Coevolutionary Particle Swarm Optimization With Bottleneck Objective Learning Strategy for Many-Objective Optimization

Xiao-Fang Liu, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Ying Gao, Jie Zhang, Sam Kwong, *Fellow, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The application of multiobjective evolutionary algorithms to many-objective optimization problems often faces challenges in terms of diversity and convergence. On the one hand, with a limited population size, it is difficult for an algorithm to cover different parts of the whole Pareto front (PF) in a large objective space. The algorithm tends to concentrate only on limited areas. On the other hand, as the number of objectives increases, solutions easily have poor values on some objectives, which can be regarded as poor bottleneck objectives that restrict solutions' convergence to the PF. Thus, we propose a coevolutionary particle swarm optimization with a bottleneck objective learning (BOL) strategy for many-objective optimization. In the proposed algorithm, multiple swarms coevolve in distributed fashion to maintain diversity for approximating different parts of the whole PF, and a novel BOL strategy is developed to improve convergence on all objectives. In addition, we develop a solution reproduction procedure with both an elitist learning strategy (ELS) and a juncture learning strategy (JLS) to improve the quality of archived solutions. The ELS helps the algorithm to jump out of local PFs, and the JLS helps to reach out to the missing areas of the PF that are easily missed by the swarms. The performance of the proposed algorithm is evaluated using two widely used test suites with different numbers of objectives. Experimental results show that the proposed algorithm compares favorably with six other state-of-the-art algorithms on many-objective optimization.**

Manuscript received July 01, 2017; revised November 24, 2017, May 02, 2018, and August 13, 2018; accepted September 28, 2018. Date of publication October 11, 2018; date of current version July 30, 2019. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundations of China under Grant 61772207 and Grant 61332002, in part by the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars under Grant 2014A030306038, in part by the Project for Pearl River New Star in Science and Technology under Grant 201506010047, in part by the GDUPS in 2016, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

X.-F. Liu, Z.-H. Zhan, Y. Gao, and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com; junzhang@ieee.org).

S. Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

J. Zhang is with the School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China.

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2018.2875430

***Index Terms*—Bottleneck objective learning (BOL), coevolution, many-objective optimization problems (MaOPs), particle swarm optimization (PSO).**

I. Introduction

During the past two decades, multiobjective evolutionary algorithms (MOEAs) have shown efficiency in finding well-diversified and well-converged nondominated solutions in multiobjective optimization problems (MOPs) [1]. However, when dealing with many-objective optimization problems (MaOPs) that involve more than three objectives, the "curse of dimensionality" in MaOPs has brought challenges in both diversity and convergence to traditional MOEAs [2], [3].

The first is the diversity challenge. As the objective space enlarges in MaOPs, it becomes more difficult for an algorithm to maintain population diversity so as to find solutions well distributed along the whole Pareto front (PF). On the one hand, as the dimensionality of the objective space increases, the number of solutions that are required for approximating the whole PF increases exponentially [1]. Therefore, an MOEA that uses only one population with a limited population size tends to concentrate only on a limited area of the large objective space [4]. On the other hand, due to the difficulty of distance evaluation in high-dimensional objective space [5], many existing solution diversity maintenance mechanisms developed for traditional MOPs may not work well on MaOPs. For example, the crowding distance (CD) measure for density estimation in NSGA-II fails in MaOPs [6], [7]. Thus, the population may lose diversity to distribute solutions on the whole PF and can approximate to only a limit part of the PF [8].

The second is the convergence challenge. As the number of objectives increases, it becomes more difficult for an algorithm to pay the same attention to all the objectives. Therefore, the evolution in different objectives may be imbalanced, causing some objectives to perform well while other objectives perform poorly. That is, a solution may have very good values on some objectives but very poor values on some other objectives. On the one hand, Pareto dominance fails to distinguish such solutions and consequently most of these solutions tend to be mutually nondominated and will be preserved. On the other hand, the loss of proper guidance on the poorly performing objectives will likely aggravate the evolution imbalance

of different objectives in solutions during the evolutionary process. Thus, the population tends to be far away from the desired areas, leading to poor convergence to the PF [9].

Therefore, it is crucial for an algorithm to achieve good diversity and impartial convergence to the PF when solving MaOPs. Many approaches have recently been developed for MaOPs, such as NSGA-III [2] and MOEA/D-based approaches [10], [35]. Most of them consider all objectives as a whole by one population. However, as discussed above, it is often difficult to maintain diversity for exploring all objectives by using only one population. Alternatively, studies have shown that coevolutionary mechanisms with multiple populations can effectively enhance algorithm diversity and improve optimization efficiency [11]. Among them, the multiple populations for multiple objectives (MPMO) framework is a coevolutionary technique and has shown good performance on MOPs [12]. Since there are many objectives in MaOPs, it is intuitive to treat each objective as a subproblem and adopt the MPMO framework to optimize different objectives in different populations. The optimization of different objectives in different populations helps to locate different parts of the PF and to avoid neglecting some objectives. Therefore, in this paper, we study the application of the MPMO framework [12] for solving MaOPs. Furthermore, as particle swarm optimization (PSO) [13] is simple but efficient in solving optimization problems, it is a good choice to be adopted for each population.

Since each swarm concentrates on one objective in the MPMO framework, we name the corresponding objective as the population objective (PO) of each swarm. By sufficiently exploring the POs in all swarms, the MPMO framework maintains diversity efficiently so as to obtain solutions well distributed along the whole PF. Moreover, in order to promote the cooperation among swarms, we maintain an archive to aggregate the information of all swarms. The archive stores the nondominated solutions from all swarms for information sharing. Furthermore, as mentioned above, a solution (i.e., particle) may have very poor values on some objectives. These poorly performing objectives are indeed the bottleneck objectives (BO) that limit the convergence rate of the particle toward the PF. Thus, paying more attention to BO optimization will help to push the particle toward the PF gradually. In fact, the archive can be used to complement BO optimization since different nondominated solutions in the archive often have good information in different POs. Therefore, we propose a bottleneck objective learning (BOL) strategy to optimize a particle's BO with the help of the solutions from the archive. In the BOL strategy, a particle first determines its own BO and then updates itself by learning from a solution, which is selected from the archive according to the dominance relation on two objectives, i.e., the PO and the BO of this particle. Hence, the BOL strategy enables the particle to optimize its PO and BO simultaneously and thereby the particle is able to better converge to the PF.

Incorporating the MPMO framework with the BOL strategy, we develop a coevolutionary PSO (CPSO) for MaOPs. The cooperation of the MPMO framework and the BOL strategy promotes the swarms to converge to different parts of the PF. In addition, a solution reproduction (SR) procedure with an elite learning strategy (ELS) and a juncture learning strategy (JLS) is performed to improve the quality of archived solutions. Since the number of nondominated solutions increases rapidly, a solution preservation (SP) procedure is carried out for archive updates to update the archived solutions.

Therefore, the contribution of the proposed CPSO is mainly in maintaining diversity by using a multipopulation coevolution technique and to accelerate convergence by steadily optimizing the BO of the solutions. Specially, three components are developed to deal with the challenges of MaOPs. First, the MPMO component with BOL strategy not only helps to generate diverse solutions to different parts of the PF, but also improves solution convergence on all objectives. Second, the SR component with ELS and JLS operators helps to jump out of local PFs and reach out to the areas of the PF that are easily missed by the swarms. Third, the SP component keeps the good solutions found by the swarms during the search process in an archive, such that the archive can facilitate communications among the swarms. In order to evaluate the advantages of the proposed CPSO, two widely used test suits are adopted and six state-of-the-art algorithms are compared. The experimental results show that the proposed CPSO is promising in dealing with MaOPs.

The rest of this paper is organized as follows. Section II discusses the background. Section III develops the proposed CPSO algorithm for MaOPs. Section IV presents the experimental results in comparison with other MOEAs for MaOPs, and makes a detailed analysis of CPSO. Finally, the conclusions are summarized in Section V.

II. Background

A. PSO

PSO is a simple yet efficient swarm intelligence algorithm proposed by Kennedy and Eberhart [13]. In a D-dimensional space, each particle i is associated with a position vector $X_i = (X_{i1}, X_{i2}, \ldots, X_{iD})$ and a velocity $V_i = (V_{i1}, V_{i2}, \ldots, V_{iD})$, where $i = 1, 2, \ldots, N$ and N is the population size. Each particle i stores its historically best position as $p\text{Best}_i = (p\text{Best}_{i1}, p\text{Best}_{i2}, \ldots, p\text{Best}_{iD})$. The best of all $p\text{Best}_i$ is denoted as the globally best position $g\text{Best} = (g\text{Best}_1, g\text{Best}_2, \ldots, g\text{Best}_D)$. At the beginning, each particle randomly initializes its position according to a uniform distribution in the search space $[X_{\min,d}, X_{\max,d}]$ for each dimension d and initializes its velocity to 0 or a random value in the range of $[-V_{\max,d}, V_{\max,d}]$ for each dimension d, where $V_{\max,d}$ is a predefined positive value and is usually set as 20% of the search space size, i.e., $V_{\max,d} = 0.2 \times (X_{\max,d} - X_{\min,d})$. In each generation, each particle i updates its velocity and position with the guidance of $p\text{Best}_i$ and $g\text{Best}$ by

$$V_{\text{id}} = \omega V_{\text{id}} + c_1 r_{1d}\left(p\text{Best}_{\text{id}} - X_{\text{id}}\right) + c_2 r_{2d}\left(g\text{Best}_d - X_{\text{id}}\right) \tag{1}$$

$$X_{\text{id}} = X_{\text{id}} + V_{\text{id}} \tag{2}$$

where ω is the inertia weight and is suggested to decrease linearly from 0.9 to 0.4 during the search process [14], c_1 and c_2 are acceleration coefficients [15], and both r_{1d} and r_{2d} are

random numbers in the range of [0, 1.0] for the dth dimension. After the update, if some dimensions of the velocity or the position violate the boundary constraints, they will be set as the bound values of the corresponding dimensions. The $p\text{Best}_i$ and $g\text{Best}$ will be updated if the new position has a better fitness value.

B. MaOPs

A minimization MaOP can be defined as

$$\text{minimize} \quad F(X) = (f_1(X), f_2(X), \ldots, f_M(X))^T \tag{3}$$

$$\text{subject to} \quad X \in S^D \tag{4}$$

where $X = (X_1, X_2, \ldots, X_D)$ is a decision vector in the D-dimensional search space S^D, M is the number of objectives ($M > 3$), and $F(X) = (f_1(X), f_2(X), \ldots, f_M(X))^T$ is the objective function vector.

Definition 1 (Pareto Dominance): Given two points Z and Y in the objective space. Z dominates Y, denoted as $Z \preceq Y$, if and only if $\forall j \in \{1, 2, \ldots, M\}, Z_j \leq Y_j$, and $Z \neq Y$.

Definition 2 (Pareto Optimal Solution): A solution $X^* \in S^D$ is called Pareto optimal if and only if $F(X^*)$ is not dominated by $F(X)$ of any solution $X \in S^D$.

Definition 3 (Pareto Set): Pareto Set (PS) is the set of Pareto optimal solutions: $\text{PS} = \{X \in S^D | X \text{ is Pareto optimal}\}$.

Definition 4 (PF): PF is the set of the corresponding objective vectors of the solutions in PS: $\text{PF} = \{F(X) | X \in \text{PS}\}$.

