Multiple Populations for Multiple Objectives: A Coevolutionary Technique for Solving Multiobjective Optimization Problems

Zhi-Hui Zhan, *Student Member, IEEE*, Jingjing Li, Jiannong Cao, *Senior Member, IEEE*, Jun Zhang, *Senior Member, IEEE*, Henry Shu-Hung Chung, *Senior Member, IEEE*, and Yu-Hui Shi, *Senior Member, IEEE*

***Abstract*—Traditional multiobjective evolutionary algorithms (MOEAs) consider multiple objectives as a whole when solving multiobjective optimization problems (MOPs). However, this consideration may cause difficulty to assign fitness to individuals because different objectives often conflict with each other. In order to avoid this difficulty, this paper proposes a novel coevolutionary technique named multiple populations for multiple objectives (MPMO) when developing MOEAs. The novelty of MPMO is that it provides a simple and straightforward way to solve MOPs by letting each population correspond with only one objective. This way, the fitness assignment problem can be addressed because the individuals' fitness in each population can be assigned by the corresponding objective. MPMO is a general technique that each population can use existing optimization algorithms. In this paper, particle swarm optimization (PSO) is adopted for each population, and coevolutionary multiswarm PSO (CMPSO) is developed based on the MPMO technique. Furthermore, CMPSO is novel and effective by using an external shared archive for different populations to exchange search information and by using two novel designs to enhance the performance. One design is to modify the velocity update equation to use the search information found by different populations to approximate the whole Pareto front (PF) fast. The other design is to use an elitist learning strategy for the archive update to bring in diversity to avoid local PFs. CMPSO is comprehensively tested on different sets of benchmark problems with different characteristics and is compared with some state-of-the-art algorithms. The results show that CMPSO has superior performance in solving these different sets of MOPs.**

***Index Terms*—Coevolutionary algorithms, multiobjective optimization problems (MOPs), multiple populations for multiple objectives (MPMO), particle swarm optimization (PSO).**

I. Introduction

MULTIOBJECTIVE optimization problems (MOPs) have received considerable attention over the past several decades because of their significance in a large number of real-world applications [1]–[4]. As evolutionary computation (EC) algorithms have gained great success in single-objective optimization, many researchers try to extend the EC algorithms to MOPs [5], [6]. However, when using the multiobjective evolutionary algorithm (MOEA) to solve MOPs, the problem on how to select good individuals for the next generation arises. Since an MOP has multiple objectives that often contradict with each other, it is difficult to say whether one individual is better than another if it is better on one objective but is worse on another objective [7]. This is the fitness assignment problem encountered by MOEA researchers. As the EC algorithms derive from the concept of "survival of the fittest" in Darwin's natural selection law, it would cause search inefficiency if we cannot address the fitness assignment problem. Therefore, one of the most significant research topics in MOEA is to design a suitable method to assign an individual's fitness [8].

To overcome the fitness assignment difficulty in MOEA, various techniques like the objective aggregation technique, the objective alternate technique, and the Pareto-based technique have been proposed in the literature [5], [6] (more information is given in Section II-C). The objective aggregation technique sums up the objectives to a single objective by weights and then optimizes the formed single objective. However, this technique requires users to determine the weights for different objectives. Moreover, only one solution can be obtained in a run by using this technique. The objective alternate technique sorts out the objectives according to their importance and optimizes them alternately. However, the ordering of the objectives may affect the performance significantly, while determining the importance of different objectives is often problem dependent. The third popular technique is to apply Pareto dominance to rank the individuals and assign fitness to them. The domination rank technique may be helpful for approximating the Pareto front (PF). However, as Pareto dominance is a partial-order relation, it is difficult to select individuals for the next generation. Thus,

Manuscript received June 3, 2011; revised September 28, 2011; accepted June 29, 2012. Date of publication August 14, 2012; date of current version April 16, 2013. This work was supported in part by the National Science Fund for Distinguished Young Scholars under Grant 61125205, by the National Natural Science Foundation of China (NSFC) under Grant 61070004, by the NSFC Joint Fund with Guangdong under Key Project U0835002, and by the Scholarship Award for Excellent Doctoral Student Granted by the Ministry of Education, China. This paper was recommended by Associate Editor Y. S. Ong.

Z.-H. Zhan and J. Zhang are with the Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275, China, with the Key Laboratory of Digital Life, Ministry of Education, China, and with the Key Laboratory of Software Technology, Education Department of Guangdong Province, China.

J. Li and J. Cao are with The Hong Kong Polytechnic University, Kowloon, Hong Kong.

H. S.-H. Chung is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.

Y.-H. Shi is with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong–Liverpool University, Suzhou 215123, China.

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/TSMCB.2012.2209115

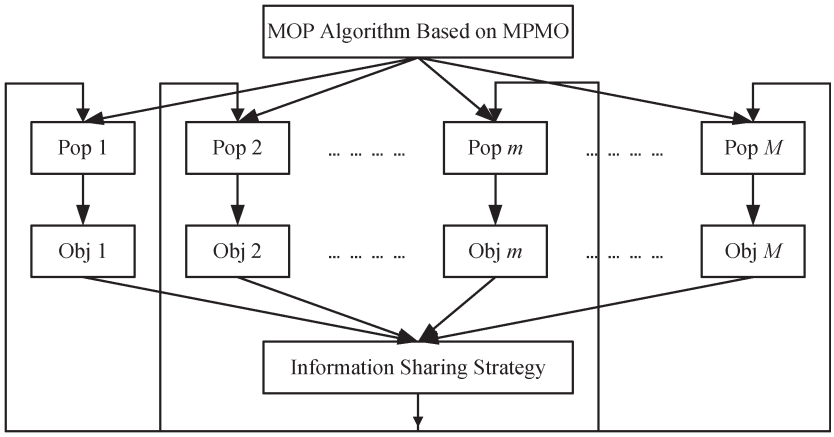


Fig. 1. Framework of MPMO-based algorithm for solving MOP.

the obtained solutions may still not spread along the whole PF if the selection operator cannot maintain sufficient diversity. Therefore, developing MOEA that can assign the individual's fitness easily and can also keep the diversity to approximate the whole true PF is still a challenging research topic in the MOEA community.

To address the fitness assignment problem and, at the same time, to design efficient MOEA for MOPs, a novel technique termed multiple populations for multiple objectives (MPMO) is proposed in this paper. The motivations of MPMO lie in that: as there are multiple objectives in the MOP, can we use multiple populations, instead of only one population, to optimize the MOP? Since it is difficult to consider all the objectives as a whole in one population, can we treat them separately in different populations? Therefore, the idea of the MPMO technique is novel and is different from the aforementioned techniques: instead of tackling all the objectives together as a whole by the same population, MPMO uses multiple populations to deal with multiple objectives, with each population being corresponded with only one objective and all populations cooperating to approximate the whole PF. Fig. 1 shows the framework of the MOP algorithm with M populations based on the MPMO technique to solve an MOP with M objectives. In every generation, the individuals in each population calculate all the objective functions like that in traditional MOP algorithms. However, when executing evolutionary operators like selection, the fitness value of an individual in the mth population is assigned by the mth objective function of the MOP, where $1 \leq m \leq M$. This way, the individuals would not be confused by different conflicting objectives anymore but are guided by the corresponding objective to search different regions of the PF. However, as each population focuses on optimizing one objective only, it may cause the problem that MPMO leads the individuals in each population to the margin of the corresponding objective, resulting in inefficient approximation of the whole PF. Therefore, another feature of MPMO is that it requires the algorithm to design an information-sharing strategy, as shown in Fig. 1. This way, different populations can share their search information and communicate with others through the information-sharing strategy to approximate the whole PF efficiently.

