41e-2) −	(2.45e-2) −	(4.85e-3) −	(1.86e-3) −	(6.75e-2) −	(2.11e-2)
IMF7	2.1658e-1	1.5485e-2	3.1431e-1	3.5347e-1	1.3161e-1	3.0129e-1	4.0611e-1	3.1991e-1	5.1792e-1	3.5681e-1	5.5037e-1	2.9009e-2
(2,1000)	(1.07e-1) −	(9.05e-3) +	(1.22e-2) −	(9.28e-2) −	(5.14e-3) −	(7.07e-3) −	(2.12e-2) −	(1.36e-2) −	(7.44e-3) −	(3.03e-2) −	(1.04e-2) −	(1.13e-2)
IMF7	5.1117e-1	1.4792e-1	3.2035e-1	3.4454e-1	2.7332e-1	4.9775e-1	5.2743e-1	3.6439e-1	5.4519e-1	3.8621e-1	5.5522e-1	1.3469e-1
(2,10000)	(1.05e-2) −	(1.10e-2) ≈	(4.05e-3) −	(9.07e-3) −	(1.92e-2) −	(5.25e-2) −	(1.65e-2) −	(1.19e-2) −	(2.19e-3) −	(1.61e-2) −	(1.15e-2) −	(1.61e-2)
IMF8	5.6187e+1	3.6223e-1	4.6019e+0	7.4007e-1	1.4731e+0	NaN	2.4303e+1	2.9137e+0	5.9699e+1	7.6623e-1	8.7144e+1	3.5468e-1
(3,1000)	(1.07e+1) −	(8.86e-3) −	(2.15e+0) −	(5.96e-3) −	(5.68e-1) −	(NaN)	(2.45e+1) −	(1.15e+0) −	(3.47e+0) −	(7.82e-2) −	(3.04e+1) −	(6.52e-3)
IMF8	6.7412e+2	9.4601e-1	1.5102e+2	3.2473e+2	1.1665e+2	NaN	4.9683e+2	1.5058e+2	5.7770e+2	3.8816e+1	5.6551e+2	3.5739e-1
(3,10000)	(4.30e+1) −	(7.24e-5) −	(8.98e+0) −	(1.26e+2) −	(9.82e+0) −	(NaN)	(1.19e+2) −	(5.10e+1) −	(1.00e+1) −	(1.60e+1) −	(1.10e+1) −	(1.50e-5)
IMF9	3.1390e-1	2.2899e-1	7.8730e-1	8.8753e-1	2.2473e-1	9.0725e-1	7.4729e-1	6.1660e-1	8.7559e-1	8.3822e-1	1.2968e-1	8.9074e-1
(2,1000)	(3.66e-2) +	(3.81e-2) +	(5.45e-2) +	(1.06e-2) ≈	(1.94e-1) +	(1.05e-2) ≈	(7.35e-3) +	(3.58e-2) +	(1.20e-2) +	(3.33e-2) +	(1.32e-2) +	(6.92e-3)
IMF9	2.8026e+0	5.2033e-1	4.4571e+0	5.4021e+0	1.4630e+0	5.7101e+0	5.0279e+0	4.0214e+0	4.9628e+0	4.8115e+0	2.5421e-1	3.6021e+0
(2,10000)	(1.10e-1) +	(2.00e-2) +	(1.36e-1) −	(3.24e-2) −	(2.63e-1) +	(1.01e-1) −	(6.40e-2) −	(2.16e-1) −	(9.76e-2) −	(5.41e-2) −	(4.74e-2) +	(2.52e+0)
IMF10	3.0487e+3	1.3846e+1	1.6681e+4	7.1890e+3	3.5251e+3	1.2062e+4	8.1398e+3	5.7349e+3	1.0291e+4	8.0069e+3	7.3851e+2	3.7153e+3
(2,1000)	(2.34e+2) +	(1.51e+1) +	(4.88e+3) −	(4.88e+2) −	(8.64e+1) ≈	(7.40e+1) −	(2.02e+2) −	(6.66e+2) −	(2.48e+2) −	(9.95e+2) −	(2.14e+2) +	(4.14e+2)
IMF10	1.0277e+5	6.2849e+3	1.7745e+5	8.0980e+4	4.4111e+4	1.2493e+5	8.0693e+4	8.1373e+4	1.2233e+5	9.5319e+4	7.3608e+3	4.1195e+4
(2,10000)	(1.77e+3) −	(2.45e+3) +	(2.05e+3) −	(1.60e+3) −	(2.26e+2) ≈	(2.71e+2) −	(2.37e+3) −	(4.65e+3) −	(7.21e+2) −	(7.61e+3) −	(2.76e+2) +	(1.40e+3)
$+/-/\approx$	3/17/0	7/10/3	1/19/0	0/19/1	2/14/4	0/15/1	1/19/0	1/19/0	1/19/0	1/19/0	4/16/0	