C. Related Work for MaOPs

In the literature, various techniques have been proposed to obtain well-diversified and well-converged solutions for MaOPs. The methods can be loosely classified into four categories, i.e., Pareto-based approaches, indicator-based approaches, preference-based approaches, and decomposition-based approaches. First, Pareto-based approaches adopt new dominance relationships or diversity maintenance mechanisms to distinguish the nondominated solutions, including relaxed dominance, e.g., the ϵ-dominance [17] and θ-dominance [18], and convergence and diversity management mechanisms, e.g., elaborate density estimator in GrEA [19], shift-based density estimation strategy [20], reference-point-based method in NSGA-III [2], knee points driven method in KnEA [21], direction- and convergence-based selection method [22], and objective space reduction method in MaOEA-R&D [8]. Second, indicator-based approaches use different performance indicators to assess the quality of solutions, such as the ϵ-indicator [24] and the hypervolume (HV) indicator [25], [26]. Among them, the HV indicator has proven to be strictly equivalent to Pareto dominance [27]. However, the high calculation cost of the exact value limits its application [28], while inaccurate estimation may degrade its performance [18]. The Two_Arch2 adopts a quality indicator for fast convergence and Pareto dominance for diversity maintenance [5]. In [23], multiple indicators are used to combine their advantages. Third, preference-based approaches introduce preference information into the optimization process for SP. For example, PICEAs [29] archive the PF by coevolving solutions and the decision maker's preferences. Fourth, decomposition-based approaches decompose the problem into multiple subproblems and optimize them simultaneously by different strategies, such as reference vector guided strategy [30] and aggregation-based strategy [10], [31]. The performance of decomposition-based algorithms greatly relies on the setting of weighting vectors and the selection of aggregation functions [32], [33]. A recent local weight sum setting method is designed in MOEA/D-LWS [34]. An adaptive scalarizing method is also developed in MOEA/D-PaS [35]. MOEA/DD combines the advantages of dominance and decomposition methods [43].

Different from the aforementioned approaches using only one population, we solve the MaOPs by the coevolutionary framework MPMO [12], [16]. Due to its good performance on MOPs, the MPMO is believed to be a promising method for MaOPs. Besides, applications of PSO on MaOPs have recently attracted attention. Multiple PSO algorithms with different learning strategies have been developed for MaOPs. For example, MaOPO/2s-pccs [36] randomly employs solutions with good convergence or good diversity for learning, but the performance of the algorithm needs to be improved; and in NMPSO [37], convergence and diversity are combined by weights to evaluate solutions for learning exemplar selection, but the performance of the algorithm is sensitive to the weight setting. Multiswarm algorithms with different archiving methods and communication topologies have also been proposed for MaOPs [38]–[40]. However, continuous restarting or overlapping of search areas in different swarms may cause computational waste [41], [42]. Thus, there is still room for the design of PSO for MaOPs.

D. MPMO

To deal with the issues of fitness assignment and diversity maintenance in MOPs, Zhan *et al.* [12] proposed the coevolutionary technique MPMO. For an M-objective problem, M populations are created and each population optimizes one different objective concurrently. Each population can adopt any single-objective optimization algorithm and the individuals compete to survive based on the corresponding PO, e.g., f_j in the jth population. The nondominated solutions discovered by all populations are stored in an archive for population communication. Using the archive, the multiple populations coevolve toward the PF.

III. CPSO for MaOPs

In this section, the algorithm framework, the search process of each swarm, and the archive update process of CPSO are presented in detail.

A. Algorithm Framework

CPSO adopts the MPMO technique and uses multiple swarms to coevolve toward the PF. Given an MaOP with M objectives, each objective is optimized by a population (swarm in CPSO) and it is denoted as a PO. The M swarms adopt PSO with BOL strategy to optimize their POs concurrently. For swarm communication, an external archive A with fixed size NA is used to store the nondominated solutions from all

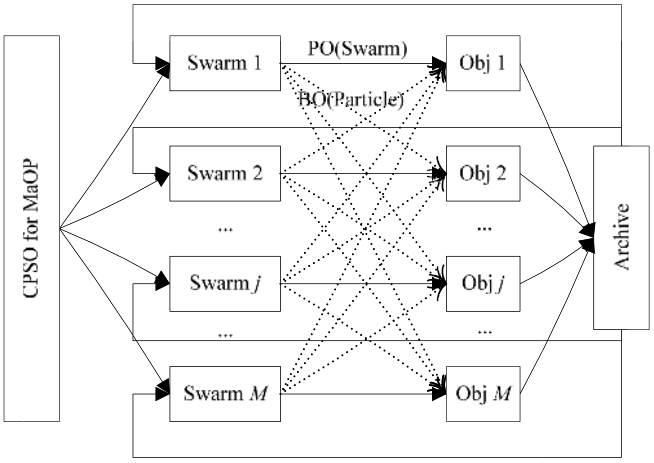

Fig. 1. Framework of CPSO for solving M-objective optimization problems. Between swarms and objectives, the solid lines with arrow represent the corresponding optimization objectives of the swarms, and the dashed lines with arrow represent the BO of different particles.

swarms. At the end of each generation, an SR procedure is performed to improve the quality of the archived solutions. Then an SP procedure is adopted to update the archive using the reproduced solutions, the solutions in the archive, and all positions and pBests of particles in all swarms. The final archive is regarded as the solution set found by the algorithm. In general, the MPMO component with BOL strategy is the basis of the CPSO algorithm, the SR component is an auxiliary operator, and the SP component is for information storage. They complement each other as a complete system as shown in Fig. 1.

B. Search Process of Each Swarm

Without loss of generality, we take the jth swarm as an example to describe the search process, as it is comparable to the other swarms. The PO of the jth swarm is objective j. In every generation, each particle in the jth swarm updates its velocity by the BOL strategy.

1) BOL Strategy: The BO of a particle can be a set of objectives possessing poor values. In this paper, we only take the worst-performed objective as the BO of the particle. To determine the BO, we define an optimization degree (OD) to indicate the performance of each objective in the particle. The OD is calculated as follows. For an objective l, we normalize the fitness value of objective l in the particle to obtain the OD, using the best and the worst values of objective l in all the nondominated solutions of last generation, denoted as f_l^{best} and f_l^{worst}, respectively. Assume the particle is indexed by i and the objective vector of its position X_i is $F(X_i) = (f_{i1}, \ldots, f_{iM})$. For a minimization problem, the OD od_{il} of objective l of particle i is calculated by

$$\text{od}_{il} = \begin{cases} \frac{f_{il} - f_l^{\text{best}}}{f_l^{\text{worst}} - f_l^{\text{best}}}, & \text{if } f_l^{\text{worst}} > f_l^{\text{best}} \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Note that in the first generation, f_l^{best} and f_l^{worst} are set as the best and the worst values of objective l in the pBests of all particles in all swarms, respectively. Then, the objective with the worst (maximal) OD value is selected as the BO b_i of particle i

$$b_i = \arg\max_{1 \le l \le M} \text{od}_{il}. \tag{6}$$

Next, particle i updates its velocity by learning from $p\text{Best}_i$ and an archive solution $\text{Archive}_i^{j,b_i}$ as

$$\begin{aligned} V_{\text{id}} = {} & \omega V_{\text{id}} + c_1 r_{1d}\left(p\text{Best}_{\text{id}} - X_{\text{id}}\right) \\ & + c_2 r_{2d}\left(\text{Archive}_{\text{id}}^{j,b_i} - X_{\text{id}}\right) \end{aligned} \tag{7}$$

where $\text{Archive}_i^{j,b_i}$ is selected from archive A according to the dominance relation on two objectives, i.e., the PO(j) and the BO(b_i). The detailed selection procedure of $\text{Archive}_i^{j,b_i}$ is shown in Algorithm 1 and described as follows. At the beginning, $\text{Archive}_i^{j,b_i}$ is set as the first solution in A. Then $\text{Archive}_i^{j,b_i}$ is compared with the solutions in A sequentially. In each comparison, if $\text{Archive}_i^{j,b_i}$ is Pareto dominated by the solution in A on the PO and the BO, it is replaced by the solution in A; otherwise, if the two solutions are not dominated by each other, then $\text{Archive}_i^{j,b_i}$ will be updated only if the solution in A has a smaller or the same convergence performance (CP) value. The CP value is calculated by the sum of OD of all the objectives as

$$\text{CP} = \sum_{l=1}^{M} \text{od}_l. \tag{8}$$

If archive A is empty, $\text{Archive}_i^{j,b_i}$ is temporally set as the gBest of a randomly selected swarm. Given the number of solutions in the archive n, since the domination comparisons are performed in only $(n-1)$ times, it cannot ensure that the best nondominated solution on PO and BO is always selected. However, this kind of pairwise comparison reduces time consumption as well as introduces randomness. An example of archive solution selection in a 3-objective problem is illustrated in Fig. 2. Assume that a particle in swarm 1 has fitness values X(2,3,4) and $p$best(1,4,5), and that archive A is $\{A^1(3,5,1), A^2(3,1,5), A^3(5,3,1), A^4(1,2,3)\}$. Then the PO of swarm 1 is f_1 and the BO of the particle is f_3. According to the dominance relation on f_1 and f_3, A^4 will be selected.

The selection of archive solution based on PO and BO takes effect on two sides. On the one hand, using only PO and BO

Algorithm 1 Archive Solution Selection Procedure for Particle Update

Input: number of solutions in archive (n), archive $A = \{A^1, A^2, \ldots, A^n\}$, PO, BO

```
A^P = A^1
For k = 2 to n
    If A^k Pareto dominates A^P on PO and BO then
        A^P = A^k
    Else if A^k and A^P are nondominated by each other on PO and BO then
        If CP_k ≤ CP_P then
            A^P = A^k
        End
    End
End
```

Output: A^P

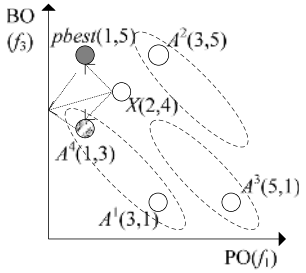

Fig. 2. Example of archive solution selection for a particle based on the PO and particle's BO.

for dominance comparisons reduces the number of objective comparison dimensions. This consequently enhances selection pressure and helps to distinguish solutions. On the other hand, using both PO and BO for dominance comparisons allows the selection of an archive solution with better values on PO and BO. Thus, the selection helps avoid the possible negative effect caused by an unsuitable archive solution with poor value on PO and is also beneficial to the optimization of BO. With the BOL strategy, the particle is able to optimize the PO and its BO and hence it converges to the PF gradually.

2) Update of pBests and gBest: The *p*Best of particle i is updated if the new position has a better or equal fitness value on objective j. Likewise, the *g*Best of the swarm will be updated if the updated *p*Best in the swarm has a better or equal fitness value on objective j.

C. Archive Update

At the end of each generation, the archive is updated. First, a new set P is initialized by the *p*Bests and positions X of all particles in all swarms, as well as the solutions in the archive. Later, two operators, the ELS and JLS, are performed on the solutions in the archive for SR. Then, all the new reproduced solutions are also added into the set P. Finally, according to the dominance relation on all objectives, the nondominated solutions in P are stored in the archive A.

1) Solution Reproduction:

Elitist Learning Strategy (ELS): To jump out of possible local optima, we adopt the ELS proposed in an adaptive PSO [15]. In CPSO, ELS is performed on the first $0.9n$ solutions in archive A. For a solution, one dimension d is randomly selected to move with a step size which is generated by Gaussian distribution as

$$X_{\mathrm{id}} = X_{\mathrm{id}} + \left(X_{\max,d} - X_{\min,d}\right)\mathrm{Gaussian}\left(0, \sigma^2\right) \tag{9}$$

$$\sigma = 0.5 - (0.5 - 0.1)\frac{\mathrm{fes}}{\mathrm{FES}} \tag{10}$$

where $X_{\min,d}$ and $X_{\max,d}$ are the lower and upper bounds of the dth dimension, respectively; fes is the already-consumed function evaluations; FES is the maximum available function evaluations; and Gaussian(0, σ^2) is a Gaussian distribution of mean 0, and its standard deviation σ decreases from 0.5 to 0.1 as fes increases. The new solutions are then added into the set P.