It should be noticed that the MPMO technique can be regarded as a coevolutionary mechanism for that it uses multiple populations to cooperatively solve problems. Coevolution has been accepted as a significant evolutionary mechanism in biology that has produced many intriguing adaptations and made significant contributions to biodiversity [9]. During the last two decades, the coevolutionary concept has also been used by scientists from the EC community [10]–[13]. The main idea in these studies is to decompose the problem into subproblems and to use multiple populations to optimize different subproblems cooperatively. In MPMO, the MOP naturally has multiple objectives, and there is no need to decompose the problem. Multiple populations optimize different objectives and cooperatively approximate the whole PF of the MOP. Therefore, we declare that MPMO is a coevolutionary technique for solving MOPs.

Being a general technique, it is straightforward to implement the MPMO algorithm that can accommodate any existing single-objective optimization algorithm in each population. In this paper, by considering that particle swarm optimization (PSO) [14] is a simple yet powerful global optimizer with very fast convergence speed, we adopt PSO for each swarm and design coevolutionary multiswarm PSO (CMPSO) as an instantiation of MPMO to solve MOPs. Based on the MPMO framework, CMPSO uses an external shared archive to implement the information-sharing strategy, with two novel designs being developed to enhance the algorithm performance. The first design is to modify the particle velocity updating equation with information obtained from an externally shared archive. The shared archive is used to store the nondominated solutions found by different swarms and is updated every generation. Not only the velocity and position of a particle are updated by considering its personal experience and its swarm's experience but also the experience fetched from the archive. Therefore, all the swarms can share their search information thoroughly via the shared archive. This is useful for the algorithm to accelerate the approximation of the whole PF. The second design is to utilize an elitist learning strategy (ELS) to update the archive in order to introduce sufficient diversity so as to avoid the occurrence of local PFs. This may be helpful for MOPs with multimodal objective functions or with complicated Pareto sets.

The innovations and advantages of the CMPSO algorithm are as follows.

1) Different from existing algorithms that treat an MOP as a whole by considering all the objectives together in a population, CMPSO is a coevolutionary multiswarm algorithm that is based on the MPMO technique. Each swarm is optimized by taking only one objective into account. Then, different swarms will cooperate with each other to approximate the whole PF efficiently.
2) As each swarm optimizes only one objective, CMPSO does not only avoid the difficulty of fitness assignment but also benefits from having each swarm with conventional PSO or improved PSO algorithms for optimizing a single objective.
3) As an external shared archive is used to store the nondominated solutions found in different swarms, each swarm can communicate with the others through the shared archive and can use the information provided by other swarms to approximate the whole PF fast.
4) As ELS is used in the archive update process, the algorithm is expected to have the ability to jump out of local PFs in MOP with multimodal objective functions.

The performance of CMPSO will be tested on different sets of benchmark problems with different characteristics in objective function, PF, and Pareto set. In order to demonstrate the advantages of CMPSO, we will compare it with not only state-of-the-art and modern MOEAs but also multiobjective PSOs (MOPSOs). The experimental results show that CMPSO has superior performance in solving MOPs.

The rest of this paper is organized as follows. In Section II, some background including the framework of the PSO algorithm, the definition of MOP, and the related work on MOP are reviewed. In Section III, the CMPSO algorithm to solve MOP is proposed. Section IV presents the experimental results, comparisons, discussions, and analysis of CMPSO. Finally, conclusions and future work are given in Section V.

II. Background

A. PSO

The PSO algorithm was first developed by Kennedy and Eberhart in 1995 as a global optimization technique [14]. When searching in a D-dimensional hyperspace, a population of N particles is used to represent solutions in the search space. The particle i is associated with a velocity vector $\boldsymbol{V_i} = [V_{i1}, V_{i2}, \ldots, V_{iD}]$ and a position vector $\boldsymbol{X_i} = [X_{i1}, X_{i2}, \ldots, X_{iD}]$ to indicate its current state, where D is the problem dimensionality. The historically best position of the particle i is denoted as $\boldsymbol{pBest_i} = [P_{i1}, P_{i2}, \ldots, P_{iD}]$. The best one of all the $\boldsymbol{pBest_i}$ is regarded as the globally best position $\boldsymbol{gBest} = [G_1, G_2, \ldots, G_D]$. The vectors $\boldsymbol{V_i}$ and $\boldsymbol{X_i}$ are initialized randomly within their corresponding ranges and are updated as (1) and (2) in every generation by the guidance of $\boldsymbol{pBest_i}$ and $\boldsymbol{gBest}$

$$V_{id} = \omega V_{id} + c_1 r_{1d}(P_{id} - X_{id}) + c_2 r_{2d}(G_d - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

where d is the index of the dimension.

The inertia weight ω is proposed to decrease linearly from 0.9 to 0.4 during the run time in order to balance global and local search abilities [15]. c_1 and c_2 are acceleration coefficients, which are commonly set to 2.0 or are adaptively controlled according to the evolutionary states [16]. r_{1d} and r_{2d} are two randomly generated values within the range of [0, 1] for the dth dimension. The velocity obtained by (1) is clamped in the range of $[-V_{\max,d}, V_{\max,d}]$, where $V_{\max,d}$ is a user-predefined positive value for a maximal velocity value. Also, the updated position obtained by (2) is clamped in the search range of $[X_{\min,d}, X_{\max,d}]$. In the literature, $V_{\max,d}$ is often set as 20% of the search range, i.e., $V_{\max,d} = 0.2(X_{\max,d} - X_{\min,d})$ [16].

After the velocity and position update, the new position is evaluated, and it will replace $\boldsymbol{pBest_i}$ if it has a better fitness. Moreover, $\boldsymbol{gBest}$ is replaced by $\boldsymbol{pBest_i}$ if $\boldsymbol{pBest_i}$ has a better fitness. All the particles perform the same operations, and the algorithm runs iteratively until a termination criterion is met.

B. MOPs

A minimization MOP can be described as follows:

$$\text{Minimize} \quad \boldsymbol{F}(\boldsymbol{X}) = \{F_1(\boldsymbol{X}), F_2(\boldsymbol{X}), \ldots, F_M(\boldsymbol{X})\} \quad (3)$$

where $\boldsymbol{X} = [X_1, X_2, \ldots, X_D] \in \Re^D$ is a point in the D-dimensional decision space (search space) and $\boldsymbol{F} = [F_1, F_2, \ldots, F_M] \in \Omega^M$ is the objective space with M minimization objectives. For minimization MOP as (3), some definitions of MOP are given as follows [6].

Definition 1: Pareto domination. Given two vectors $\boldsymbol{U} = [U_1, U_2, \ldots, U_M] \in \Omega^M$ and $\boldsymbol{W} = [W_1, W_2, \ldots, W_M] \in \Omega^M$ in the objective space. We say that $\boldsymbol{U}$ **dominates** (**is better than**) $\boldsymbol{W}$ if $U_m \leq W_m$, for all $m = 1, 2, \ldots, M$, and $\boldsymbol{U} \neq \boldsymbol{W}$.

Definition 2: $\mathbf{Pareto\ optimal}$. Given a vector $\boldsymbol{X} = [X_1, X_2, \ldots, X_D] \in \Re^D$ in the decision space. We say that $\boldsymbol{X}$ is $\mathbf{Pareto\ optimal}$ if there is no $\boldsymbol{X}^* \in \Re^D$ such that $\boldsymbol{F}(\boldsymbol{X}^*)$ dominates $\boldsymbol{F}(\boldsymbol{X})$.