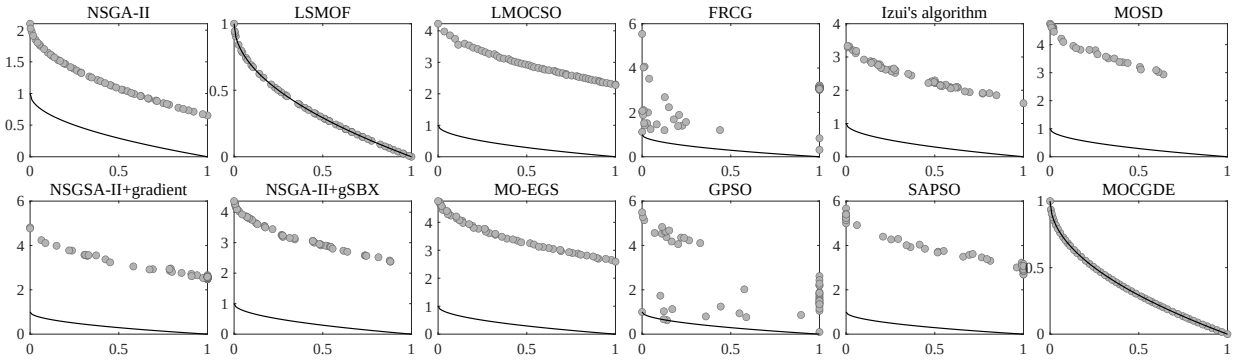


Fig. 3. Populations (in objective space) with median IGD values obtained by twelve algorithms on ZDT1 with 1 000 decision variables.

II+gradient adopts only a first-order method, which has a slow convergence speed; NSGA-II+gSBX and SAPSO integrate the first-order gradient information into the variation operators, where the solutions are not specifically optimized by mathematical programming methods; MO-EGS approximates the first partial derivatives by (3) rather than finite difference, the gradients obtained by which are inaccurate and unable to optimize the solutions effectively. On the other hand, the proposed MOCGDE can obtain a set of solutions uniformly distributed along the whole PF, holding a good balance between convergence and diversity. The superiority of MOCGDE is more significant on 1 000-variable IMF1 as shown in Fig. 4, where the problem introduces complex linkages between variables and only MOCGDE can obtain a set of well-converged and diverse solutions.

Moreover, Fig. 5 plots the IGD values obtained by the compared algorithms on ZDT1 and IMF1 with 1 000 to 10 000 decision variables, where the CPU time is proportionally set for different numbers of decision variables.

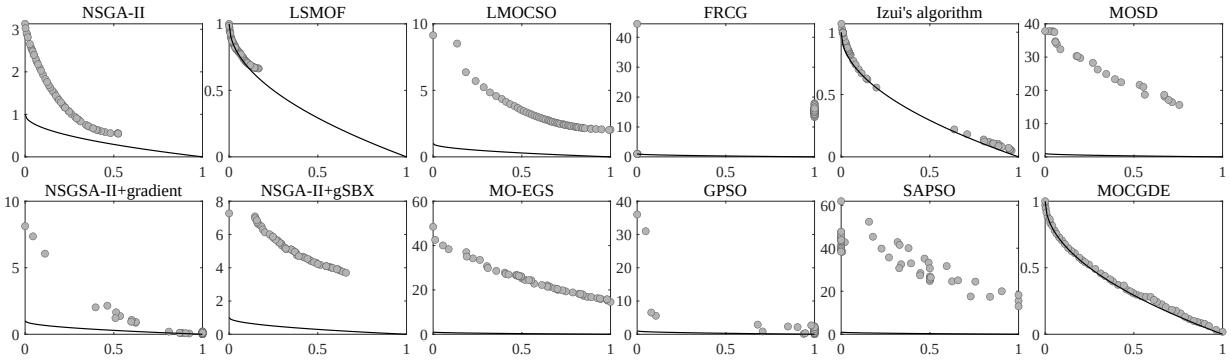


Fig. 4. Populations (in objective space) with median IGD values obtained by twelve algorithms on IMF1 with 1 000 decision variables.

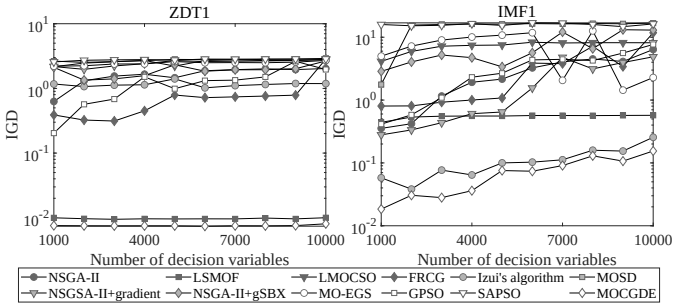


Fig. 5. IGD values obtained by twelve algorithms on ZDT1 and IMF1 with different numbers of decision variables.