JLS: Since different swarms focus on different objectives, they move toward different areas especially the margins of the PF. Thus, the approximated fronts from all swarms may lose their juncture areas, such as the center of the PF. Therefore, we introduce an operator JLS to generate new solutions in the juncture areas by learning from different swarms. In JLS, a crossover and mutation process is performed to inherit different dimensions from different solutions. Two parents are randomly selected from the archive and then perform simulated binary crossover (SBX) and polynomial mutation (PM) [2]. In every generation, the JLS is performed to generate $0.1n$ new solutions and these new solutions are then added into the set P.

2) Solution Preservation: The next step is to choose the nondominated solutions from P to update A. First, all the dominated solutions are deleted from P. Then, the best and the worst values of each objective among the nondominated solutions are denoted as f_j^{best} and f_j^{worst} for objective j. Meanwhile, archive A is emptied and the size of archive n is set as 0. If the size of P is equal to or less than NA, the solutions in P are stored in A, and n is updated as the size of P. Otherwise, only NA solutions that maximize the diversity of the nondominated solutions will be chosen to store in A. The solutions are selected based on reference points, which are generated by double-layer method [2]. Different from NSGA-III [2], the number of reference points is allowed to be larger than NA. Among the reference points, we select NA reference points with maximal diversity for solution selection. In addition, we also introduce the metric CP into solution selection procedure to improve convergence. The reference-point-based solution selection procedure is described as follows.

The best values of all objectives form the ideal point $(f_1^{\mathrm{best}}, \ldots, f_M^{\mathrm{best}})$ of set P. The objective vector of each solution in P subtracts the ideal point to form a new objective vector. Then in set P, M extreme points in M objective axes are identified by minimizing the corresponding achievement scalarizing functions [2]. Next, these extreme points are used to construct a hyperplane. Then the interception points of the hyperplane and the axes are determined to normalize the solutions in set P. Note that if the M extreme points cannot construct a $(M-1)$ dimensional hyperplane or the obtained interception points are nonpositive, the interception points of each axis are set as $(f_j^{\mathrm{worst}} - f_j^{\mathrm{best}})$ corresponding to objective j. Later, each normalized solution is associated with a reference line according to its perpendicular distance from the reference line. The reference lines are constructed by joining each reference point with the origin. Finally, a niche preservation operation is performed to select NA solutions in NA steps. Each reference line owns a count to calculate the number of already selected solutions associated with it. A set R stores the reference lines with minimum count and an additional set K records the visited reference lines. In each step, the reference line with minimum count is first selected. If there are multiple reference lines having minimum count (the size of R is larger than 1), then the one r with the largest distance d_r to the visited reference lines in K will be selected by

$$r = \arg\max_{q \in R} d_q \tag{11}$$

$$d_q = \sum_{k \in K} \mathrm{dis}\left(w_q, w_k\right), q \in R \tag{12}$$

$$\mathrm{dis}(w_q, w_k) = \sum_{j=1}^{M} |w_{qj} - w_{kj}| \tag{13}$$

where d_q is the total distances of reference line q to the visited reference lines in K, w_q and w_k represent the normalized weights of reference lines q and k, respectively. If K is empty, a random reference line is selected from R. After that, the r is added into K, and one solution associated with r will be selected and added into archive A. If the count of r is 0, the solution with minimum perpendicular distance to r is chosen. Otherwise, the solution with smallest CP value defined as (8) is selected. The step of reference line and solution selection repeats until n becomes NA.

The overall flowchart of CPSO is shown in Fig. 3.

D. Computational Complexity of One Generation of CPSO

Given an MaOP with M objectives in a D-dimensional decision space. Assume that the population size of each swarm is S, the number of reference points used in the SP component is H, the maximum archive size is NA, and the current archive size is $n \leq NA$. The computational complexity of one generation of CPSO includes that of the MPMO, the SR, and the SP components.

First, in the MPMO component, the calculation of CP values requires $O(M \times n)$ computations for all archived solutions. In the BOL strategy, the determination of BO requires $O(M)$ computations, archive solution selection requires $O(n)$ computations, and particle update requires $O(D)$ computations. Thus, the complexity of the BOL strategy is $O(M+n+D)$. Therefore, the complexity of the MPMO component, including the search process of M swarms of size S, is $O(M \times S \times (M+n+D) + M \times n)$, that is $O(M \times S \times (M+n+D))$.

Second, in the SR component, the ELS requires $O(n)$ computations and the JLS requires $O(n \times D)$ computations. Therefore, the complexity of the SR component is $O(n \times D)$.

Third, in the SP component, the nondominated sorting of a set P of size $(M \times S \times 2 + n \times 2)$ requires $O((M \times S+n)^2 \times M)$ computations. The identification of the ideal point requires $O(M \times (M \times S + n))$, the calculation of new objective vectors requires $O(M \times (M \times S + n))$, and the determination of the extreme points requires $O(M^2 \times (M \times S + n))$ computations. Then hyperplane construction and intercept identification require $O(M^3)$ computations. Next, the normalization of the solutions requires $O(M \times (M \times S + n))$ computations. Thereafter, associating all solutions to reference lines requires $O(M \times H \times (M \times S+n))$ computations. In the niche preservation operation, the calculation of the CP values of all solutions in set P requires $O(M \times (M \times S+n))$ computations. The distances between any two reference lines (13) only need to be calculated one time [with complexity of $O(H^2 \times M)$] and can be stored at the beginning of the algorithm, so it is not included in each generation. Similarly, by using a memory of size H to store and accumulate the total distances of each reference line to the visited reference lines (12), the selection of reference line only requires $O(H)$ computations and the selection of a solution associated with the selected reference line requires $O(M \times S+n)$ computations. Once one reference line is selected and becomes visited, the total distances of each reference line to the visited reference lines are updated, which requires $O(H)$ computations. Hence, the niche preservation operation requires $O(NA \times (H + M \times S + n + H) + M \times (M \times S + n))$ computations to preserve NA solutions. Thus, the SP component requires $O((M \times S + n)^2 \times M + M^2 \times (M \times S + n) + M^3 + M \times H \times (M \times S + n) + NA \times (H + M \times S + n))$ computations. In our simulations, we have used $H \approx NA$, $n \leq NA$, and usually $NA > M$. Thus, the complexity of the SP component is $O(NA^2 \times M + M^3 \times S^2 + M^2 \times S \times NA)$.

Therefore, the overall worst-case complexity of one generation of CPSO is $O(M \times S \times (M + n + D) + n \times D + NA^2 \times M + M^3 \times S^2 + M^2 \times S \times NA)$, which can be reduced to $O(M^3 \times S^2 + M^2 \times S \times NA + NA^2 \times M + NA \times D + M \times S \times D)$. Define the total size of all swarms as $N = M \times S$. In all of our simulations, we have used $NA = N$. Thus, the worst-case complexity of one generation of CPSO can be simplified to $O(N^2 \times M + N \times D)$. That is, the larger one between $O(N^2 \times M)$ and $O(N \times D)$. This shows that the complexity of one generation of CPSO is dominated by the nondominated sorting with $O(N^2 \times M)$ and solution generation with $O(N \times D)$ like many other algorithms for MaOPs. Therefore, the added operators are light and do not increase the overall algorithm complexity.

TABLE I
SETTING OF THE DTLZ AND WFG PROBLEMS

Problems	M	D	Parameter
DTLZ1	5,10	$M-1+k$	k=5
DTLZ2-DTLZ6	5,10	$M-1+k$	k=10
DTLZ7	5,10	$M-1+k$	k=20
WFG1-WFG9	5,10	$k+l$	$k = M-1$, l =20

IV. EXPERIMENTS AND COMPARISONS

A. Experimental Design

1) Benchmark Problems: Two widely used test suites, DTLZ problems DTLZ1-DTLZ7 [44], and WFG problems WFG1-WFG9 [45] are used to evaluate the algorithm performance. These challenging problems have different characteristics. The DTLZ problems possess features, such as nonuniform, disconnected, multimodal, and nonconvex; while the WFG problems have properties of scalable objective, nonseparable, deceptive, mixed shape, degenerated, and dependencies between position- and distance-related parameters. In the experiments, two instances with 5-objective and 10-objective for each problem are tested. The settings of decision variable dimension D, position parameter k, and distance parameter l for each problem are listed in Table I following the suggestions in [44] and [45].

2) Performance Metric: Two widely used performance metrics, inverted generational distance (IGD) [46] and HV [47], are adopted to evaluate the performances of the solutions. In each problem, a large number of reference points (i.e., 100 000 samples) are generated for IGD metric following [8], [55]. For the PFs of DTLZ1-4 and WFG4-9 that are regular as hyperplanes or hyperspheres, we sample reference points randomly from the PFs. Since random sampling may give nonuniform reference points [56], we propose a two-step strategy to sample points in this paper. Based on the