Definition 3: Pareto set. The **Pareto set** PS is defined as

$$PS = \{\boldsymbol{X} \in \Re^D \text{ and } \boldsymbol{X} \text{ is } \mathbf{Pareto\ optimal}\}. \quad (4)$$

Definition 4: PF. The **Pareto front** PF is defined as

$$PF = \{\boldsymbol{F}(\boldsymbol{X}) \mid \boldsymbol{X} \in PS\}. \quad (5)$$

C. Related Work on MOP

There are many algorithms, like MOEAs and MOPSOs, for solving MOPs [5], [6].

Some researchers used the aggregation approach to solve MOPs [17]. That is, the multiple objectives are weighted and summed up to form a single objective, and the obtained single-objective problem is then optimized. However, the weights for different objectives are dependent on the problem or the decision makers. Therefore, they are difficult to be determined. Moreover, the aggregation approach obtains only one solution in a run, which is not sufficient in practical applications. To overcome these disadvantages, Zhang and Li [8] proposed to

decompose the MOP into different scalar optimization subproblems and use different weights to these subproblems to obtain a set of Pareto solutions in a single run.

Some other researchers proposed to optimize the objectives alternately when solving MOPs. This approach is also named as the lexicographic ordering approach [6] because the objectives are often ranked in the order of importance and the solutions are obtained by optimizing the objective functions alternately according to their ranks. For example, in Hu and Eberhart's MOPSO [18], the neighborhood of one particle is formed by some nearest particles according to the objective space of one objective, while the best particle in the neighborhood is determined by the fitness of the other objective. However, as the approach used in [18] involves using two objectives for determining the neighbors and the neighborhood best particle, respectively, it seems to be useful only in MOP with two objectives. Moreover, determining the importance of different objectives is problem dependent, and the ordering of the objectives may affect the performance significantly.

Other researchers applied the concept of Pareto dominance to solve MOPs. The multiobjective genetic algorithm (GA) assigns a rank to each individual according to the number of other individuals that dominate it [19]. Then, the fitness is assigned to the individual based on its rank. The niching Pareto GA compares every two individuals based on the Pareto domination tournament strategy [20]. The nondominated sorting GA (NSGA-II) sorts out all the individuals according to the Pareto domination relationship and selects individuals with better ranks to form the next-generation population [7]. Moreover, many MOPSOs adopt the Pareto domination concept when assigning the fitness value of the individuals [21], [22], e.g., when determining the personal historically best position [23] or selecting the globally best position [24].

Recently, some new work has been reported to use new techniques to help solve MOPs more efficiently. In the studies of Rachmawati and Srinivasan [25], Karahan and Koksalan [26], and Zitzler *et al.* [27], the preference information is used in MOEAs for better selecting the individuals for the next generation. In designing efficient MOEAs, Wang *et al.* [28] used a hybrid technique, Adra *et al.* [29] used a convergence acceleration operator, Avigad and Moshaiov [30] used an interactive concept, Lara *et al.* [31] used a hill climber with a sidestep local search strategy, Song and Kusiak [32] used a data mining process, and Zhang *et al.* [33] used a Gaussian process model. Moreover, some multiobjective optimization algorithms based on a memetic algorithm (MA) [34], quantum GA [35], differential evolution (DE) [36], [37], and estimation of distribution algorithm (EDA) [38] have also been proposed in recent years.

III. CMPSO FOR MOP

The CMPSO algorithm is based on the MPMO technique that uses multiple swarms to optimize different objectives. In this section, details of the evolutionary process for each swarm and the information-sharing strategy for all swarms are described. Later, the complete CMPSO process is presented, and the novelties of CMPSO are discussed.

A. CMPSO Evolutionary Process

Suppose that there are M objectives in the MOP, and therefore, there are M swarms working concurrently in CMPSO to optimize the MOP. The evolutionary process in each swarm is similar to that in a conventional PSO that is used to optimize a single-objective problem. Without loss of generality, we herein consider only one of the M swarms, indicating by index of m, to describe the evolutionary process.

In the initialization, each particle i in the mth swarm randomly initializes its velocity $\boldsymbol{V_i^m}$ and position $\boldsymbol{X_i^m}$, evaluates $\boldsymbol{X_i^m}$, and lets $\boldsymbol{pBest_i^m}$ be the same as $\boldsymbol{X_i^m}$. In every generation during the evolutionary process, for the mth swarm, each particle i updates its velocity and position by the influences of its personal historically best position $\boldsymbol{pBest_i^m}$, the swarm's globally best position $\boldsymbol{gBest^m}$, and an archive position $\boldsymbol{Archive_i^m} = [A_{i1}^m, A_{i2}^m, \ldots, A_{iD}^m]$ selected from the archive by the particle i. The velocity update is

$$V_{id}^m = \omega V_{id}^m + c_1 r_{1d}\,(P_{id}^m - X_{id}^m) + c_2 r_{2d}\,(G_d^m - X_{id}^m) + c_3 r_{3d}\,(A_{id}^m - X_{id}^m) \quad (6)$$

where d is the index of the dimension and r_{3d} is a random value in the range of [0, 1]. The position update is

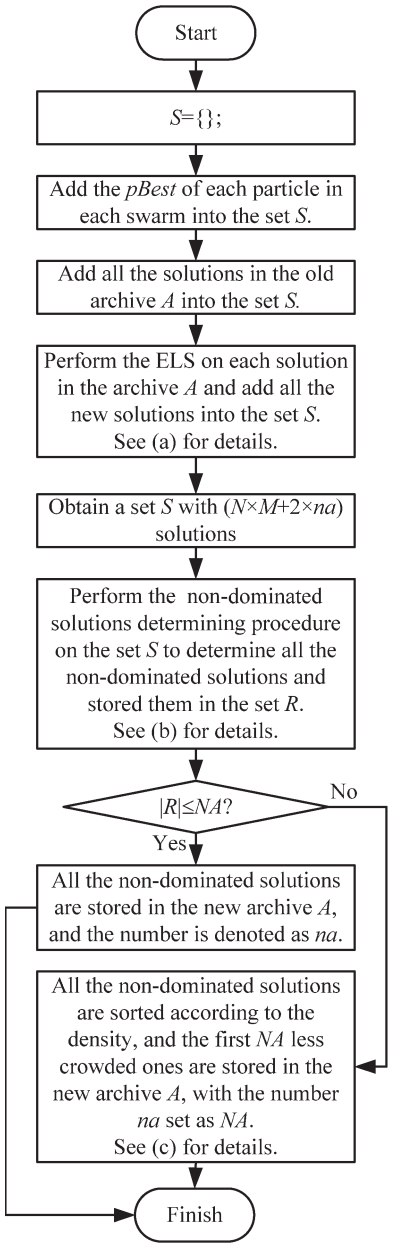
$$X_{id}^m = X_{id}^m + V_{id}^m. \quad (7)$$

In the velocity update equation, the term $c_3 r_{3d}(A_{id}^m - X_{id}^m)$ is the sharing information from the shared archive. With the help of solution information in the shared archive, the particle can use the search information not only from its own swarm but also from other swarms. The particle is expected to search along the whole PF by using the whole search information of all the swarms instead of being attracted to the margin only by the search information of its own swarm. Therefore, the algorithm can approximate the whole PF fast with the help of the archived information. $\boldsymbol{Archive_i^m}$ is chosen by randomly selecting a solution from the shared archive by the particle i. A random selection method is rapid, has advantages of high diversity, and is with low computational cost. Therefore, it is adopted in this paper. One notice is that if the archive is empty, then $\boldsymbol{Archive_i^m}$ is selected randomly from the $\boldsymbol{gBests}$ of the other $M-1$ swarms, excluding the mth swarm itself.