It can be found that the proposed MOCGDE obtains the best IGD values on all the test instances. Besides, the IGD values of MOCGDE fluctuate slightly on ZDT1 and increase steadily on IMF1, since IMF1 contains variable linkages that are very difficult to be handled.

The superiority of the proposed MOCGDE over evolutionary algorithms is owing to the assistance of conjugate gradients, which highly improve the convergence speed in huge search spaces. According to the experimental results in Table II, MOCGDE outperforms the evolutionary algorithms NSGA-II, LSMOF, and LMOCSO on all the problems besides ZDT4, DTLZ1, and DTLZ3. In comparison with LSMOF, the inferiority of MOCGDE on ZDT4, DTLZ1, and DTLZ3 is due to their multimodal landscapes, which pose stiff challenges to existing optimizers. For example, a D-variable DTLZ1 problem contains $11^{D-1}-1$ local Pareto fronts and one global Pareto front, which is extremely difficult to be handled when the number of decision variables is large. In fact, LSMOF is still unable to find the global Pareto fronts of the three problems; instead, it can approximate a local Pareto front quickly.

The superiority of the proposed MOCGDE over mathematical programming methods is owing to the evolutionary framework optimizing multiple solutions simultaneously. In particular, the differential evolution enhances the exploration ability of gradient descent and thus reduces the probability of trapping in the local optimums of nonconvex landscapes, and the archive can preserve the population diversity on both convex and nonconvex Pareto fronts. As evidenced by Table II, for ZDT4, DTLZ1, and DTLZ3 with highly multimodal landscapes, MOCGDE significantly outperforms the mathematical programming methods FRCG, Izui's algorithm, and MOSD. For ZDT1 with a convex Pareto front and ZDT2 with a concave Pareto front, MOCGDE gains very similar IGD values on the two problems, while the IGD values of FRCG, Izui's algorithm, and MOSD on ZDT2 are much worse than those on ZDT1.

D. Comparisons on Real-World Problems

Table IV lists the HV values obtained by the compared algorithms on the TREE problems and FMO problems. It is noteworthy that the results of MOSD and MO-EGS are not reported since MOSD can only handle two objectives or constraints and MO-EGS cannot obtain any feasible solution for the TREE problems. Besides, some of the results are marked as 'NaN' since no feasible solution is obtained. According to the table, the proposed MOCGDE obtains the best HV values on 13 test instances and LSMOF obtains the best HV values on 5 test instances. In terms of statistical tests, the performance of MOCGDE is better than or similar to all the compared algorithms besides LSMOF on all the test instances. Therefore, the effectiveness of MOCGDE on applications is verified.

Fig. 6 plots the populations with median HV values obtained by the compared algorithms on TREE1 with 1 200 decision variables. It can be observed that LSMOF obtains a set of solutions with good diversity but poor convergence, GPSO and NSGA-II+gradient obtain multiple solutions with good convergence but poor diversity, and the other compared algorithms can only find a few non-dominated solutions. In particular, despite the uniformly distributed weight vectors adopted in FRCG, only a single non-dominated solution can be found by FRCG. By contrast, the proposed MOCGDE gains a set of well-converged and diverse solutions for TREE1, which is the only algorithm achieving good convergence and diversity performance. Moreover, Fig. 7 plots the populations with median HV values obtained by the compared algorithms on FMO1, where the solutions obtained by MOCGDE are with obviously higher quality than those obtained by the compared algorithms.