TABLE II
PERFORMANCE COMPARISON BETWEEN CPSO AND OTHER ALGORITHMS IN TERMS OF IGD VALUE ON DTLZS AND WFGS

Problem	M	Approaches	Pareto based						Decomposition based		Indicator based		Preference based	
		MPMO based	Relax dominance		Convergence and Diversity Management									
		CPSO	θ-DEA		NSGA-III		NMPSO		MOEA/D-PaS		HypE		PICEA-g	
		Mean(Std Dev)	Mean(Std Dev)		Mean(Std Dev)		Mean(Std Dev)		Mean(Std Dev)		Mean(Std Dev)		Mean(Std Dev)	
DTLZ1	5	**0.0601(0.0008)**	0.0638(0.0002)	+	0.0642(0.0015)	+	0.1241(0.0217)	+	*0.0604(0.0007)*	+	0.3729(0.3280)	+	0.3705(0.0025)	+
	10	0.1550(0.0197)	*0.1152(0.0180)*	−	**0.1116(0.0032)**	−	0.1936(0.0297)	+	0.2069(0.0295)	+	0.9901(0.6360)	+	0.4682(0.0003)	+
DTLZ2	5	0.2378(0.0079)	*0.2270(0.0001)*	−	**0.2268(0.0004)**	−	0.2474(0.0055)	+	0.2383(0.0041)	=	0.4149(0.0295)	+	0.2553(0.0002)	+
	10	**0.3860(0.0139)**	0.4283(0.0005)	+	0.4291(0.0007)	+	*0.4282(0.0033)*	+	0.5598(0.0605)	+	0.6622(0.0723)	+	0.8594(0.0033)	+
DTLZ3	5	0.8857(0.4540)	*0.2283(0.0011)*	−	0.2309(0.0029)	−	0.3152(0.0401)	−	0.2511(0.0208)	−	3.9702(1.9291)	+	1.1298(0.0044)	+
	10	5.7622(2.3730)	**0.4706(0.1037)**	−	0.9039(0.8552)	−	*0.6030(0.0712)*	−	0.8703(0.2350)	−	14.963(9.9181)	+	1.2733(0.0006)	−
DTLZ4	5	0.2802(0.0155)	*0.2272(0.0001)*	−	**0.2270(0.0003)**	−	0.2464(0.0049)	−	0.4690(0.1859)	+	0.5468(0.1558)	+	0.4838(0.0283)	+
	10	**0.4171(0.0110)**	*0.4305(0.0010)*	+	0.4308(0.0014)	+	0.4358(0.0035)	+	0.5971(0.0867)	+	0.8237(0.0947)	+	0.6965(0.0070)	+
DTLZ5	5	0.0733(0.0174)	0.1293(0.0368)	+	0.2043(0.0542)	+	0.0543(0.0078)	−	**0.0488(0.0121)**	−	0.3018(0.0815)	+	*0.0492(0.0008)*	−
	10	*0.1586(0.0563)*	0.3038(0.0819)	+	0.3625(0.0628)	+	0.7308(0.0530)	+	0.1699(0.0217)	=	0.3672(0.0270)	+	**0.1129(0.0011)**	−
DTLZ6	5	*0.1095(0.0320)*	0.4775(0.0478)	+	2.4094(0.2313)	+	0.1200(0.0853)	=	**0.0933(0.0324)**	−	5.9388(0.4454)	+	0.4790(0.0179)	+
	10	*0.2858(0.1029)*	0.5019(0.1008)	+	3.9171(0.2979)	+	0.7391(0.0476)	+	**0.1936(0.0367)**	−	6.1862(0.4157)	+	0.5583(0.0312)	+
DTLZ7	5	0.4653(0.0475)	0.5986(0.0753)	+	*0.4160(0.0140)*	−	**0.3324(0.0098)**	−	0.4529(0.0580)	=	0.6829(0.0360)	+	2.4058(0.0066)	+
	10	1.3426(0.0909)	1.7397(0.2174)	+	1.9717(0.3030)	+	*1.1074(0.0574)*	−	**1.0998(0.0167)**	−	1.5134(0.2861)	+	5.2729(0.0051)	+
WFG1	5	1.7592(0.0461)	1.2622(0.0705)	−	*1.0180(0.1221)*	−	1.1048(0.3725)	−	**0.8942(0.0942)**	−	1.6418(0.0312)	−	1.2166(0.0004)	−
	10	2.5571(0.0631)	2.7687(0.5060)	=	**1.2386(0.1110)**	−	*1.5994(0.7389)*	−	1.8621(0.1512)	−	2.5627(0.0630)	=	1.9078(0.0007)	−
WFG2	5	0.7962(0.0546)	1.1457(0.0297)	+	0.8662(0.0734)	+	1.4676(0.6132)	+	*0.7140(0.0683)*	−	1.5597(0.2796)	+	**0.5364(0.0148)**	−
	10	2.5139(0.6022)	2.7889(1.1534)	=	3.7634(1.6069)	+	3.4290(1.1155)	+	*1.4524(0.1326)*	−	3.7529(0.5824)	+	**1.1533(0.0219)**	−
WFG3	5	0.3928(0.0774)	0.6600(0.1022)	+	0.7800(0.1933)	+	*0.3134(0.0280)*	−	0.4406(0.0947)	+	1.0036(0.1390)	+	**0.0957(0.0002)**	−
	10	*1.0914(0.1800)*	5.3124(1.8801)	+	1.9977(0.6377)	+	1.2032(0.1367)	+	1.3501(0.0850)	+	3.1994(0.8084)	+	**0.2620(0.0006)**	−
WFG4	5	*1.3040(0.0073)*	1.3050(0.0014)	=	**1.3029(0.0050)**	=	1.4642(0.0497)	+	1.3811(0.0383)	+	2.0412(0.0982)	+	1.3471(0.0602)	=
	10	**4.1432(0.0222)**	4.6158(0.0151)	+	4.6132(0.0114)	+	*4.3611(0.0433)*	+	6.6108(0.8608)	+	7.2014(0.8797)	+	9.6639(0.4542)	+
WFG5	5	*1.2784(0.0153)*	1.2952(0.0003)	+	1.2923(0.0026)	+	1.3224(0.0302)	+	1.4972(0.0494)	+	2.1444(0.1925)	+	**1.2637(0.0003)**	−
	10	**4.0110(0.0284)**	4.5833(0.0044)	+	4.5764(0.0070)	+	*4.2822(0.0361)*	+	4.8856(0.1383)	+	6.1066(0.4866)	+	7.5169(0.2429)	+
WFG6	5	**6.1466(0.0095)**	6.1955(0.0153)	+	6.3063(0.1837)	+	6.6241(0.0059)	+	*6.1571(0.0148)*	+	6.2308(0.0278)	+	6.5922(0.0076)	+
	10	17.068(0.0749)	*17.031(0.0760)*	=	17.504(0.4035)	+	17.630(0.1275)	+	**16.915(0.0141)**	−	17.071(0.0732)	=	17.486(0.0217)	+
WFG7	5	1.3555(0.0242)	**1.3083(0.0015)**	−	*1.3093(0.0023)*	−	1.5123(0.0494)	+	1.5114(0.0407)	+	2.7510(0.1818)	+	1.3174(0.0018)	−
	10	**4.2162(0.0216)**	4.6414(0.0178)	+	4.6415(0.0262)	+	*4.5252(0.0598)*	+	5.2527(0.3120)	+	7.7086(0.8782)	+	8.0494(0.3172)	+
WFG8	5	1.2885(0.0056)	*1.2798(0.0016)*	−	**1.2779(0.0033)**	−	1.4366(0.0246)	+	1.6650(0.0849)	+	2.6658(0.3804)	+	1.2881(0.0010)	=
	10	**4.0397(0.0310)**	4.4835(0.0345)	+	4.5288(0.0342)	+	*4.3981(0.0492)*	+	6.1131(0.5283)	+	9.0060(0.6243)	+	8.1231(0.3902)	+
WFG9	5	**6.3712(0.0258)**	6.7127(0.5437)	+	*6.6081(0.4870)*	+	7.1335(0.6388)	+	6.6724(0.5559)	+	6.9437(0.4918)	+	6.8837(0.2036)	+
	10	**17.270(0.0685)**	17.673(0.5690)	+	18.176(0.9077)	+	18.130(0.5780)	+	*17.533(0.6082)*	=	17.903(0.4577)	+	17.876(0.1126)	+
# +/=/−		DTLZ	9/0/5		8/0/6		7/1/6		5/3/6		14/0/0		11/0/3	
		WFG	11/4/3		13/1/4		15/0/3		12/1/5		15/2/1		8/2/8	
		Total	20/4/8		21/1/10		22/1/9		17/4/11		29/2/1		19/2/11	

symmetry and correlation of the PF, we first randomly generate the values of different objective dimensions within their maximum available ranges sequentially, and then randomly shuffle different objective dimensions to form a reference point. In this way, each point on the PF has an equal probability to be sampled. With a large number of samples, we can obtain well-distributed reference points. For example, given a 5-objective problem with PF of $f_1 + f_2 + f_3 + f_4 + f_5 = 0.5$. To generate a point $(r_1, \ldots, r_5)$, r_1 is first randomly generated in the range of [0, 0.5], then r_j $(2 \leq j \leq 4)$ is randomly generated in the range of $[0, 0.5 - \sum_{k=1}^{j-1} r_k]$ sequentially, and at last r_5 is set to $0.5 - \sum_{k=1}^{4} r_k$. After that, the values of $r_1, \ldots, r_5$ are randomly shuffled to form a reference point. Note that for WFG4-9 with scale objectives, we first sample points on normalized PFs and scale them afterward. With regard to DTLZ5-7 and WFG1-3 problems that have irregular or degenerated PFs, we first randomly sample points in their PSs and then use their corresponding objectives as reference points following [22]. The reference points for all the problems are archived as supplemental data of this paper. To calculate HV, the reference point for the DTLZ problems is set at 22 in each objective dimension [8]. Since WFG are scaled problems, we set the jth objective value of the reference points as $2j + 8$ [45].

3) Competitor Algorithms: Six state-of-the-art algorithms are taken for comparisons to verify the CPSO, i.e., θ-DEA [18], NSGA-III [2], NMPSO [37], MOEA/D-PaS [35], HypE [28], and PICEA-g [29]. They are typical methods from different categories with different techniques (as defined in Section II). Among them, θ-DEA is a recent relaxed dominance-based approach. NSGA-III and NMPSO are diversity and convergence improvement methods. NSGA-III adopts a reference point-based method for solution diversity maintenance. NMPSO considers convergence and diversity simultaneously for solution selection. MOEA/D-PaS is a decomposition-based algorithm with Pareto adaptive scalarizing method. HypE adopts the HV indicator to evaluate solution quality. PICEA-g is a preference-based coevolution approach. These representative algorithms make the comparisons more comprehensive and convincing.

4) Parameter Setting: The common settings for the algorithms are given as follows. The SBX and PM are used in all the algorithms. The crossover probability is set as 1.0 and the mutation probability is set as $1/D$, where D is the dimension of the decision space. The distribution index for mutation is set as 20. The distribution indices for crossover η_c are different in different algorithms. For θ-DEA, NSGA-III, NMPSO, and CPSO, η_c is set to 30. For HypE, η_c is set to 20. For MOEA/D-PaS and PICEA-g, η_c is set to 15. Among all the algorithms, if not otherwise stated, a population size $N = 100$ is adopted. In θ-DEA and NSGA-III, since the population size is related to the number of reference points, we adopt the integer closest to 100 as the population size, i.e., 86 and 66 for 5- and 10-objective problems, respectively, following [8]. The number of reference points are set as 85 and 65 for 5- and 10-objective problems, respectively. In CPSO, the population size of each

TABLE III
PERFORMANCE COMPARISON BETWEEN CPSO AND OTHER ALGORITHMS IN TERMS OF HV VALUE ON DTLZS AND WFGS

Problem	θ-DEA	NSGA-III	NMPSO	MOEA/D-PaS	HypE	PICEA-g
DTLZ	5/6/3	5/6/3	6/5/3	1/5/8	12/0/2	13/0/1
WFG	9/2/7	5/0/13	5/4/9	9/1/8	18/0/0	13/1/4
#of+/=/−	14/8/10	10/6/16	11/9/12	10/6/16	30/0/2	26/1/5

swarm is set as 100/M. The number of reference points used in CPSO is set as 85 and 110 for 5- and 10-objective problems, respectively.

Besides the parameters mentioned above, θ-DEA, NMPSO, MOEA/D-PaS, HypE, PICEA-g, and CPSO have their own parameters. The penalty parameter in θ-DEA is set at 5 following [18]. In NMPSO, the inertia weight ω is set in the range of [0.1, 0.5] and the acceleration coefficients c_1, c_2, and c_3 are randomly generated in [1.5, 2.5] following [37]. In MOEA/D-PaS, the neighborhood size T is set as $0.1 \times N$ and the replacement size is set as $0.2 \times T$ [35]. In HypE, the number of sample points for HV value estimation is set as 10 000 following [18]. In PICEA-g, the number of goals is set as $M \times 100$ [29]. In the proposed CPSO, we adopt the common configurations that ω linearly decreases from 0.9 to 0.4 and c_1 and c_2 are set to 1.49 [12], [48]–[54]. The archive size NA is set as 100.

In the experiments, NSGA-III adopts the widely used implementation jMetal (http://jmetal.github.io/jMetal/) and all the other compared algorithms adopt the source codes provided by the original authors. For fair comparisons, all results reported are obtained based on 30 independent runs with 100 000 function evaluations (FEs).

B. Experimental Results

In this section, detailed results are presented and compared. Tables II and III report the experimental results on IGD and HV metrics. In order to provide a statistical conclusion, we use Wilcoxon's rank-sum test between CPSO and each competitor algorithm at a 0.05 significance level on each problem instance. Three symbols $+, =$, and $-$, indicate that CPSO is significantly better, equivalent to, or significantly worse than the competing algorithm, respectively. The number of instances on each significance case $(+/=/-)$ is summarized at the last three rows of the tables. For clarity, the results of the best algorithm are marked in boldface and the results of the second best algorithm are marked with *italic*.

1) DTLZ Problems: Table II presents the IGD results of different algorithms on the 14 DTLZ problems with 5 and 10 objectives. We can see that CPSO performs the best or second best on most cases. Both CPSO and NSGA-III rank the first on 3 out of 14 instances, and MOEA/D-PaS obtains the best value on 4 instances. Therefore, the performance of CPSO is comparable with these algorithms and it is much better than θ-DEA, NMPSO, PICEA-g, and HypE, for that θ-DEA, NMPSO and PICEA-g scale the best on only one instance, while HypE does not perform the best on any instance. If taking both the best results (indicated by boldface) and the second best results (indicated by *italic*) into consideration, both CPSO and θ-DEA do well on 6 out of the 14 instances, outperforming all the other 5 competitors. Moreover, from the results of the significance test, CPSO is significantly superior to θ-DEA, NSGA-III, NMPSO, MOEA/D-PaS, HypE, and PICEA-g on 9, 8, 7, 5, 14, and 11, while worse than them on 5, 6, 6, 6, 0, and 3 instances, respectively. Therefore, CPSO has an overall better performance than these competitors in terms of the IGD metric.