B. CMPSO Archive Update

CMPSO uses an external archive to store the nondominated solutions from all the swarms. This archive is used not only to store the nondominated solutions to be reported at the end of the evolution like in the traditional MOP algorithm [39] but also for information sharing among different swarms. Therefore, the particles in all the swarms can access the information in the archive and use it to guide the flying, as indicated in (6).

Denoted as A, the archive is initialized to be empty and is updated in every generation. Research shows that it is better to use an archive with a fixed maximal size because the number of nondominated solutions may increase very fast [6]. Therefore, in CMPSO, the archive is set with a maximal size of NA, and the size of the archive in the current generation is denoted as na.




```
Procedure Elitist_learning_strategy
Begin
  For each solution A_i in the archive A
    E_i = A_i;
    d = rand (1, D);
    E_id = E_id + (X_max,d − X_min,d)Gaussian(0, 1);
    Keep the E_id within the range [X_min,d, X_max,d];
    Evaluate all the objectives of the new solution E_i;
  End of For
End.
```

(a)

```
Procedure Nondominated_solutions_determining
Begin
  R = {};
  For each solution i in the set S
    flag = ture;
    For each solution j in the set S
      If j dominates i
        flag = false;
        break;
      End of If
    End of For
    If flag == ture
      Add solution i into set R;
    End of If
  End of For
End.
```

(b)

```
Procedure Density_based_selection
Begin
  L = sizeof (R);
  For each solution i in the set R
    d_i = 0;
  End of For
  For m=1 to M //Each objective
    Sort R according to the m^th objective value;
    d_1 = MAX; d_L = MAX;
    for i=2 to L − 1
      d_i = d_i + (R_i+1.F_m − R_i−1.F_m)/(F_m, max − F_m, min);
    End of For
  End of For
  Sort R according to the d_i;
  Select the first NA solutions with large d_i into the archive A;
End.
```

(c)

Fig. 2. Archive update process. (a) Pseudocode of the ELS procedure. (b) Pseudocode of the nondominated-solution-determining procedure. (c) Pseudocode of the density-based selection procedure.

At the end of every generation, the archive A is updated. The archive update process is shown in Fig. 2 and is described as follows. First, a new set S is initialized to be empty. Then, the $\boldsymbol{pBest}$ of each particle in each swarm is added into the set S. Then, all the solutions in the old archive A are added into the set S. Later, ELS [16] is performed on each solution in the archive A to form a new solution. All the new solutions are then added into the set S. After the aforesaid operations, the set S would have $(N \times M + 2 \times na)$ solutions, where N and M are the population size of a swarm and the swarm number (objectives number), respectively, and na is the number of solutions of the old archive A. Then, a nondominated-solution-determining procedure is performed on the set S to determine all the nondominated solutions and store them in a set R. If the size of R (the number of nondominated solutions) is not larger than NA, then all the nondominated solutions are stored in the new archive A, and na is set as the size of R. Otherwise, all the nondominated solutions are sorted according to the density, and the first NA less crowded ones are selected to store in the new archive A, with the number na set as NA.

The following parts give the details of the ELS procedure, the nondominated-solution-determining procedure, and the density-based selection procedure.

1) ELS: ELS was first introduced in adaptive PSO [16] for the globally best particle to jump out of possible local optima. In this paper, we perform ELS on all the solutions in the archive because they are all globally best solutions of CMPSO. The pseudocode of the procedure is presented in Fig. 2(a) and is described as follows.

For each solution A_i in the archive A, first, let the new solution E_i equal to A_i, and then, a random dimension d is selected to perform Gaussian perturbation as

$$E_{id} = E_{id} + (X_{\max,d} - X_{\min,d})Gaussian(0,1) \quad (8)$$

where $X_{\max,d}$ and $X_{\min,d}$ are the upper and lower bounds of the dth dimension, respectively. $Gaussian(0,1)$ is a random value generated by a Gaussian distribution with a mean value of 0 and a standard deviation of 1.

After the perturbation, E_{id} is checked and is guaranteed to be in the search range of $[X_{\min,d}, X_{\max,d}]$; otherwise, E_{id} is set to the corresponding bound. Later, all the objectives of the new solution E_i are calculated, and the solution E_i is added into the set S.

2) Nondominated-Solution Determination: This procedure is to determine the nondominated solutions in a given solution set S. The procedure is described as follows, and its pseudocode is shown in Fig. 2(b). A set R is used to store the nondominated solutions, and it is initialized to be empty. Then, for each solution i in the set S, the procedure checks whether the solution i is dominated by any other solution j. If the solution i is not dominated by any other solution, then the solution i is added into the set R. This checking process is performed on all the solutions in the set S, and all the nondominated solutions can be determined and stored in the set R.

3) Density-Based Selection: This procedure is performed if the size of R is larger than the maximal size of the archive (NA). The function is to select less crowded solutions in the first NA into the new archive. The density estimation metric was introduced in [7] and is adopted in this paper. For the detailed calculation, the procedure is shown in Fig. 2(c) and is described as follows. Given a set of solutions R, the distance of each solution is initialized to be zero. Then, all the solutions are sorted according to each objective value, from the smallest to the largest. For each objective, the boundary solutions, i.e., the solutions with the smallest and the largest value on this objective, are assigned an infinite distance value. The distance of the other solutions is increased by the absolute normalized difference of the objective values between the two adjacent solutions.

After the density estimation, all the solutions in the set R are assigned with a distance. Then, we can select the first NA solutions with large distances to the new archive A.

C. Complete Algorithm

The previous subsections describe the evolutionary process and the archive update process. The complete CMPSO algorithm is shown in Fig. 3 and is described as follows.

In the initialization, M swarms with N particles in each swarm are initialized, and the archive A is set to be empty. For each particle i in the mth swarm, initialize its velocity $\boldsymbol{V_i^m}$ and position $\boldsymbol{X_i^m}$, calculate all the objectives of $\boldsymbol{X_i^m}$, and let $\boldsymbol{pBest_i^m}$ be the same as $\boldsymbol{X_i^m}$. All the $\boldsymbol{pBest_i^m}$ are compared according to the fitness of the mth objective to determine the globally best position $\boldsymbol{gBest^m}$ for the mth swarm. After the initialization, the archive A is updated, and the algorithm starts its evolutionary process.

```
Algorithm CMPSO
Begin
  //Initialization
  A={}; gen=0;
  For each swarm m=1 to M
    For each particle i=1 to N in the m^th swarm
      Random initialize its velocity V_i^m, position X_i^m;
      Evaluate X_i^m;
      pBest_i^m = X_i^m;
    End of For
    gBest^m=arg pBest_i^m that min{pBest_i^m.F_m};
  End of For
  Update the archive A;
  //Evolutionary process
  While not stop
    For each swarm m=1 to M
      For each particle i=1 to N in the m^th swarm
        Select an archive solution randomly from A for the particle i;
        Update the velocity and position of the particle i as Eqs. (6) and (7);
        Evaluate X_i^m;
        If X_i^m.F_m<pBest_i^m.F_m
          pBest_i^m=X_i^m;
        If pBest_i^m.F_m<gBest^m.F_m
          gBest^m=pBest_i^m;
      End of For
    End of For
    Update the archive A;
    gen=gen+1;
  End of While
  //Finish
  Report the solutions in A;
End.
```

Fig. 3. Complete CMPSO algorithm.