TABLE IV
HV VALUES OBTAINED BY TEN ALGORITHMS ON THE TREE PROBLEMS AND FMO PROBLEMS

Problem (M, D)	NSGA-II	LSMOF	LMOCSO	FRCG	Izui's algorithm	NSGA-II+ gradient	NSGA-II+ gSBX	GPSO	SAPSO	MOCGDE
TREE1	8.2109e-1	8.3052e-1	8.4344e-1	8.4499e-1	1.5787e-1	8.4917e-1	8.0982e-1	8.4850e-1	6.9572e-1	8.4945e-1
(2,1200)	(6.57e-3) −	(3.75e-4) −	(1.82e-3) −	(2.51e-3) −	(9.33e-2) −	(8.94e-5) −	(1.52e-2) −	(3.56e-4) −	(1.76e-2) −	(3.41e-4)
TREE1	6.6655e-1	8.3693e-1	7.6384e-1	8.4562e-1	1.4733e-1	8.5523e-1	6.4487e-1	8.5448e-1	6.8046e-1	8.5534e-1
(2,6000)	(2.86e-3) −	(1.22e-4) −	(4.72e-2) −	(1.10e-3) −	(5.12e-4) −	(1.23e-4) ≈	(4.27e-2) −	(6.72e-5) −	(2.47e-2) −	(1.09e-4)
TREE1	5.6229e-1	8.3156e-1	6.9670e-1	8.3717e-1	1.4656e-1	8.4891e-1	6.2225e-1	8.4922e-1	7.6437e-1	8.5096e-1
(2,9900)	(2.20e-3) −	(1.26e-4) −	(7.23e-2) −	(2.48e-3) −	(2.22e-4) −	(1.05e-3) −	(3.14e-2) −	(2.18e-3) ≈	(1.52e-2) −	(1.28e-4)
TREE2	7.7075e-1	8.5078e-1	8.3320e-1	8.5436e-1	1.8706e-1	8.5638e-1	7.4271e-1	8.0156e-1	6.8103e-1	8.5642e-1
(2,1200)	(1.30e-2) −	(1.55e-4) −	(3.92e-3) −	(5.70e-4) −	(5.95e-2) −	(9.59e-6) −	(9.20e-3) −	(7.04e-2) −	(7.68e-3) −	(4.82e-6)
TREE2	6.4800e-1	8.4993e-1	7.4939e-1	8.4619e-1	8.8094e-2	8.4875e-1	6.7533e-1	8.5631e-1	NaN	8.5659e-1
(2,6000)	(4.59e-3) −	(7.01e-5) −	(1.85e-2) −	(5.55e-3) −	(9.82e-2) −	(4.79e-3) −	(3.24e-2) −	(2.07e-4) −	(NaN)	(3.09e-5)
TREE2	NaN	8.4996e-1	NaN	8.3766e-1	2.0674e-1	8.4598e-1	6.6409e-1	8.5495e-1	7.4044e-1	8.5660e-1
(2,9900)	(NaN)	(1.17e-4) −	(NaN)	(1.19e-2) −	(4.11e-7) −	(4.54e-3) −	(3.52e-2) −	(3.87e-3) −	(1.35e-2) −	(3.02e-4)
TREE3	8.2593e-1	8.8717e-1	7.2881e-1	NaN	1.1392e-1	8.6809e-1	7.1618e-1	7.7397e-1	NaN	8.8752e-1
(2,1200)	(1.80e-3) −	(2.25e-4) −	(1.80e-2) −	(NaN)	(2.27e-2) −	(2.08e-3) −	(1.34e-2) −	(1.23e-1) −	(NaN)	(1.61e-4)
TREE3	NaN	8.8741e-1	NaN	NaN	1.3118e-1	NaN	4.9867e-1	8.7650e-1	NaN	8.8642e-1
(2,6000)	(NaN)	(1.58e-5) +	(NaN)	(NaN)	(1.36e-2) −	(NaN)	(3.77e-2) −	(5.83e-3) −	(NaN)	(7.53e-4)
TREE3	NaN	8.8825e-1	NaN	NaN	1.0960e-1	NaN	4.7658e-1	8.7282e-1	NaN	8.8640e-1
(2,9600)	(NaN)	(5.96e-6) +	(NaN)	(NaN)	(2.23e-2) −	(NaN)	(2.82e-2) −	(9.64e-3) −	(NaN)	(9.89e-4)
TREE4	7.4506e-1	9.5529e-1	7.6944e-1	NaN	1.1466e-1	8.9713e-1	3.8256e-1	8.5193e-1	NaN	9.5481e-1
(2,1200)	(5.03e-2) −	(2.51e-5) +	(1.70e-2) −	(NaN)	(2.28e-2) −	(2.95e-3) −	(2.34e-1) −	(9.70e-2) ≈	(NaN)	(1.47e-4)
TREE4	NaN	9.6426e-1	4.1560e-2	NaN	1.0658e-1	5.8767e-1	0.0000e+0	7.8639e-1	NaN	9.5625e-1
(2,6000)	(NaN)	(6.10e-5) +	(4.56e-2) −	(NaN)	(2.30e-2) −	(6.57e-3) −	(0.00e+0) −	(8.60e-2) −	(NaN)	(2.87e-3)
TREE4	NaN	9.6579e-1	0.0000e+0	NaN	1.1560e-1	5.4919e-1	0.0000e+0	5.2775e-1	NaN	9.5267e-1
(2,9600)	(NaN)	(8.24e-5) +	(0.00e+0) −	(NaN)	(2.30e-2) −	(1.71e-2) −	(0.00e+0) −	(4.14e-1) −	(NaN)	(5.44e-3)
TREE5	8.2979e-1	9.2911e-1	9.2732e-1	NaN	1.2649e-1	9.3340e-1	6.8158e-1	9.1376e-1	NaN	9.3814e-1
(2,1200)	(2.70e-2) −	(1.55e-4) −	(3.83e-3) −	(NaN)	(1.04e-2) −	(3.59e-3) −	(8.82e-2) −	(2.47e-2) −	(NaN)	(5.85e-4)
TREE5	NaN	9.2750e-1	6.4796e-1	NaN	1.1512e-1	8.6963e-1	3.6853e-1	9.0154e-1	NaN	9.3853e-1
(2,6000)	(NaN)	(4.20e-5) −	(1.04e-2) −	(NaN)	(7.01e-3) −	(5.62e-2) −	(4.34e-2) −	(2.52e-2) −	(NaN)	(5.98e-4)
TREE5	NaN	9.2751e-1	NaN	NaN	1.1381e-1	8.6203e-1	3.5158e-1	7.2272e-1	NaN	9.3806e-1
(2,9600)	(NaN)	(8.37e-5) −	(NaN)	(NaN)	(6.05e-3) −	(6.57e-2) ≈	(2.97e-2) −	(6.40e-2) −	(NaN)	(4.83e-4)
FMO1	5.4781e-1	9.9576e-1	2.3477e-1	9.7014e-1	8.2646e-1	9.9786e-1	4.0572e-1	9.7558e-1	7.6495e-2	9.9912e-1
(2,3000)	(2.49e-2) −	(1.24e-3) −	(4.21e-2) −	(3.78e-3) −	(1.89e-2) −	(2.06e-4) −	(1.38e-2) −	(3.35e-3) −	(9.86e-3) −	(8.77e-5)
FMO2	6.2618e-1	9.9122e-1	2.2844e-1	5.8281e-2	9.1302e-1	9.2494e-1	5.3965e-1	9.5477e-1	8.6886e-2	9.9458e-1
(2,3539)	(2.05e-2) −	(1.10e-3) −	(4.39e-2) −	(3.90e-3) −	(4.31e-3) −	(3.54e-3) −	(1.37e-2) −	(4.63e-3) −	(6.11e-3) −	(3.51e-4)
FMO3	7.6678e-1	9.9860e-1	4.8231e-1	2.6681e-1	6.4358e-2	9.7179e-1	5.1671e-1	9.9135e-1	2.9571e-1	9.9905e-1
(2,4625)	(2.88e-2) −	(6.04e-4) −	(2.86e-2) −	(1.93e-2) −	(5.09e-4) −	(8.96e-4) −	(2.32e-2) −	(3.78e-4) −	(1.60e-2) −	(1.39e-4)
$+/-/\approx$	0/11/0	5/13/0	0/14/0	0/9/0	0/18/0	0/14/2	0/18/0	0/16/2	0/8/0	