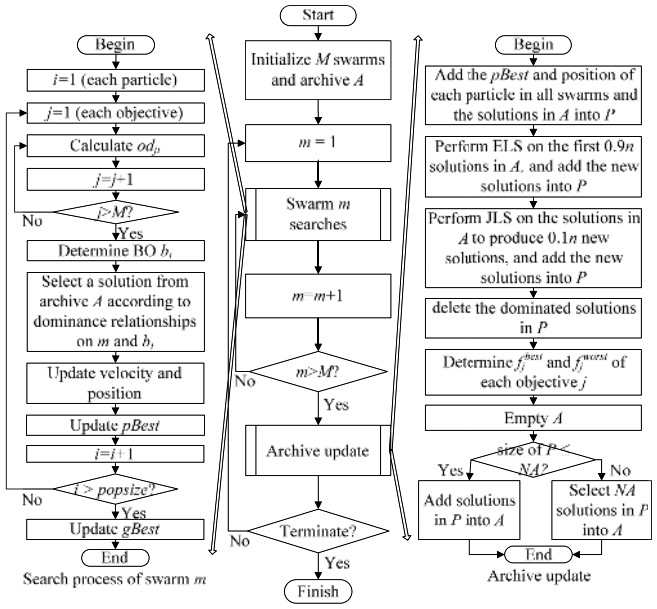

Fig. 3. Flowchart of the CPSO.

In details, Table II shows that the proposed CPSO generally performs well for all the problem instances. First, DTLZ1 is a difficult MaOP with many local PFs that trapped many existing MaOP algorithms. Both of NSGA-III and θ-DEA perform well on the 10-obejctive instance due to the strong guidance of the reference points in these two algorithms. However, CPSO performs the best on the 5-objective instance and ranks third on the 10-objective instance. This shows that CPSO is able to jump out of local PFs and find the global PF. As for DTLZ2 with spherical PF and DTLZ4 with nonuniform distributed PF, CPSO works the best on their 10-objective instances. This shows that CPSO has a strong search ability on different PF shapes, owing to the coevolution of the multiple swarms. In contrast, MOEA/D-PaS presents poor performance on both DTLZ2 and DTLZ4 since it wastes computational resources on finding a suitable scalarizing method. Similarly, PICEA-g performs poorly due to the lack of proper preference goals. For DTLZ3 with many local PFs, CPSO shows a mediocre performance. However, CPSO still performs better than PICEA-g on the 5-objective DTLZ3 and HypE on both 5-objective and 10-objective instances. For DTLZ5 and DTLZ6 with degenerate PFs, PICEA-g, and MOEA/D-PaS perform well since they can adapt to the PF geometry of the problems. In contrast, both θ-DEA and NSGA-III perform poorly since the reference points consider a higher-dimensional approximate front as the obtained front and wrongly depict the PF geometry. However, although

CPSO also uses reference points, CPSO ranks second for both DTLZ5 and DTLZ6 except the 5-objective instance of DTLZ5 because the swarms in CPSO coevolve under the guidance of not only the archive but also their own POs. Hence, CPSO can deal with the redundant information and ease the misdirection of the reference points on the PF geometry in degenerate cases. For the DTLZ7 with 2^{M-1} disconnected Pareto-optimal regions in search space, θ-DEA and NSGA-III work poorly especially on the 10-objective instance. In contrast, CPSO ranks third for the 10-objecitve instance of DTLZ7. This shows that CPSO can maintain diversity on different PF regions. In summary, CPSO can effectively solve different kinds of DTLZ problems, and can even achieve better performance on the instances with a larger number of objectives.

The detailed results of HV metric are reported in Table S.I, in the supplementary material, due to space limit and the significance test results are summarized in Table III. The results show that CPSO is significantly better than or equivalent to all the other algorithms on DTLZ1, DTLZ2, and DTLZ4. Moreover, CPSO performs significantly better than θ-DEA, NSGA-III, NMPSO, MOEA/D-PaS, HypE, and PICEA-g on 5, 5, 6, 1, 12, and 13 instances, while performing worse on 3, 3, 3, 8, 2, and 1 instances, respectively.

The results on IGD and HV metrics show that CPSO presents both strong diversity and good convergence on the DTLZ problems. To intuitively illustrate the performance of each algorithm, Fig. 4 shows the distributions of the solutions obtained by CPSO and the six compared algorithms on the 10-objective DTLZ1. The solutions in each figure are obtained in one random run out of the 30 independent runs. Due to the high objective dimensions, we plot each solution by connecting the values of each objective. On the PF of DTLZ1, each objective can reach a maximum value of 0.5. From Fig. 4(a)–(c), it can be seen that CPSO, θ-DEA, and NSGA-III are able to converge to the PF and find solutions well distributed along the whole PF. This is because the multiple swarms of CPSO can approximate different parts of the PF to maintain diversity and the BOL strategy improves convergence to the PF; the θ-dominance simultaneously considers convergence and diversity and thus it enables θ-DEA to find well-converged and well-diversified solutions; and the reference points in NSGA-III provide a strong guidance for the population toward the PF. In contrast, NMPSO approximates only some areas of the PF. The reason may be that a single population is very likely to concentrate only on some areas. Likewise, MOEA/D-PaS obtains optima only on some subproblems and converges to some parts of the PF. Similarly, the solutions of HypE and PICEA-g only bias toward a few objectives. The poor performance of HypE may be due to the inaccurate estimation of HV value for solution selection. In PICEA-g, the preferences may make negative effect on guidance and cause diversity loss.

2) WFG Problems: The IGD results of all algorithms on the WFG problems are presented in Table II. We can see that CPSO performs the best on 7 instances, followed by PICEA-g with 5 instances. NSGA-III and MOEA/D-PaS work the best on 3 and 2 instances, respectively. θ-DEA performs the best on only one instance, while NMPSO and HypE do not obtain the best IGD value on any instance. Moreover, according to the significance test, CPSO performs significantly better than θ-DEA, NSGA-III, NMPSO, MOEA/D-PaS, HypE, and PICEA-g on 11, 13, 15, 12, 15, and 8 out of 18 instances, while worse than them on only 3, 4, 3, 5, 1, and 8 instances, respectively. Thus, CPSO is significantly better than other algorithms in terms of IGD metric.

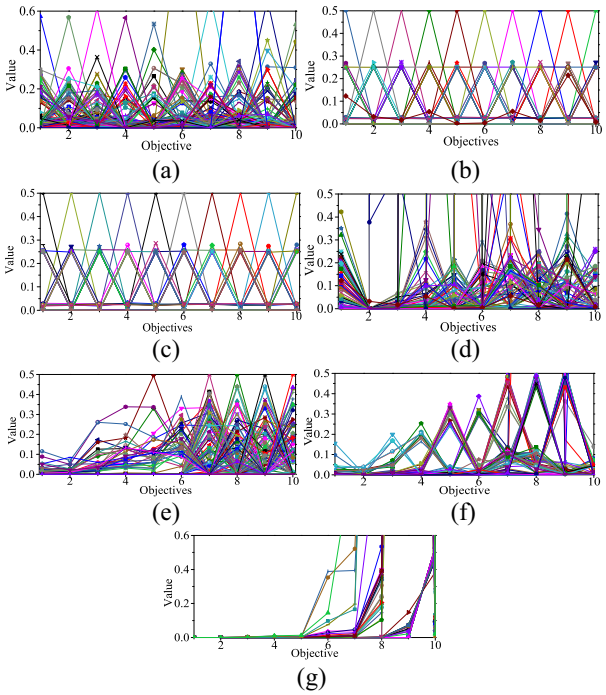

Fig. 4. Distributions of the solutions obtained by seven algorithms on DTLZ1 with ten objectives. (a) CPSO. (b) θ-DEA. (c) NSGA-III. (d) NMPSO. (e) MOEA/D-PaS. (f) HypE. (g) PICEA-g.

For more detail from Table II, CPSO performs well on almost all problems. More importantly, the performance of CPSO becomes better when the number of objectives increases from 5 to 10. This shows the strong global search capacity of CPSO in high-dimensional objective space of scaled problems. For WFG1 and WFG7 with separable and unimodal characteristics, CPSO is beaten by MOEA/D-PaS on WFG1 with mixed PF but outperforms MOEA/D-PaS on WFG7 with regular PF and ranks first on the 10-objective WFG7. For nonseparable and unimodal problems WFG3 and WFG6, PICEA-g performs the best on WFG3 with a degenerate PF due to its adaptation ability to PF geometry but fails on WFG6 with regular PF. In contrast, CPSO not only ranks second or third on different instances of WFG3 but also works well on WFG6. For the hard nonseparable problem WFG8, CPSO performs the best on the 10-objective instance. These show that CPSO is able to deal with both separable problems and nonseparable problems. For the deceptive problem WFG5, although PICEA-g performs the best on the 5-objective instance, CPSO works the best on the 10-objective instance. For WFG2 and WFG4 with multimodal characteristics, there are many local PFs. CPSO is beaten by PICEA-g and MOEA/D-PaS on WFG2 but outperforms them on WFG4 which has larger "hill sizes." For example,

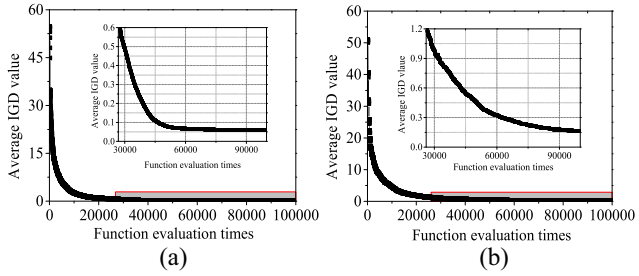

Fig. 5. Average IGD values on 30 runs of DTLZ1 with (a) five objectives and (b) ten objectives.

CPSO ranks second and first on the 5-objective and 10-objective instances of WFG4, respectively. Therefore, CPSO has stronger ability than others to jump out of local optima. For the difficult problem WFG9 combined with nonseparable, multimodal, deceptive, and bias characteristics, CPSO still performs the best. In summary, CPSO is able to solve not only simple problems, but also difficult problems with different characteristics.

For WFG4-WFG9 with regular PFs, θ-DEA, and NSGA-III perform well on most cases due to the strong guidance of the reference points toward the PFs. However, for WFG1-WFG3 with irregular PFs, NMPSO, MOEA/D-PaS, and PICEA-g perform better due to their adaptive abilities to the PF geometry. It is interesting to find that θ-DEA and NSGA-III also perform well on WFG1 since the PF of WFG1 are regular in each segment and the reference points can provide good guidance for the population toward the PF. In contrast, CPSO not only performs the best on most 10-objective instances of WFG4-WFG9 but also ranks second or third on WFG2 and WFG3. The coevolution of multiple swarms enables CPSO to ease the misdirection of the wrongly depicted PF geometry caused by the reference points on problems with irregular PFs while following the guidance of the reference points well on the problems with regular PFs. In addition, the SR component also enhances the global search ability of CPSO.

For the HV metric, Table S.I, in the supplementary material, shows that CPSO performs the best on 4 out of 18 instances. Meanwhile, NSGA-III, NMPSO, and MOEA/D-PaS work the best on 5, 5, and 2 instances, respectively, while both θ-DEA and PICEA-g perform the best on only 1 instance. Among all the 18 WFG instances, CPSO performs significantly better than or is equivalent to all competitors except NSGA-III on nearly or more than half of instances, while it is significantly worse than θ-DEA, NSGA-III, NMPSO, MOEA/D-PaS, and PICEA-g on 7, 13, 9, 8, and 4 instances, respectively. Moreover, CPSO performs better than HypE on all instances. Therefore, CPSO is competitive with the benchmark models.