In every generation during the evolutionary process, each particle i in the mth swarm randomly selects first an archive solution from A and then updates its velocity and position according to (6) and (7). All the objectives of the new position $\boldsymbol{X_i^m}$ are calculated, and $\boldsymbol{X_i^m}$ will replace the historically best position $\boldsymbol{pBest_i^m}$ if the mth objective of $\boldsymbol{X_i^m}$ is smaller than the mth objective of $\boldsymbol{pBest_i^m}$. Moreover, $\boldsymbol{gBest^m}$ will be replaced by $\boldsymbol{pBest_i^m}$ if $\boldsymbol{pBest_i^m}$ has a smaller fitness value on the mth objective than $\boldsymbol{gBest^m}$ does. At the end of the generation, the archive A is updated, and the process repeats until the termination criterion is met. When the algorithm finishes, the solutions in the archive A are reported.

D. Novelties of CMPSO

The novelties of the CMPSO algorithm lie not only in that it is based on the MPMO technique that addresses the fitness assignment problem naturally but also in that it uses an external shared archive and two novel designs to implement the information-sharing strategy efficiently. One design is to modify the velocity update equation to use the shared-archive information to accelerate the speed in order to approximate the PF, and the other is to use ELS on the archive solution update to avoid local optimal PFs. Therefore, CMPSO is different from many existing multiple-population algorithms.

First, CMPSO is novel because the MPMO technique is different from existing multiple-population techniques that try to design a parallel [40] or distributed [41] algorithm for MOP. For example, the approach designed by Ewald *et al.* [42] is to use multiple populations, with each population adopting the same

TABLE I
CHARACTERISTICS OF THE TEST PROBLEMS

Name	D	Range	Optimum	Comments
ZDT1	30	$x_i\in[0, 1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=0$, $2\le i\le D$	Convex, F_1 U†, F_2 U
ZDT2	30	$x_i\in[0, 1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=0$, $2\le i\le D$	Concave, F_1 U, F_2 U
ZDT3	30	$x_i\in[0, 1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=0$, $2\le i\le D$	Convex, Disconnected, F_1 U, F_2 M‡
ZDT4	10	$x_1\in[0, 1]$, $x_i\in[-5,5]$, $2\le i\le D$	$x_1\in[0, 1]$, $x_i=0$, $2\le i\le D$	Concave, F_1 U, F_2 M
ZDT6	10	$x_i\in[0, 1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=0$, $2\le i\le D$	Concave, Non-uniformly, F_1 M, F_2 M
DTLZ1	10	$x_i\in[0, 1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=0.5$, $2\le i\le D$	Linear, F_1 M, F_2 M
DTLZ2	10	$x_i\in[0, 1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=0.5$, $2\le i\le D$	Concave, F_1 U, F_2 U
WFG1	10	$z_i\in[0, 2i]$, $1\le i\le D$	$z_i\in[0, 2i]$, $1\le i\le k$, $z_i=0.35$, $k+1\le i\le D$	Convex, Mixed, F_1 U, F_2 U
WFG2	10	$z_i\in[0, 2i]$, $1\le i\le D$	$z_i\in[0, 2i]$, $1\le i\le k$, $z_i=0.35$, $k+1\le i\le D$	Convex, Disconnected, F_1 U, F_2 M
WFG3	10	$z_i\in[0, 2i]$, $1\le i\le D$	$z_i\in[0, 2i]$, $1\le i\le k$, $z_i=0.35$, $k+1\le i\le D$	Linear, Degenerate, F_1 U, F_2 U
WFG4	10	$z_i\in[0, 2i]$, $1\le i\le D$	$z_i\in[0, 2i]$, $1\le i\le k$, $z_i=0.35$, $k+1\le i\le D$	Concave, F_1 M, F_2 M
UF1	30	$x_1\in[0, 1]$, $x_i\in[-1,1]$, $2\le i\le D$	$x_1\in[0, 1]$, $x_i=\sin(6\pi x_1+\frac{j\pi}{D})$, $2\le i\le D$	Convex, F_1 M, F_2 M
UF2	30	$x_1\in[0, 1]$, $x_i\in[-1,1]$, $2\le i\le D$	$x_1\in[0, 1]$, $x_i=\begin{cases}(0.3x_1^2\cos(24\pi x_1+\frac{4i\pi}{D})+0.6x_1)\cos(6\pi x_1+\frac{j\pi}{D}), & j\in J_1\\ (0.3x_1^2\cos(24\pi x_1+\frac{4i\pi}{D})+0.6x_1)\sin(6\pi x_1+\frac{j\pi}{D}), & j\in J_2\end{cases}$, $2\le i\le D$	Convex, F_1 M, F_2 M
UF3	30	$x_i\in[0,1]$, $1\le i\le D$	$x_1\in[0, 1]$, $x_i=x_1^{0.5(1.0+\frac{3(i-2)}{D-2})}$, $2\le i\le D$	Convex, F_1 M, F_2 M
UF4	30	$x_1\in[0, 1]$, $x_i\in[-2,2]$, $2\le i\le D$	$x_1\in[0, 1]$, $x_i=\sin(6\pi x_1+\frac{j\pi}{D})$, $2\le i\le D$	Concave, F_1 M, F_2 M
UF5	30	$x_1\in[0, 1]$, $x_i\in[-1,1]$, $2\le i\le D$	$(F_1, F_2)=(\frac{i}{2N}, 1-\frac{i}{2N})$, $0\le i\le 2N$, $N=10$	Scatter, F_1 M, F_2 M
UF6	30	$x_1\in[0, 1]$, $x_i\in[-1,1]$, $2\le i\le D$	$F_1=\bigcup_{i=1}^{N}[\frac{2i-1}{2N}, \frac{2i}{2N}]$, $F_2=1-F_1$, $N=2$	Linear, Disconnected, F_1 M, F_2 M
UF7	30	$x_1\in[0, 1]$, $x_i\in[-1,1]$, $2\le i\le D$	$x_1\in[0, 1]$, $x_i=\sin(6\pi x_1+\frac{j\pi}{D})$, $2\le i\le D$	Linear, F_1 M, F_2 M

†U: Unimodal, ‡M: Multimodal

NSGA-II [7], to optimize a whole MOP with all the objectives. In designing efficient multiple populations, Leong and Yen proposed to use dynamic population size in [43], while they proposed to use dynamic number of populations in [44]. However, these approaches do not treat the objectives separately but consider all the objectives as a whole in each population.

Second, CMPSO is novel because it has an efficient information-sharing strategy. Even though some similar ideas that use multiple populations to deal with different objectives can be also found in the literature, they are not designed fully to use the information of different populations efficiently. In the vector-evaluated GA (VEGA) developed by Schaffer [45], different subpopulations are selected from the current generation according to different objectives. Later, Parsopoulos *et al.* [46] modified the VEGA idea to fit the PSO framework and proposed the vector-evaluated PSO (VEPSO). In VEPSO, two or more swarms are used to search the space. When updating the velocity and position, a particle learns from its personal best experience and the best experience of its neighbor swarm. Also, in [47], Chow and Tsui decomposed the function into several subfunctions and used multiple swarms to optimize each subfunction separately. They also defined a static ring topology for these populations to exchange search information among the neighbors. However, the information-sharing strategy is not sufficiently studied in these approaches, and how to sufficiently share the search information of different populations so as to enhance the algorithm performance is still a crucial problem [48]. Therefore, in this paper, we develop the algorithm based on the MPMO technique but also the efficient information-sharing strategy in the algorithm for different populations to exchange their research information during the evolutionary process. Moreover, the information-sharing strategy is efficient for that it uses two novel designs. The first novel design is to modify the velocity update equation to use the shared-archive information to accelerate the convergence speed to approximate the PF, and the other design is to use ELS to avoid local optimal PFs.