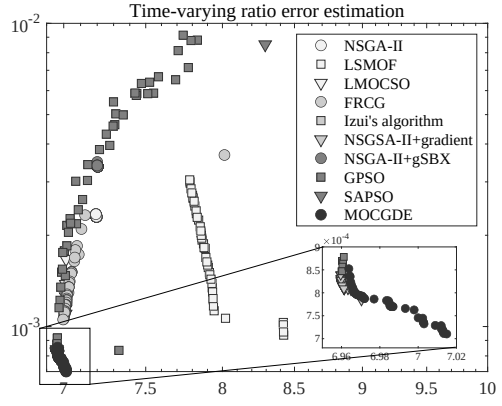


Fig. 6. Populations (in objective space) with median HV values obtained by ten algorithms on TREE1 with 1 200 decision variables.

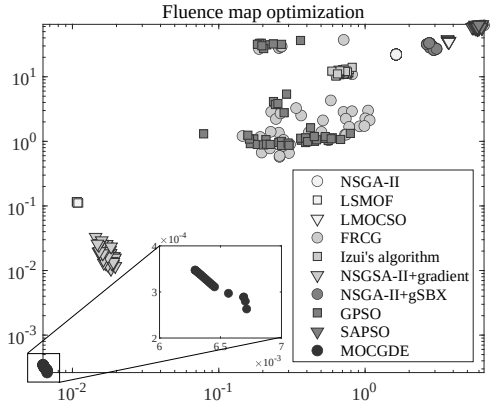


Fig. 7. Populations (in objective space) with median HV values obtained by ten algorithms on FMO1.

E. Ablation Studies on the Proposed MOCGDE

To verify the effectiveness of the core component of MOCGDE, i.e., updating solutions by both conjugate gradients and differential evolution, the proposed MOCGDE is compared with four variants on ZDT1 and IMF1, where the first variant uses conjugate gradients to update only the variables out of $diff$, the second variant uses conjugate gradients to update all the variables, the third variant uses differential evolution to update only the variables in $diff$, and the fourth variant uses

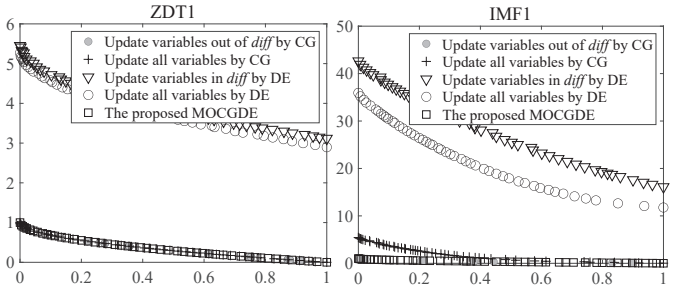


Fig. 8. Populations (in objective space) obtained by MOCGDE and its variants on ZDT1 and IMF1 with 1 000 decision variables.