Comprehensively considering the results of IGD and HV metrics, CPSO generally presents stronger diversity and better convergence than the competitors on the WFG problems. For example, although CPSO obtains worse IGD values than PICEA-g on WFG5 and WFG7, it gets better HV values on all these problem instances. The reversed performance on IGD and HV metrics may be due to the relatively reversed concentration on diversity and convergence. To intuitively show the algorithm performance, Fig. S.1, in the supplementary material, illustrates the distributions of the obtained solutions of different algorithms in one run on the 10-objective WFG8. From Fig. S.1, in the supplementary material, we can see that CPSO finds solutions well distributed on the margins as well as the center of the PF. In contrast, both θ-DEA and NSGA-III obtain well-converged but poorly diversified solutions. NMPSO gets nonuniform solutions that concentrate more on the objectives with large orders of magnitude. Thus, even though the solutions accumulate to a larger HV value, NMPSO still gets a poor IGD value due to the loss of diversity. MOEA/D-PaS converges to some marginal parts of the PF and bias toward only a few objectives. The small number of converged solutions cannot accumulate to a large HV. Similar situations also occur on HypE and PICEA-g. HypE gets solutions well converged but poorly diversified. PICEA-g gets nonconvergent or nondiversified solutions. Thus, both HypE and PICEA-g receive a large IGD and a small HV.

C. Convergence Analysis of CPSO

In this section, we discuss the convergence of CPSO. To take DTLZ1 as an example, the convergence curves of IGD value are plotted in Fig. 5. As shown in Fig. 5(a), on the 5-objective instance, the IGD value decreases quickly to 0.52 in 30 000 FEs, further reduces to 0.061 in 60 000 FEs and finally reaches 0.0601. For the harder 10-objective instance shown in Fig. 5(b), the IGD value decreases to 0.3 in 60 000 FEs and reaches 0.1550 at last. These show that CPSO can converge to the PF steadily.

To further investigate the search behavior of CPSO, we report the swarm information per 100 generations. Since CPSO converges faster on the 5-objective instance, we take the 5-obejective instance as an example. To maintain the logic of the search behavior, only one run is taken randomly. There are totally 500 generations. The pBest and position X of all particles in all swarms are presented once per 100 generations in Fig. 6(a)–(e), and the solutions in the output archive are presented in Fig. 6(g). The pBest and X of all swarms in each generation are combined and the nondominated solutions among them are also presented in Fig. 6(f). From Fig. 6(a)–(e), we can see that the swarms gradually converge to the PF. In the early stage, the particles have poor values on almost all objectives and the extreme points are far from the PF. Through swarm coevolution, different objective values of all particles become smaller. By the 400th generation, the maximum values of each objective have been reduced from 300 to 0.4. The swarms are significantly closer to the PF. It is interesting to find that there are no extreme points of the PF after the 400th generation as shown in Fig. 6(d) and (e). This is because the extreme points obtained earlier are pushed to the center of the PF since their BO are optimized by the BOL strategy gradually. Thus, different objective values except the corresponding PO become closer in the later stage. The swarms move along the PF and maintain stable. This can be confirmed by Fig. 6(f) that all the nondominated solutions obtained by the swarms along the search process can well cover the whole PF. Nevertheless, with the help of archive update (SR and SP

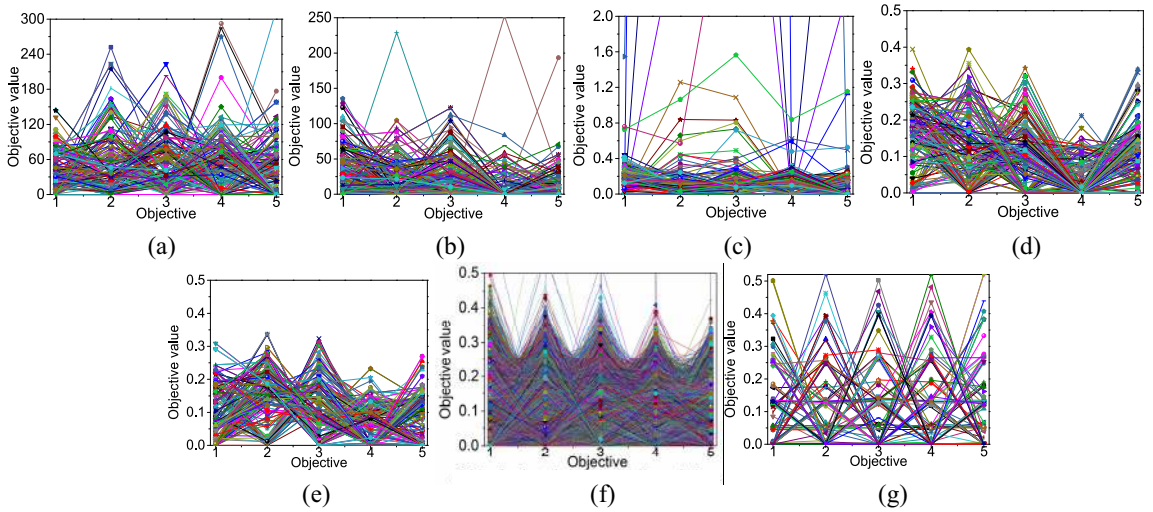

Fig. 6. Distributions of solutions obtained by CPSO on DTLZ1 with five objectives. (a)–(e) *p*Best and position *X* of all particles in all swarms per 100 generations. (f) Nondominated solutions of all the *p*Best and *X* obtained by all the swarms along the whole search process. (g) Solutions in the output archive.

components), the good solutions found by the swarms are preserved and the missing areas of the PF are filled as shown in Fig. 6(g).

D. Contributions of Each Component in CPSO

In CPSO, there are three components: 1) MPMO component with BOL strategy; 2) SR component with ELS and JLS operators; and 3) SP component. To observe the contributions of each component, we perform experiments of five CPSO variants with different combinations of the components, i.e., MPMO, SR, MPMO+SP, SR+SP, and MPMO+SR. Since SP does not generate solutions, it cannot be tested independently. Therefore, MPMO, SR, MPMO+SP, SR+SP, and MPMO+SR are CPSO variants with only MPMO, only SR, only both MPMO and SP, only both SR and SP, and only both MPMO and SR, respectively. For the variants without SP, the archive size is set as infinite and all the nondominated solutions are stored. The results in terms of IGD metric are summarized in Table IV, and the detail is presented in Table S.II, in the supplementary material. The number of the best cases of each variant is presented in Table V. From Tables IV, V, and S.II, in the supplementary material, it can be seen that CPSO outperforms all its variants. CPSO, MPMO+SP, SR+SP, MPMO+SR, MPMO, and SR perform the best on 12, 2, 7, 2, 6, and 3 cases, respectively. Moreover, CPSO is significantly better than MPMO+SP, SR+SP, MPMO+SR, MPMO, and SR on 24, 15, 23, 24, and 25 cases while is beaten by them on only 4, 8, 6, 8, and 3 out of 32 cases, respectively.

The MPMO performs worse than CPSO on most cases but is the best on 6 instances, i.e., 10-objective instances of DTLZ5, DTLZ6, WFG1, and WFG2, and also 5-objective instances of WFG2 and WFG3. Together with SP, the performance of MPMO (i.e., the MPMO+SP variant) improves on most cases with regular PFs but degenerates on DTLZ5-DTLZ7 and WFG1-WFG3 with irregular PFs. This is because the SP enhances the diversity of the archive such that the archive provides strong guidance for the swarms on regular PFs. However, for DTLZ5-DTLZ7 and WFG1-WFG3 with irregular PFs, SP may provide misleading information and cause the waste of computing resources on improper PF geometry. Specially, on the 5-objective WFG1 and 10-objective WFG3, MPMO+SP performs better than MPMO since the guidance of well-diversified archive solutions accelerates the algorithm convergence.

The SR performs the best on three 5-objective instances of DTLZ4, DTLZ7, and WFG5, but performs poorly on other instances, especially on the 10-objective instances. Since the JLS combines good information of different solutions while the ELS helps to jump out of local optima, SR can be seen as a combination of global search and local search. However, although SR can handle some 5-objective instances, it still faces difficulties on 10-objective instances. With SP, the performance of SR (i.e., the SR+SP variant) improves greatly especially on the 10-objective instances. The SP helps SR to avoid the repeated search of the same areas. Nevertheless, the SR+SP variant is still beaten by CPSO on most cases. Thus, the global search ability of SR is not efficient enough.

TABLE IV
PERFORMANCE COMPARISON BETWEEN CPSO AND ITS VARIANTS MPMO+SP, SR+SP, MPMO+SR, MPMO, AND SR IN TERMS OF IGD VALUES ON DTLZS AND WFGS

Problem	Two components			Only one component	
	MPMO+SP	SR+SP	MPMO+SR	MPMO	SR
DTLZ	12/2/0	7/3/4	11/1/2	12/0/2	12/0/2
WFG	12/2/4	8/6/4	12/2/4	12/0/6	13/4/1
#of+/=/−	24/4/4	15/9/8	23/3/6	24/0/8	25/4/3

TABLE V
NUMBER OF BEST CASES OF CPSO AND ITS VARIANTS MPMO+SP, SR+SP, MPMO+SR, MPMO, AND SR IN TERMS OF IGD VALUES ON DTLZS AND WFGS

CPSO	Two components			Only one component	
	MPMO+SP	SR+SP	MPMO+SR	MPMO	SR
12	2	7	2	6	3

Combining the MPMO with the SR, the obtained MPMO+SR variant still performs poorly and it works the best on only 2 instances, i.e., 5-objective WFG4 and 10-objective WFG6. Compared with SR, the MPMO+SR variant performs better on most 10-obejective instances, i.e., DTLZ1-DTLZ5, DTLZ7, WFG1-WFG3, WFG5, WFG6, and WFG9. This shows the strong global search ability of MPMO in high-dimensional objective space. Compared with MPMO, the MPMO+SR variant performs better especially on problems with regular PFs. The SR helps the MPMO to jump out of local optima and fill the missing areas of the PF that are missed by the swarms. Nevertheless, without SP, the rapid increase of the nondominated archived solutions causes the decrease of selection pressure and consequently the ability of the BOL strategy is limited. By adding the SP into MPMO+SR, the resultant MPMO+SR+SP (which is actually CPSO) performs better on most cases. Thus, the combination of the three components performs the best.

In general, MPMO is the basis of CPSO, SR assists to jump out of local PFs and reach out to the missing areas of the PF, and SP stores good solutions for information sharing among the swarms. Without the help of SR and SP, the MPMO is not efficient and is easily trapped in local optima. Likewise, without the help of MPMO and SP, the SR has insufficient global search ability to solve MaOPs especially when the objective number is large. For the SP, on the one hand, the SP enables CPSO to keep the good solutions found along the search process; on the other hand, the SP enables the archive to be a communication environment for the swarms. Thus, all the three components play different yet significant roles in CPSO.

E. Benefit of BOL Strategy in MPMO Component

The BOL strategy promotes the coevolution of the swarms. In this section, we consider three CPSO variants for comparisons, i.e., CPSO-noCP, CPSO-rand, and CPSO-pgbestra, to verify the effects of the CP evaluation in the BOL strategy, the BO optimization in the BOL strategy, and the BOL strategy, respectively. They differ from CPSO only in velocity update. CPSO-noCP selects a solution randomly when the two solutions are nondominated on PO and BO during archive solution selection in the BOL strategy. CPSO-rand adopts a rand strategy that implements velocity update by learning from *p*Best and a solution randomly selected from archive. CPSO-pgbestra adopts the learning strategy pgbestra proposed in [12] that implements velocity update by learning from the *p*Best, the *g*Best of the swarm, and a solution randomly selected from archive. Since the JLS combines solutions from different swarms, it may affect the observation of the coevolution effect of the BOL strategy. Thus, we also compare their corresponding versions without JLS, named noJLS, noJLS-noCP, noJLS-rand, and noJLS-pgbestra, respectively.