IV. EXPERIMENTAL STUDIES

A. Test Problems

Various test problems have been proposed to evaluate the multiobjective optimization algorithms in the literature [49]. First, we adopt the most frequently used problems ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 from the ZDT test set [50]. However, some researchers argue that ZDT problems lack characteristics such as variable linkage, objective function modality, and PF shape. Therefore, we also adopt DTLZ1 and DTLZ2 from the DTLZ test set [51], and WFG1, WFG2, WFG3, and WFG4 from the WFG test set [52], which are with multimodal objective functions and nonseparable variables. Most recently, Zhang *et al.* [53] proposed a new UF test set where problems are with complicated Pareto sets. In this paper, we further choose all the two-objective unconstrained problems UF1–UF7 to evaluate the algorithm performance.

Totally, 18 test problems (5 ZDT, 2 DTLZ, 4 WFG, and 7 UF) are used, and their characteristics are described in Table I. The problems are with characteristics of different

TABLE II
PARAMETER SETTINGS OF THE ALGORITHMS

Algorithms	Parameters Settings
NSGA-II	N=100, p_x=0.9, p_m=1/D, η_c=20, and η_m=20
GDE3	N=100, CR=0.0, and F=0.5
MOEA/D-DE	N=100, CR=1.0, F=0.5, η=20, p_m=1/D, T=20, δ=0.9, and n_r=2
MOCLPSO	N=50, p_c=0.1, p_m=0.4, ω=0.9→0.2, and c=2.0
OMOPSO	N=100, ω=*rand*(0.1, 0.5), c_1=*rand*(1.5, 2.0), c_2=*rand*(1.5, 2.0)
VEPSO	N=100, χ=0.729, c_1=c_2=2.05, and M=6
CMPSO	N=20, ω=0.9→0.4, and c_1=c_2=c_3=4.0/3

dimensionalities like 10 and 30, of different objective functions like unimodal and multimodal, of different PFs like scatter, linear, convex, concave, disconnected, nonuniformly, and mixed, and of different Pareto set shapes like simple and complicated. Therefore, they are comprehensive and useful for testing the algorithm's performance from different aspects. For more details of the problems, please refer to [50]–[53] for ZDT, DTLZ, WFG, and UF, respectively.

B. Performance Metric

The inverted generational distance (IGD) indicator is adopted in this paper as the performance metric because it can reflect both the convergence and diversity of the obtained solutions to the true PF. The indicator has been widely adopted and strongly recommended in the MOP community in recent years [53]. Assuming that the set of nondominated solutions obtained by an algorithm is A and a set of solutions uniformly sampled along the true PF is P, the calculation of $IGD(A, P)$ is

$$IGD(A,P) = \frac{\sum_{i=1}^{|P|} d(P_i, A)}{|P|} \tag{9}$$

where $|P|$ is the size of the set P and $d(P_i, A)$ denotes the distance between the solution P_i and the solution in the set A that is nearest to P_i, measured by the Euclidean distance in the objective space. This IGD indicator has an assumption that the true PF is known. In this paper, we sample 500 uniformly distributed points along the PF to form the set P for each problem [7]. Intuitively, if the nondominated solutions in the set A have a good spread along the true PF, then the indicator IGD will have a small value. Some more performance metrics are also adopted to evaluate the algorithm performance, and the results are given in Section IV-I.

C. Experimental Settings

In this paper, we compare the results obtained by CMPSO with not only MOEAs but also MOPSOs. The MOEAs include NSGA-II [7], generalized DE 3 (GDE3) [36], and MOEA with decomposition and DE operators (MOEA/D-DE) [37], while the MOPSOs include multiobjective comprehensive learning PSO (MOCLPSO) [54], optimized MOPSO (OMOPSO) [55], and VEPSO [46]. These algorithms are chosen because NSGA-II and MOCLPSO are two state-of-the-art algorithms, GDE3 and MOEA/D-DE are two most recently well-performing MOEAs, OMOPSO is a very salient MOPSO according to a very recent comparative study [56], and VEPSO is a MOPSO that also uses multiple populations. Therefore, these algorithms are representative and helpful to make the comparisons more comprehensive and convincing.

The parameters of the aforementioned algorithms are set according to the proposals in their corresponding references, as summarized in Table II. For CMPSO, we adopt the common configurations that the inertia weight is linearly decreasing from 0.9 to 0.4 [15], and the acceleration coefficients c_1, c_2, and c_3 are set to be 4.0/3. We set such a value for c_i because the sum of c_i is usually 4.0 in PSO. As CMPSO uses different swarms to optimize different objectives, we set a relative small population size of 20 for each swarm. In order to make the comparisons fair, all the seven algorithms have the same archive size of 100 [7], [56]. The nondominated solutions in the archive are updated and used to calculate the IGD value in every generation and are reported at the end of the algorithm running.

It should be noticed that when solving different kinds of MOP, different population size and different maximal number of function evaluations (FEs) are used. The population sizes in Table II are used when solving the ZDT problems, and the maximal number of FEs is set to be 25 000 [7]. However, when solving the more difficult DTLZ and WFG problems, the population size is set to be 200 for all the algorithms (except CMPSO that still uses a population size of 20 for each swarm), and the maximal number of FEs is 1×10^5. When solving the complicated UF problems, all the algorithms (CMPSO still uses a population size of 20 for each swarm) are with a population size of 300 and a maximal number of FEs of 3×10^5 [53]. The impacts of population size on CMPSO performance will be investigated in Section IV-H. Moreover, the experimental results are the average values of 30 independent runs. The best results are denoted by the **bold** font. The results obtained by different algorithms are compared with that by CMPSO by the Wilcoxon rank sum test with significant level $\alpha = 0.05$.

D. Experimental Results on ZDT Problems

Table III compares the results on the ZDT problems. The results show that CMPSO is promising in dealing with ZDT problems with both convex and concave PFs. It performs best on ZDT1 that is with convex PF and on ZDT2 that is with concave PF. It also performs the second best on ZDT6 (only slightly worse than MOCLPSO) that is with nonuniform concave PF. Moreover, CMPSO is very promising (the third best) on ZDT3 whose PF is disconnected convex.

As CMPSO has the best performance on ZDT1 and ZDT2 whose objectives are all unimodal, it indicates that CMPSO has the strong convergence ability to approximate the PF of MOP with simple objectives. Table III also shows that all the MOPSOs are beaten by MOEAs on ZDT4. This may be caused

TABLE III
RESULT COMPARISONS ON THE ZDT PROBLEMS

Problems		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO
ZDT1	Mean	5.00×10^{-3}	1.27×10^{-2}	0.16	4.80×10^{-3}	7.02×10^{-3}	0.31	**4.13×10^{-3}**
	Std	2.33×10^{-4}	1.56×10^{-3}	1.93×10^{-2}	1.76×10^{-4}	7.83×10^{-4}	3.39×10^{-2}	**8.30×10^{-5}**
	Rank	3 –	5 –	6 –	2 –	4 –	7 –	**1**
ZDT2	Mean	0.19	2.97×10^{-2}	0.23	0.38	6.06×10^{-3}	0.33	**4.32×10^{-3}**
	Std	0.28	1.82×10^{-2}	3.07×10^{-2}	0.30	3.81×10^{-4}	0.11	**1.03×10^{-4}**
	Rank	4 –	3 –	5 –	7 –	2 –	6 –	**1**
ZDT3	Mean	1.54×10^{-2}	1.16×10^{-2}	0.23	**5.49×10^{-3}**	2.30×10^{-2}	0.60	1.39×10^{-2}
	Std	2.71×10^{-2}	2.24×10^{-3}	2.17×10^{-2}	**2.49×10^{-4}**	5.99×10^{-3}	9.40×10^{-2}	3.49×10^{-3}
	Rank	4 –	2 +	6 –	**1** +	5 –	7 –	3
ZDT4	Mean	**0.29**	0.34	0.31	3.26	16.47	26.75	0.79
	Std	0.40	0.37	**0.23**	1.35	4.12	6.06	0.26
	Rank	**1** +	3 +	2 +	5 –	6 –	7 –	4
ZDT6	Mean	6.22×10^{-3}	7.36×10^{-2}	1.54	**3.69×10^{-3}**	4.61×10^{-3}	0.41	3.72×10^{-3}
	Std	7.02×10^{-4}	9.16×10^{-2}	0.13	**1.31×10^{-4}**	3.36×10^{-4}	0.19	1.47×10^{-4}
	Rank	4 –	5 –	7 –	**1** ≈	3 –	6 –	2
Final	Total	16	18	26	16	20	33	**11**
Rank	Final	2	4	6	2	5	7	**1**
Better – Worse		-3	-1	-3	-2	-5	-5	
Algorithms		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO