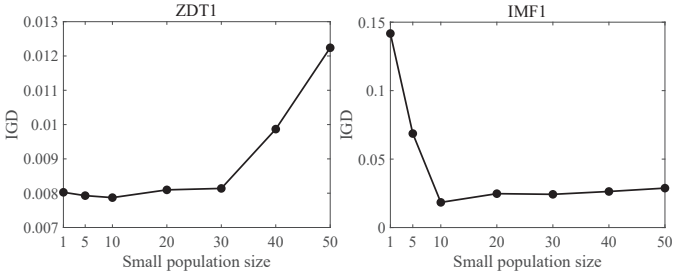


Fig. 9. Mean IGD values obtained by MOCGDE with different population sizes on ZDT1 and IMF1 with 1 000 decision variables.

differential evolution to update all the variables. It can be found from Fig. 8 that, updating only the variables out of $diff$ by conjugate gradients leads to a few well-converged solutions, since the other variables are not updated and the population diversity cannot be ensured. While updating all the variables by conjugate gradients can lead to a set of well-converged and diverse solutions on ZDT1, the solutions obtained on IMF1 have worse convergence and diversity since IMF1 is much more complex. Besides, using only differential evolution cannot obtain any well-converged solution. As a consequence, updating solutions by both conjugate gradients and differential evolution is the most effective way for the proposed MOCGDE to solve LSMOPs.

To study the influence of the population size on the performance of MOCGDE, Fig. 9 shows the IGD values of MOCGDE when the population size varies from 1 to 50. It is obvious that a small population size is effective for ZDT1 since a single optimal solution can be easily diversified to cover the whole Pareto front, while a large population size is effective for IMF1 since the spread of optimal solutions is hindered by the complex linkages between variables. As a result, the population size of 10 is the most suitable setting for MOCGDE.

V. Conclusions and Future Work

Currently, continuous LSMOPs are mostly handled by evolutionary algorithms, since they can find a set of diverse solutions in a single run [6]. However, existing evolutionary algorithms are criticized for the slow convergence speed in huge search spaces, mainly due to the neglect of gradient information. Even for the objective functions whose first partial derivatives are unavailable, the gradients can still be approximated by finite difference. Therefore, this paper has proposed a hybrid algorithm by absorbing the fast convergence speed of mathematical programming and diversity preservation strategies of evolutionary computation. The proposed algorithm uses conjugate gradients and differential evolution to drive a small population towards the Pareto front, and maintains a large archive to store a set of diverse solutions found so far. The experimental results have demonstrated that the proposed algorithm is superior over existing evolutionary algorithms, mathematical programming methods, and hybrid algorithms on a variety of benchmark and real-world LSMOPs.

This paper has revealed the effectiveness of coupling evolutionary computation with mathematical programming for large-scale multi-objective optimization. In the future, more advanced evolutionary algorithms and mathematical programming methods can be considered to solve specific types of LSMOPs. For example, using the coevolutionary constrained multi-objective optimization framework [70] with the sequential quadratic programming [18] to solve constrained LSMOPs [2], using the multi-objective competitive swarm optimizer [9] with the Levenberg-Marquardt algorithm [17] to maximize receiver operating characteristics convex hull [71], and using the sparse multi-objective evolutionary algorithm [24] with momentum assisted gradient descent methods [12] to find sparse deep neural networks [20].

References

[1] L. M. Antonio and C. A. Coello Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*, pp. 2758–2765.

[2] C. He, R. Cheng, C. Zhang, Y. Tian, Q. Chen, and X. Yao, "Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 868–881, 2020.

[3] G. Li and D. Cao, "A multi-objective particle swarm algorithm for the optimization of IMRT inverse planning," in *Proceedings of the 3rd International Conference on Biomedical Engineering and Informatics*, 2010.

[4] J. Dias, H. Rocha, and M. d. C. L. T. Ventura, B. Ferreira, "Automated fluence map optimization based on fuzzy inference systems," *Medical Physics*, vol. 43, no. 3, pp. 1083–1095, 2016.

[5] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4108–4121, 2017.

[6] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys*, vol. 54, no. 8, p. 174, 2022.

[7] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multi-objective optimization based on problem transformation," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 260–275, 2018.

[8] 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 Transactions on Cybernetics*, vol. 51, no. 6, pp. 3115–3128, 2021.

[9] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multi-objective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, 2020.

[10] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process-based inverse modeling," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 6, pp. 838–856, 2015.

[11] S. S. Petrova and A. D. Solov'ev, "The origin of the method of steepest descent," *Historia Mathematica*, vol. 24, no. 4, pp. 361–375, 1997.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[13] M. Avriel, "Nonlinear programming: Analysis and methods," *Dover Publishing*, 2003.