The detailed results of CPSO and its variants with JLS on IGD and HV metrics are reported in Table S.III, in the supplementary material, and the significance test results are summarized in Table VI. The results of noJLS and other variants without JLS in terms of IGD and HV metrics are summarized in Table VII and the detail is presented in Table S.IV, in the supplementary material.

The results in Table S.III, in the supplementary material show that CPSO converges faster than CPSO-noCP, CPSO-rand, and CPSO-pgbestra in most cases. Moreover, CPSO gets the best IGD values on 13 out of 32 cases and the best HV values on 22 out of 32 cases. Thus, the CPSO with the BOL strategy performs the best. The advantage of the BOL strategy is more obvious in the comparisons between noJLS and its variants as shown in Table S.IV, in the supplementary material. The noJLS achieves the best IGD values on 15 out of 32 cases and the best HV values on 25 out of 32 cases.

For the CP evaluation in the BOL strategy, CPSO performs better than CPSO-noCP on 7 cases while worse on only 5 out of 32 cases in terms of the IGD metric, and also works better than CPSO-noCP on 12 cases while worse on only 2 out of 32 cases in terms of the HV metric. These show that, between two nondominated solutions, selecting a solution with better CP as archive solution helps to speed up convergence. Similar phenomenon appears between the comparison between noJLS and noJLS-noCP.

For the BO optimization in the BOL strategy, CPSO performs better than CPSO-rand on 10 cases while worse on only 5 out of 32 cases on IGD metric. The corresponding numbers of better and worse cases on HV metric are 14 and 5, respectively. Thus, the BOL strategy converges faster than the rand strategy. It is interesting to see that CPSO-rand gets better HV values than CPSO on 5 instances of the DTLZs since the random selection in the rand strategy enhances diversity and helps to obtain well-diversified solutions. When comparing the variants without the JLS (i.e., noJLS and noJLS-rand) in Table V, the BOL strategy still beats the rand strategy. For example, for DTLZs, noJLS obtains better IGD values than noJLS-rand on most instances and worse HV values on only one instance; and for WFGs, noJLS performs significantly better or at least equivalently on all instances on HV metric. In general, the BOL strategy performs better than the rand strategy and the BO optimization can accelerate convergence.

Next, we compare the BOL strategy with the pgbestra strategy [12]. Compared with CPSO-pgbestra, CPSO performs remarkably better on both IGD and HV metrics. It indicates the advantage of the BOL strategy. In fact, the BOL strategy may affect swarm diversity in two opposite directions: it tends to decrease diversity and improve convergence by moving particles closer to the solutions with good values on PO, but it is also possible to increase diversity by speeding up the optimization of different BO toward the PF. Nevertheless, the cooperation of the BOL strategy and the MPMO helps different swarms to approximate different parts of the PF, and thus the diversity and convergence can be improved simultaneously.

F. Benefit of ELS and JLS Operators in SR Component

In this section, the effects of two operators ELS and JLS in SR component are investigated. We consider three variants of CPSO, i.e., noELS, noJLS, and noELS-noJLS, and they differ from CPSO only in that noELS does not adopt the ELS, noJLS

TABLE VI
PERFORMANCE COMPARISON BETWEEN CPSO AND ITS VARIANTS WITH DIFFERENT LEARNING STRATEGIES IN TERMS OF IGD AND HV VALUES ON DTLZS AND WFGS

Problem	CPSO-noCP		CPSO-rand		CPSO-pgbestra	
	IGD	HV	IGD	HV	IGD	HV
DTLZ	5/9/0	1/11/2	5/8/1	0/9/5	6/7/1	2/10/2
WFG	2/11/5	11/7/0	5/9/4	14/4/0	4/10/4	14/4/0
#of+/=/−	7/20/5	12/18/2	10/17/5	14/13/5	10/17/5	16/14/2

TABLE VII
PERFORMANCE COMPARISON BETWEEN NOJLS AND ITS VARIANTS WITH DIFFERENT LEARNING STRATEGIES NOJLS-NOCP, NOJLS-RAND, NOJLS-PGBESTRA IN TERMS OF IGD AND HV VALUES ON DTLZS AND WFGS

Problem	noJLS-noCP		noJLS-rand		noJLS-pgbestra	
	IGD	HV	IGD	HV	IGD	HV
DTLZ	6/8/0	3/11/0	10/3/1	1/12/1	8/5/1	4/9/1
WFG	2/12/4	14/4/0	3/10/5	14/4/0	3/10/5	14/4/0
#of+/=/−	8/20/4	17/15/0	13/13/6	15/16/1	11/15/6	18/13/1

TABLE VIII
PERFORMANCE COMPARISON BETWEEN CPSO AND ITS VARIANTS NOELS, NOJLS, AND NOELS-NOJLS IN TERMS OF IGD AND HV VALUES ON DTLZS AND WFGS

Problem	noELS		noJLS		noELS-noJLS	
	IGD	HV	IGD	HV	IGD	HV
DTLZ	10/1/3	9/5/0	8/5/1	5/9/0	12/2/0	13/1/0
WFG	6/7/5	9/5/4	8/9/1	11/7/0	12/2/4	17/0/1
#of+/=/−	16/8/5	18/10/4	16/14/2	16/16/0	24/4/4	30/1/1

does not adopt JLS, and noELS-JLS does not adopt either ELS or JLS. The comparison results on IGD and HV metrics are summarized in Table VIII and the detail is reported in Table S.V, in the supplementary material.

From Table VIII, we can see that CPSO gets the best IGD values on 16 instances and the best HV values on 22 out of 32 instances, which are much better than the values obtained by noELS, noJLS, and noELS-noJLS since noELS, noJLS, and noELS-noJLS achieve the best IGD values on 7, 4, and 5 instances, and the best HV values on 10, 10, and 0 instances, respectively. Particularly, the noELS-noJLS performs the worst on most instances, showing that the two operators can improve algorithm performance.

The problems with multimodal characteristics, e.g., DTLZ1, DTLZ3, DTLZ6, DTLZ7, WFG2, WFG4, and WFG9, have many local PFs. Compared with noELS, CPSO obtains significantly better or at least equal IGD values on all these problems except the 5-objective WFG4, and performs better on all these problems except the 5-objective WFG2 in terms of the HV metric. Thus, the ELS does help the CPSO to jump out of local optima. Specifically, on the 5-objective instances of WFG4 and WFG7, CPSO gets significantly worse IGD values but better HV values. This may be their reversed concentration on diversity and convergence since CPSO consumes more computation on ELS to get more diverse solutions while noELS has more generations to move faster toward the PF. Overall, from the better performance of CPSO on most instances, it can be seen that the adoption of ELS can improve the performance of CPSO.

As for the JLS, from Table VIII, we can see that CPSO performs significantly better than or equivalently to noJLS on almost all problems in terms of both IGD and HV metrics. It shows that the JLS generates solutions on the areas of the PF that are missed by the swarms and hence CPSO can get well-diversified solutions to achieve better IGD and HV scores.

TABLE IX
PERFORMANCE COMPARISON BETWEEN CPSO AND ITS VARIANTS SP-RAND, SP-CD IN TERMS OF IGD AND HV VALUES ON DTLZS AND WFGS

Problem	SP-rand		SP-CD	
	IGD	HV	IGD	HV
DTLZ	14/0/0	12/1/1	10/0/4	8/3/3
WFG	14/1/3	16/0/2	8/2/8	10/2/6
#of+/=/−	28/1/3	28/1/3	18/2/12	18/5/9

G. Benefit of Reference-Point-Based Selection Method in SP Component

In this section, we investigate the influence of the reference-point-based selection method in SP component for archive update. As the number of objectives increases, the number of nondominated solutions increases rapidly. The SP is expected to control archive size and simultaneously ensure solution diversity. We compare CPSO with two variants SP-rand and SP-CD, where SP-rand is a CPSO variant that randomly selects *NA* solutions from the nondominated solutions for updating archive, and SP-CD selects *NA* solutions according to the CD metric proposed in NSGA-II [6]. The results are summarized in Table IX and the detail is reported in Table S.VI, in the supplementary material. CPSO performs better than SP-rand and SP-CD on 28 and 18 out of 32 instances, respectively, on both of IGD and HV metrics. In contrast, SP-rand performs better than CPSO on only three instances, i.e., two instances of WFG1 and one 5-objective instance of WFG2, since a random selection easily misses some areas of the PF. For SP-CD, it performs better than CPSO on the 5-objective instances of DTLZ2, DTLZ4, WFG2, WFG4, WFG5, and WFG9, but worse than CPSO on their 10-objective instances. This shows that CD metric fails to estimate solution diversity in a high-dimensional objective space. It is interesting to observe that SP-CD performs well on DTLZ7 and WFG1 with irregular PFs, but poorly on DTLZ5, DTLZ6, and WFG3 who also have irregular PFs. This shows that the CD metric may be able to track the PF geometry of the problems and can handle some problems with irregular PF. In general, compared with random selection and CD metric, the reference-point-based selection method can handle most of problems with regular or irregular PFs and has advantages especially on 10-objective instances.

H. Influence of Parameter Settings

In this section, we investigate the influence of the parameters: population size, inertia weight ω, and acceleration coefficients c_i. DTLZ3 and WFG2 are chosen from DTLZs and WFGs as examples since these two problems include most typical features among the two test suits. Note that the parameters except the tested one remain the same as described in Section IV-A.

1) Population Size: To demonstrate the influence of the population size in each swarm, experiments are conducted with different population size for each swarm of 10, 20,

30, 40, and 50. The average IGD results are illustrated in Fig. S.2(a) and (b), in the supplementary material. CPSO gets similar results with population size in the range of [20, 40] on WFG2 while works well in the range of [10, 20] on DTLZ3. On 10-objective instances, a small population size also leads to better results on DTLZ3 while on WFG2 the CPSO performs better with a population size of 10 or in the range of [30, 50]. Although a large population size increases swarm diversity, the degeneration of convergence still causes the poor results on DTLZ3. Therefore, we set the population size as 20 for 5-objective and 10 for 10-objective problems. Therefore, the total population size is 100 for all the problems.

2) Inertia Weight ω: We perform experiments on DTLZ3 and WFG2 with ω set as 0.1, 0.3, 0.5, 0.7, and 0.9, respectively. From Fig. S.2(c), in the supplementary material, the results show that the range of [0.1, 0.5] for the 5-objective instance and the range of [0.5, 0.7] for the 10-objective instance are promising on DTLZ3. For WFG2, the 5-objective instance is less sensitive to ω value while the 10-objective instance prefers a relatively large value in the range of [0.7, 0.9]. This is because a large value is beneficial to global search in a high-dimensional objective space. On the contrary, a small value helps the algorithm to focus more on local areas in a low-dimensional objective space. Thus, we set the value of ω linearly decreasing from 0.9 to 0.4.

3) Acceleration Coefficients c_i: The acceleration coefficients c_i in the BOL strategy are tested in the range of [0.5, 2.0] with step sizes of 0.1 as well as the value of 1.49, with totally $17 \times 17 = 289$ combinations. The results are illustrated in Fig. S.2(d)–(g), in the supplementary material. In each subfigure, the projection of the 3-*D* surface is presented on the bottom of the 3-*D* space. For the 5-objective DTLZ3, the combination of the ranges in the lower right corner seems poor, indicating that excessively large values (e.g., $c_1 > 1.8$ and $c_2 > 1.6$) are not suitable. However, for the 10-objective DTLZ3, the combination of the ranges in the lower left corner seems poor, while a range of [0.5, 1.8] for c_1 and a range of [1.0, 2.0] for c_2 are preferred. For the 5-objective WFG2, the algorithm is less sensitive to c_1 and performs well with c_2 in the range of [1.0, 2.0]. For the 10-objective WFG2, a range of [0.5, 1.8] for c_1 and a range of [0.9, 1.6] for c_2 perform well while the combination of two small values or two large values performs poorly. This shows that a relatively medium value for c_1 and c_2 benefits exploitation and faster convergence. From these four instances, we can see that the c_i values are indeed problem dependent and should be carefully considered. Generally speaking, the performance of CPSO is less sensitive to c_1 but more dependent on c_2. CPSO can work well in a relatively large range of [0.5, 1.8] for c_1 but in a relatively small range of [1.0, 1.6] for c_2 on all the tested instances. Specially, a too small (i.e., [0.5, 0.9]) or too large value (i.e., [1.7, 2.0]) of c_2 may lead to less satisfactory results in some cases. The former may cause false convergence due to the lack of sufficient information to optimize the other objectives, while the latter may cause excessively fast convergence and get the deception of local optima. Therefore, a reasonable setting of c_1 and c_2 helps to enhance the global search ability of CPSO.