'+', '–', and '≈' indicate that the results of the algorithm are significantly better than, worse than, and similar to the ones of CMPSO by Wilcoxon's rank sum test with α=0.05.
The row 'Better – Worse' shows the number of '+' minus '–'. Value smaller than 0 means generally beaten by CMPSO.

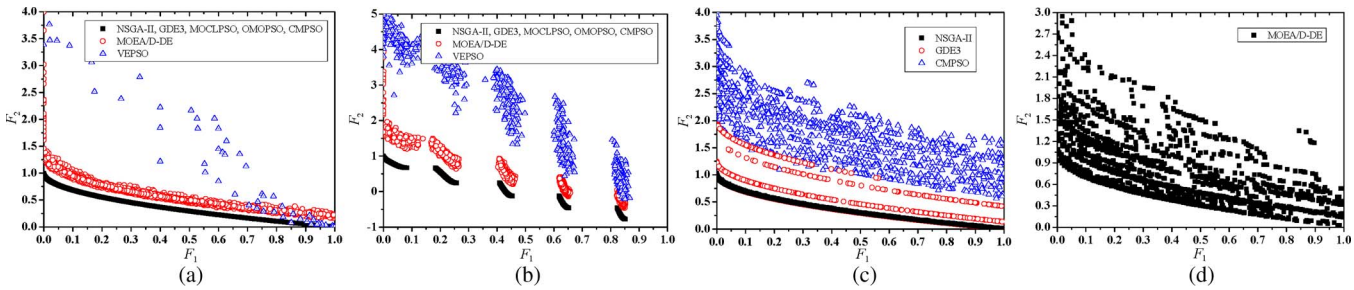


Fig. 4. Final nondominated solutions of the ZDT problems in all the 30 runs. (a) ZDT1. (b) ZDT3. (c) ZDT4. (d) ZDT4 (MOEA/D-DE).

by the local PFs of ZDT4 for that it is with the multimodal Rastrigin function. However, CMPSO is still the best algorithm among all the four MOPSOs, and only CMPSO can obtain comparable results with MOEAs. This indicates that CMPSO has the ability to avoid local PFs caused by complex objectives. Moreover, when the general performance is considered over all the problems, CMPSO is the winner for that it has the first average rank among the seven contenders over all the five problems. The Wilcoxon rank sum tests also indicate that CMPSO significantly outperforms all the six competitors on the ZDT set problems.

Fig. 4 shows the final nondominated solutions obtained by different algorithms in all the 30 runs when solving some of the ZDT problems. Notice that some algorithms have similar performance on the same problems, and therefore, we only use one algorithm as the representative. For example, the figures of the solutions obtained by NSGA-II, GDE3, MOCLPSO, OMOPSO, and CMPSO on ZDT1 are not evidently different, and therefore, we only use the solutions obtained by CMPSO to plot in Fig. 4(a) for clarity. The figures in Fig. 4 show that the solutions obtained by CMPSO not only approximate the whole PF well but also form a good spread along the whole PF. Fig. 4(c) and (d) shows the obtained solutions to ZDT4. As MOCLPSO, OMOPSO, and VEPSO perform very poorly on this problem, the solutions are not plotted in the figures.

E. Experimental Results on DTLZ and WFG Problems

The results on the DTLZ and WFG problems are presented and compared in Table IV. The results show that CMPSO performs competitively with GDE3, NSGA-II, and MOEA/D-DE on DTLZ1 and performs the best on DTLZ2. Moreover, CMPSO yields the best IGD values on all the four WFG problems. In general, CMPSO has the first average rank over all the six DTLZ and WFG problems, followed by other MOPSOs and then MOEAs. The statistics by the Wilcoxon rank sum tests also confirm that CMPSO has significantly better performance than all the six contenders. As these problems are with mixed, disconnected, or degenerated PFs, the good performance of CMPSO indicates that it is promising not only in MOPs with simple PFs like the ZDT problems but also in MOPs with complex PFs like the DTLZ and WFG problems.

Fig. 5 further confirms the advantages of CMPSO by plotting the obtained nondominated solutions in 30 runs. Although all the algorithms fail to obtain the true PF of WFG1, CMPSO gives the best PF approximation and yields the best diversity to spread along the PF. For WFG2, WFG3, and WFG4, only CMPSO obtain solutions approximating to the PFs. CMPSO is observed to perform remarkably better than not only the other MOPSOs but also all the compared MOEAs.

TABLE IV
RESULT COMPARISONS ON THE DTLZ AND WFG PROBLEMS

Problems		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO
DTLZ1	Mean	2.75×10^{-3}	**2.42×10^{-3}**	2.75×10^{-3}	38.75	43.04	91.56	5.67×10^{-2}
	Std	2.86×10^{-4}	**1.34×10^{-4}**	4.36×10^{-4}	7.36	8.06	23.42	2.21×10^{-2}
	Rank	2 +	**1** +	3 +	5 –	6 –	7 –	4
DTLZ2	Mean	5.81×10^{-3}	5.49×10^{-3}	3.33×10^{-2}	8.79×10^{-3}	6.65×10^{-3}	6.07×10^{-2}	**4.62×10^{-3}**
	Std	4.70×10^{-4}	6.85×10^{-4}	3.02×10^{-3}	8.06×10^{-4}	6.08×10^{-4}	8.69×10^{-3}	**1.50×10^{-4}**
	Rank	3 –	2 –	6 –	5 –	4 –	7 –	**1**
WFG1	Mean	1.80	2.58	2.57	1.37	1.38	1.64	**1.23**
	Std	0.31	1.03×10^{-2}	**1.62×10^{-3}**	0.13	3.71×10^{-3}	0.30	6.69×10^{-2}
	Rank	5 –	7 –	6 –	2 –	3 –	4 –	**1**
WFG2	Mean	0.46	1.14	0.96	0.38	0.40	0.41	**0.11**
	Std	4.03×10^{-2}	4.66×10^{-2}	7.94×10^{-2}	**$3.45\times\times10^{-2}$**	4.65×10^{-2}	4.34×10^{-2}	6.19×10^{-2}
	Rank	5 –	7 –	6 –	2 –	3 –	4 –	**1**
WFG3	Mean	0.36	1.66	0.78	0.30	0.30	0.33	**1.47×10^{-2}**
	Std	3.07×10^{-2}	0.28	4.83×10^{-2}	2.36×10^{-2}	2.61×10^{-2}	2.25×10^{-2}	**5.80×10^{-4}**
	Rank	5 –	7 –	6 –	2 –	3 –	4 –	**1**
WFG4	Mean	0.35	1.09	0.65	0.22	0.22	0.27	**1.37×10^{-2}**
	Std	4.17×10^{-2}	0.12	3.34×10^{-2}	1.00×10^{-2}	1.23×10^{-2}	1.68×10^{-2}	**4.99×10^{-4}**
	Rank	5 –	7 –	6 –	2 –	3 –	4 –	**1**
Final	Total	25	31	33	18	22	30	**9**
Rank	Final	4	6	7	2	3	5	**1**
Better – Worse		-4	-4	-4	-6	-6	-6	
Algorithms		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO

'+', '–', and '≈' indicate that the results of the algorithm are significantly better than, worse than, and similar to the ones of CMPSO by Wilcoxon's rank sum test with α=0.05.
The row 'Better – Worse' shows the number of '+' minus '–'. Value smaller than 0 means generally beaten by CMPSO.