[14] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Mathematics of Computation*, vol. 24, pp. 647–656, 1970.

[15] K. G. Murty, *Linear programming*. John Wiley & Sons, 2000.

[16] M. Li, "Generalized Lagrange multiplier method and KKT conditions with an application to distributed optimization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 2, pp. 252–256, 2019.

[17] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[18] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1–51, 1995.

[19] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd, "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, vol. 169, pp. 1042–1068, 2016.

[20] 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 Transactions on Neural Networks and Learning Systems*, 2021.

[21] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[22] Y. D. Sergeyev, D. E. Kvasov, and M. S. Mukhametzhanov, "On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget," *Scientific Reports*, vol. 8, p. 453, 2018.

[23] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[24] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An evolutionary algorithm for large-scale sparse multi-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 380–393, 2020.

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[26] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator based multi-objective evolutionary algorithm with reference point adaptation for better versatility," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 609–622, 2018.

[27] M. Li and J. Wei, "A cooperative co-evolutionary algorithm for large-scale multi-objective optimization problems," in *Proceedings of the 2018 Annual Conference on Genetic and Evolutionary Computation Conference*. ACM, 2018, pp. 1716–1721.

[28] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, 2018.

[29] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, and X. Yao, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 949–961, 2019.

[30] Y. Feng, L. Feng, S. Kwong, and K. C. Tan, "A multi-variation multifactorial evolutionary algorithm for large-scale multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2021.

[31] H. Qian and Y. Yu, "Solving high-dimensional multi-objective optimization problems with low effective dimensions," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 875–881.

[32] Y. Tian, C. Lu, X. Zhang, F. Cheng, and Y. Jin, "A pattern mining based evolutionary algorithm for large-scale sparse multi-objective optimization problems," *IEEE Transactions on Cybernetics*, 2020.

[33] Y. Tian, Y. Feng, X. Zhang, and C. Sun, "A fast clustering based evolutionary algorithm for super-large-scale sparse multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, 2021.

[34] Y. Zhang, Y. Tian, and X. Zhang, "Improved SparseEA for sparse large-scale multi-objective optimization problems," *Complex & Intelligent Systems*, 2021.

[35] X. Wang, K. Zhang, J. Wang, and Y. Jin, "An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2021.

[36] 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," *Information Sciences*, vol. 509, pp. 457–469, 2020.

[37] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3129–3142, 2021.

[38] X. Yao, Q. Zhao, D. Gong, and S. Zhu, "Solution of large-scale many-objective optimization problems based on dimension reduction and solving knowledge guided evolutionary algorithm," *IEEE Transactions on Evolutionary Computation*, 2021.

[39] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 525–537, 2018.

[40] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k2)$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[41] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.

[42] C. G. Broyden, *Quasi-Newton Methods*. London: Academic Press, 1972, ch. Numerical Methods for Unconstrained Optimization.

[43] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, p. 409, 1952.

[44] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[45] R. T. Marler and J. S. Arora, "The weighted sum method for multi-objective optimization: new insights," *Struct Multidiscip Optim*, vol. 41, pp. 853–862, 2010.

[46] A. P. Wierzbicki, "A mathematical basis for satisficing decision making," *Mathematical Modelling*, vol. 3, no. 5, pp. 391–405, 1982.

[47] K. Miettinen, *Nonlinear multiobjective optimization*. Springer, 1999.

[48] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Mathematical Methods of Operations Research*, vol. 51, no. 3, pp. 479–494, 2000.

[49] X. Liu and A. C. Reynolds, "A multiobjective steepest descent method with applications to optimal well control," *Computational Geosciences*, vol. 20, pp. 355–374, 2016.

[50] J. Fliege, L. M. G. na Drummond, and B. F. Svaiter, "Newton's method for multiobjective optimization," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 602–626, 2009.

[51] K. Izui, T. Yamada, S. Nishiwaki, and K. Tanaka, "Multiobjective optimization using an aggregative gradient-based method," *Structural and Multidisciplinary Optimization*, vol. 51, pp. 173–182, 2015.

[52] Y. Sato, K. Izui, T. Yamada, and S. Nishiwaki, "Gradient-based multiobjective optimization using a distance constraint technique and point replacement," *Engineering Optimization*, vol. 48, no. 7, pp. 1226–1250, 2016.

[53] M. M. Noel, "A new gradient based particle swarm optimization algorithm for accurate computation of global minimum," *Applied Soft Computing*, vol. 12, pp. 353–359, 2012.

[54] K. Harada, K. Ikeda, and S. Kobayashi, "Hybridization of genetic algorithm and local search in multiobjective function optimization: recommendation of GA then LS," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 667–674.