4) Combination Test: Since population size, ω, and c_i interact with each other, we also investigate their influence together on various combinations. According to the above results, we select values of 10, 20, and 30 for population size, 0.1, 0.5, and 0.9 for ω, and 1.0, 1.49, and 2.0 for c_i to construct different combinations. Note that c_1 and c_2 are set to the same values in this test. These values are selected due to the representativeness of their combinations according to the above results. The results are reported in Table S.VII, in the supplementary material. From Table S.VII, in the supplementary material, we can see that the performance under different parameter combinations is consistent with the above independent tests. A population size of 20 can obtain relatively good performance on 5-objective instances of both DTLZ3 and WFG2 whilst a population size of 10 is preferred by the 10-objective instances. For ω, the values of 0.1 and 0.5 are preferred by DTLZ3 while the values of 0.5 and 0.9 are preferred by WFG2. For c_i, the values of 1.0, 1.49, and 2.0 perform well on different instances.

V. Conclusion

When solving MaOPs, traditional MOEAs face challenges in terms of diversity and convergence. On the one hand, coevolutionary mechanisms with multiple populations are beneficial to maintain diversity for finding solutions well-distributed on the whole PF. On the other hand, the poorly performed objectives of a solution often become a bottleneck for the solution to converge toward PF. These factors motivate our consideration of CPSO, which incorporates the MPMO coevolutionary framework for diversity enhancement and a novel BOL strategy to optimize BO for convergence improvement. Multiple swarms optimize different objectives concurrently and they coevolve with the BOL strategy toward PF.

Experiments have been conducted on DTLZ and WFG problems with various characteristics and different number of objectives. IGD and HV metrics are adopted to evaluate the algorithm performance. The experimental results show that, compared with several state-of-the-art algorithms for MaOPs, the proposed CPSO is promising. The proposed CPSO can find a set of well-diversified and well-converged nondominated solutions for MaOPs.

References

[1] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 2424–2431.

[2] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, Aug. 2014.

[3] J. Maltese, B. M. Ombuki-Berman, and A. P. Engelbrecht, "A scalability study of many-objective optimization algorithms," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 79–96, Feb. 2018.

[4] L. M. Antonio and C. A. Coello Coello, "Coevolutionary multi-objective evolutionary algorithms: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2017.2767023.

[5] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, Aug. 2015.

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

[7] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," in *Evolutionary Multi-Criterion Optimization* (LNCS 4403). Heidelberg, Germany: Springer, 2007, pp. 742–756.

[8] Z. N. He and G. Yen, "Many-objective evolutionary algorithm: Objective space reduction and diversity improvement," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 145–160, Feb. 2016.

[9] R. C. Purshouse and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 770–784, Dec. 2007.

[10] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[11] K. C. Tan, Y. J. Yang, and C. K. Goh, "A distributed cooperative coevolutionary algorithm for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 527–549, Oct. 2006.

[12] Z.-H. Zhan *et al.*, "Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems," *IEEE Trans. Cybern.*, vol. 43, no. 2, pp. 445–463, Apr. 2013.

[13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[14] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput.*, 1998, pp. 69–73.

[15] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[16] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2832640.

[17] K. Deb, M. Mohan, and S. Mishra, "Evaluating the-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions," *Evol. Comput.*, vol. 13, no. 4, pp. 501–525, 2005.

[18] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 16–37, Feb. 2016.

[19] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 721–736, Oct. 2013.

[20] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 348–365, Jun. 2014.

[21] X. Y. Zhang, Y. Tian, and Y. C. Jin, "A knee point-driven evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 761–776, Dec. 2015.

[22] J. Cheng, G. G. Yen, and G. Zhang, "A many-objective evolutionary algorithm with enhanced mating and environmental selections," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 592–605, Aug. 2015.

[23] B. Li, K. Tang, J. Li, and X. Yao, "Stochastic ranking algorithm for many-objective optimization based on multiple indicators," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 924–938, Dec. 2016.

[24] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. Parallel Prob. Solving Nat.*, 2004, pp. 832–842.

[25] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *Eur. J. Oper. Res.*, vol. 181, no. 3, pp. 1653–1669, Sep. 2007.

[26] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications," *Theor. Comput. Sci.*, vol. 425, no. 1, pp. 75–103, 2012.

[27] M. Fleischer, "The measure of Pareto optima applications to multiobjective metaheuristics," in *Proc. Evol. Multi Crit. Optim.*, Faro, Portugal, 2003, pp. 519–533.

[28] J. Bader and E. Zitzler, "HypE: An algorithm for fast hypervolume-based many-objective optimization," *Evol. Comput.*, vol. 19, no. 1, pp. 45–76, Mar. 2011.

[29] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 474–494, Aug. 2013.

[30] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, Oct. 2016.

[31] E. J. Hughes, "Multiple single objective Pareto sampling," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 2678–2684.

[32] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Adaptation of scalarizing functions in MOEA/D: An adaptive scalarizing function based multiobjective evolutionary algorithm," *Evolutionary Multi-Criterion Optimization, EMO 2009* (LNCS 5467). Heidelberg, Germany: Springer, 2009, pp. 438–452.

[33] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Simultaneous use of different scalarizing functions in MOEA/D," in *Proc. Genet. Evol. Comput.*, 2010, pp. 519–526.

[34] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 3–18, Feb. 2018.

[35] R. Wang, Q. Zhang, and T. Zhang, "Decomposition based algorithms using Pareto adaptive scalarizing methods," *IEEE Trans. Evol. Comput.*, vol. 20, no. 6, pp. 821–837, Dec. 2016.

[36] W. Hu, G. G. Yen, and G. Luo, "Many-objective particle swarm optimization using two-stage strategy and parallel cell coordinate system," *IEEE Trans. Cybern.*, vol. 47, no. 6, pp. 1446–1459, Jun. 2016.

[37] Q. Lin *et al.*, "Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 32–46, Feb. 2018.

[38] A. Britto, S. Mostaghim, and A. Pozo, "Iterated multi-swarm: A multi-swarm algorithm based on archiving methods," in *Proc. Genet. Evol. Comput.*, 2013, pp. 583–590.

[39] A. de Campos, A. T. R. Pozo, and E. P. Duarte, "Evaluation of gossip vs. broadcast as communication strategies for multiple swarms solving MaOPs," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 1499–1506.

[40] J. L. Matos and A. Britto, "Multi-swarm algorithm based on archiving and topologies for many-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 1877–1884.

[41] Y. G. Woldesenbet and G. G. Yen, "Dynamic evolutionary algorithm with variable relocation," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 500–513, Jun. 2009.

[42] C. Li, T. T. Nguyen, M. Yang, M. Mavrovouniotis, and S. Yang, "An adaptive multipopulation framework for locating and tracking multiple optima," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 590–605, Aug. 2016.

[43] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 694–716, Oct. 2015.

[44] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization* (Advanced Information and Knowledge Processing). London, U.K.: Springer, 2005, pp. 105–145.

[45] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Trans. Evol. Comput.*, vol. 10, no. 5, pp. 477–506, Oct. 2006.

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

[47] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms: A comparative case study," in *Proc. Parallel Prob. Solving Nat.*, 1998, pp. 292–301.

[48] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 84–88.

[49] M. Jiang, Y. P. Luo, and S. Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Inf. Process. Lett.*, vol. 102, no. 1, pp. 8–16, 2007.

[50] R. Poli and D. Broomhead, "Exact analysis of the sampling distribution for the canonical particle swarm optimiser and its convergence during stagnation," in *Proc. Genet. Evol. Comput.*, 2007, pp. 134–141.

[51] R. Poli, "Mean and variance of the sampling distribution of particle swarm optimizers during stagnation," *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 712–721, Aug. 2009.

[52] V. Gazi, "Stochastic stability analysis of the particle dynamics in the PSO algorithm," in *Proc. IEEE Int. Symp. Intell. Control*, 2012, pp. 708–713.

[53] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm variants: Standardized convergence analysis," *Swarm Intell.*, vol. 9, nos. 2–3, pp. 177–203, 2015.

[54] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm stability: A theoretical extension using the non-stagnate distribution assumption," *Swarm Intell.*, vol. 12, no. 1, pp. 1–22, 2018.

[55] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Reference point specification in inverted generational distance for triangular linear Pareto front," *IEEE Trans. Evol. Comput.*, to be published, doi: 10.1109/TEVC.2017.2776226.
[56] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired co-evolutionary algorithms using weight vectors," *Eur. J. Oper. Res.*, vol. 243, no. 2, pp. 423–441, 2015.

Xiao-Fang Liu (S'14) received the B.S. degree in computer science from Sun Yat-Sen University, Guangzhou, China, in 2015.

She is currently a Research Fellow with the South China University of Technology, Guangzhou, China. Her current research interests include artificial intelligence, evolutionary computation, swarm intelligence, and their applications in design and optimization, such as cloud computing resources scheduling.

Zhi-Hui Zhan (S'18–M'13–SM'18) received the bachelor's degree and the Ph.D. degree in computer science from the Department of Computer Science, Sun Yat-Sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the Doctoral Dissertation Award by the China Computer Federation Outstanding Dissertation and the IEEE Computational Intelligence Society Outstanding Dissertation, the Outstanding Youth Science Foundation from the National Natural Science Foundations of China in 2018, and the Wu Wen Jun Artificial Intelligence Excellent Youth Award from the Chinese Association for Artificial Intelligence in 2017. He is listed as one of the Most Cited Chinese Researchers in Computer Science.

Ying Gao received the bachelor's degree and the, master's degree in computer science from the Central South University of China, Changsha, China, in 1997 and 2000, respectively, and the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 2006.

She is currently a Professor with the School of Computer Science and Engineering, South China University of Technology. She has published over 30 papers in international journals and conferences. Her current research interests include computer vision, software architecture and network security.

Jie Zhang received the master's degree in computer science from China University of Mining and Technology, Xuzhou, China, in 1999.

She is currently an Associate Professor with the Beijing University of Chemical Technology, Beijing, China. Her current research interests include formal verification and PHM for power and embedded system design.

Ms. Zhang is a member of Chinese Institute of Electronics Embedded Expert Committee.

Sam Kwong (F'13) received the B.Sc. degree from the State University of New York at Buffalo, Buffalo, NY, USA, in 1983, the M.A.Sc. degree in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1985, and the Ph.D. degree from the University of Hagen, Hagen, Germany, in 1996.

From 1985 to 1987, he was a Diagnostic Engineer with the Control Data Canada, Bloomington, MN, USA, where he designed the diagnostic software to detect the manufacture faults of the VLSI chips in the Cyber 430 machine. He later joined the Bell Northern Research Canada, Ottawa, ON, Canada, as a Member of Scientific staff. In 1990, he joined the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, as a Lecturer, where he is currently a Professor with the Department of Computer Science. His current research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong was appointed as IEEE Distinguished Lecturer for IEEE Systems, Man and Cybernetics (SMC) Society in 2017. He has been the Vice President for IEEE SMC for conferences and meetings since 2014.

Jun Zhang (F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Changjiang Chair Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include computational intelligence, cloud computing, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits. He has published over 100 technical papers in the above areas.

Dr. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.