[55] R. Santos, G. Borges, A. Santos, M. Silva, C. Sales, and J. Costa, "A semi-autonomous particle swarm optimizer based on gradient information and diversity control for global optimization," *Applied Soft Computing*, vol. 69, pp. 330–343, 2018.

[56] R. Salomon, "Evolutionary algorithms and gradient search: Similarities and differences," *IEEE Transactions on Evolutionary Computation*, vol. 2, no. 2, pp. 45–55, 1998.

[57] C. K. Goh, Y. S. Ong, K. C. Tan, and E. J. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation*, 2008.
[58] J. C. Strikwerda, *Finite Difference Schemes and Partial Differential Equations*. Wadsworth Publ. Co., 1989.
[59] R. Martí, P. M. Pardalos, and M. G. C. Resende, *Handbook of Heuristics*. Springer, 2018.
[60] Y. Tian, X. Xiang, X. Zhang, R. Cheng, and Y. Jin, "Sampling reference points on the Pareto fronts of benchmark multi-objective optimization problems," in *Proceedings of the 2018 IEEE Congress on Evolutionary Computation*. IEEE, 2018.
[61] G. Taguchi, S. Chowdhury, and S. Taguchi, *Robust engineering*. New York: McGraw-Hill, 2000.
[62] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 4, pp. 115–148, 1995.
[63] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, no. 4, pp. 30–45, 1996.
[64] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.
[65] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, pp. 173–195, 2000.
[66] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization*. Springer, 2005, pp. 105–145.
[67] C. A. C. Coello and N. C. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Programming and Evolvable Machines*, vol. 6, pp. 163–190, 2005.
[68] K. Shang, H. Ishibuchi, L. He, and L. M. Pang, "A survey on the hypervolume indicator in evolutionary multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, 2020.
[69] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.
[70] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multi-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 102–116, 2021.
[71] F. Cheng, Q. Zhang, Y. Tian, and X. Zhang, "Maximizing receiver operating characteristics convex hull via dynamic reference point-based multi-objective evolutionary algorithm," *Applied Soft Computing*, vol. 86, p. 105896, 2020.

Ye Tian received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively.

He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. His current research interests include evolutionary computation and its applications. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the 2022 IEEE Computational Intelligence Society Outstanding PhD Dissertation Award.

Haowen Chen received the B.Sc. degree from Hebei Agricultural University, Baoding, China, in 2020, where she is currently pursuing the M.Sc. degree with the Institutes of Physical Science and Information Technology at Anhui University, Hefei, China.

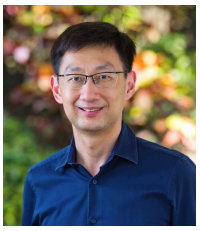
Her current research interests include evolutionary computation and mathematical programming.

Haiping Ma received the B.E. degree from Anhui University, Hefei, China, in 2008 and the Ph.D. degree from University of Science and Technology of China, Hefei, China, in 2013.

She is currently a Lecturer with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. Her current research interests include data mining and multi-objective optimization methods and their applications.

Xingyi Zhang (SM'18) received the B.Sc. degree from Fuyang Normal College, Fuyang, China, in 2003, and the M.Sc. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2009, respectively.

He is currently a Professor with the School of Artificial Intelligence, Anhui University, Hefei, China. His current research interests include unconventional models and algorithms of computation, evolutionary multi-objective optimization, and logistic scheduling. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award.



Kay Chen Tan (SM'08-F'14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor of the Department of Computing at the Hong Kong Polytechnic University, Hong Kong. He has published over 200 refereed articles and six books, and holds one U.S. patent on surface defect detection.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He has served as the Editor-in-Chief of IEEE Transactions on Evolutionary Computation from 2015-2020 and IEEE Computational Intelligence Magazine from 2010-2013, and currently serves as the Editorial Board Member of over 10 journals. He is currently an IEEE Distinguished Lecturer Program (DLP) speaker and Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.

Yaochu Jin (SM'02-F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Germany, in 2001.

He is an Alexander von Humboldt Professor for Artificial Intelligence, Chair of Nature Inspired Computing and Engineering, Faculty of Technology, Bielefeld University, Germany. He is also a Distinguished Chair, Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, U.K. He was a "Finland Distinguished Professor" of University of Jyväskylä, Finland, "Changjiang Distinguished Visiting Professor", Northeastern University, China, and "Distinguished Visiting Scholar", University of Technology Sydney, Australia. His main research interests include evolutionary optimization, evolutionary learning, trustworthy machine learning, and evolutionary developmental systems.

Prof. Jin is presently the Editor-in-Chief of Complex & Intelligent Systems. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS in 2016-2021 and an IEEE Distinguished Lecturer in 2013–2015 and 2017–2019, the Vice President for Technical Activities of the IEEE Computational Intelligence Society (2015–2016). He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, and the 2015, 2017, and 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award. He was named by the Web of Science as a Highly Cited Researcher from 2019 to 2021 consecutively. He is a Member of Academia Europaea.