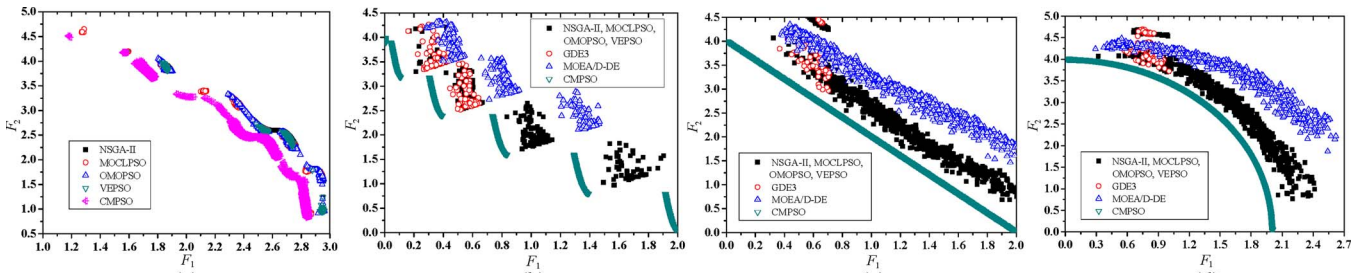


Fig. 5. Final nondominated solutions of the WFG problems in all the 30 runs. (a) WFG1. (b) WFG2. (c) WFG3. (d) WFG4.

F. Experimental Results on UF Problems

The previous two subsections have demonstrated that CMPSO shows a very good performance on the ZDT problems, and the advantages of CMPSO become more evident on the DTLZ and WFG problems as the objective functions and the PFs become more complex. In this subsection, the algorithm performance is further compared on the UF problems, which are the recently proposed problems with complicated Pareto set [53].

The results compared in Table V show that CMPSO performs best on UF2, UF4, and UF5 while MOEA/D-DE performs best on UF3 and UF7, NSGA-II on UF6, and GDE3 on UF1. Even though GDE3 and MOEA/D-DE outperform CMPSO on UF1, the results are not significantly different according to the Wilcoxon test. In general, CMPSO has the first average rank, followed by GDE3 and MOEA/D-DE over all the seven UF problems. By the Wilcoxon rank sum tests, CMPSO is also the most promising algorithm in solving the UF set problems.

Fig. 6 further compares MOEA/D-DE with CMPSO by plotting the 30 nondominated fronts obtained by the two algorithms. MOEA/D-DE is observed to be weaker than our CMPSO on UF1 and UF2 for that MOEA/D-DE misses the solutions located in the middle of the PFs. For UF3, CMPSO has difficulty in obtaining enough good solutions along the PF as that by MOEA/D-DE. However, CMPSO seems to be more stable than MOEA/D-DE on UF4 because the solutions obtained by CMPSO locate closer to the true PF of UF4, as shown in Fig. 6(g) and (h).

G. Benefit of Shared Archive

The CMPSO algorithm uses an external shared archive to let different swarms share their search information and communicate with each other efficiently. In this subsection, the benefit of the shared archive is investigated, including the benefit of ELS used in the archive update process, and the benefit of using the shared-archive solution information in the particle update equation. The experimental results are given in Table VI.

As ELS is demonstrated to be helpful for bringing in diversity to avoid being trapped into local optima when solving single-objective optimization problems [16], it is also expected that

TABLE V
RESULT COMPARISONS ON THE UF PROBLEMS

Problems		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO
UF1	Mean	7.30×10^{-2}	**5.75×10^{-2}**	5.96×10^{-2}	0.10	9.81×10^{-2}	0.71	6.64×10^{-2}
	Std	2.46×10^{-2}	2.48×10^{-2}	2.15×10^{-2}	**7.17×10^{-3}**	7.91×10^{-3}	0.15	1.99×10^{-2}
	Rank	4 ≈	**1** ≈	2 ≈	6 –	5 –	7 –	3
UF2	Mean	2.06×10^{-2}	2.02×10^{-2}	6.63×10^{-2}	0.11	7.24×10^{-2}	0.15	**1.69×10^{-2}**
	Std	3.67×10^{-3}	3.81×10^{-3}	1.32×10^{-2}	3.39×10^{-3}	3.54×10^{-3}	1.29×10^{-2}	**3.37×10^{-3}**
	Rank	3 –	2 –	4 –	6 –	5 –	7 –	**1**
UF3	Mean	6.95×10^{-2}	0.16	**3.89×10^{-2}**	0.48	0.37	0.58	9.80×10^{-2}
	Std	**1.14×10^{-2}**	6.66×10^{-2}	1.57×10^{-2}	1.55×10^{-2}	9.71×10^{-3}	4.83×10^{-2}	1.39×10^{-2}
	Rank	2 +	4 –	**1** +	6 –	5 –	7 –	3
UF4	Mean	4.26×10^{-2}	2.95×10^{-2}	4.72×10^{-2}	0.12	0.16	0.17	**2.38×10^{-2}**
	Std	**4.46×10^{-4}**	1.03×10^{-3}	1.59×10^{-3}	1.10×10^{-2}	1.39×10^{-2}	6.22×10^{-3}	1.90×10^{-3}
	Rank	3 –	2 –	4 –	5 –	6 –	7 –	**1**
UF5	Mean	0.32	0.21	0.33	0.51	0.74	3.25	**0.20**
	Std	**8.41×10^{-2}**	1.61×10^{-2}	5.41×10^{-2}	0.18	0.12	0.53	2.01×10^{-2}
	Rank	3 –	2 –	4 –	5 –	6 –	7 –	**1**
UF6	Mean	**0.12**	0.30	0.14	0.40	0.40	2.83	0.14
	Std	**1.93×10^{-2}**	1.72×10^{-2}	9.05×10^{-2}	4.30×10^{-2}	3.40×10^{-2}	0.78	2.04×10^{-2}
	Rank	**1** +	4 –	3 –	6 –	5 –	7 –	2
UF7	Mean	0.16	2.97×10^{-2}	**8.34×10^{-3}**	0.19	0.22	0.69	0.12
	Std	0.16	1.02×10^{-3}	**9.40×10^{-4}**	0.15	0.15	0.16	0.13
	Rank	4 ≈	2 +	**1** +	5 –	6 –	7 –	3
Final Rank	Total	20	17	19	39	38	49	**14**
	Final	4	2	3	6	5	7	**1**
Better – Worse		-1	-4	-2	-7	-7	-7	
Algorithms		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO

'+', '–', and '≈' indicate that the results of the algorithm are significantly better than, worse than, and similar to the ones of CMPSO by Wilcoxon's rank sum test with α=0.05.
The row 'Better – Worse' shows the number of '+' minus '–'.

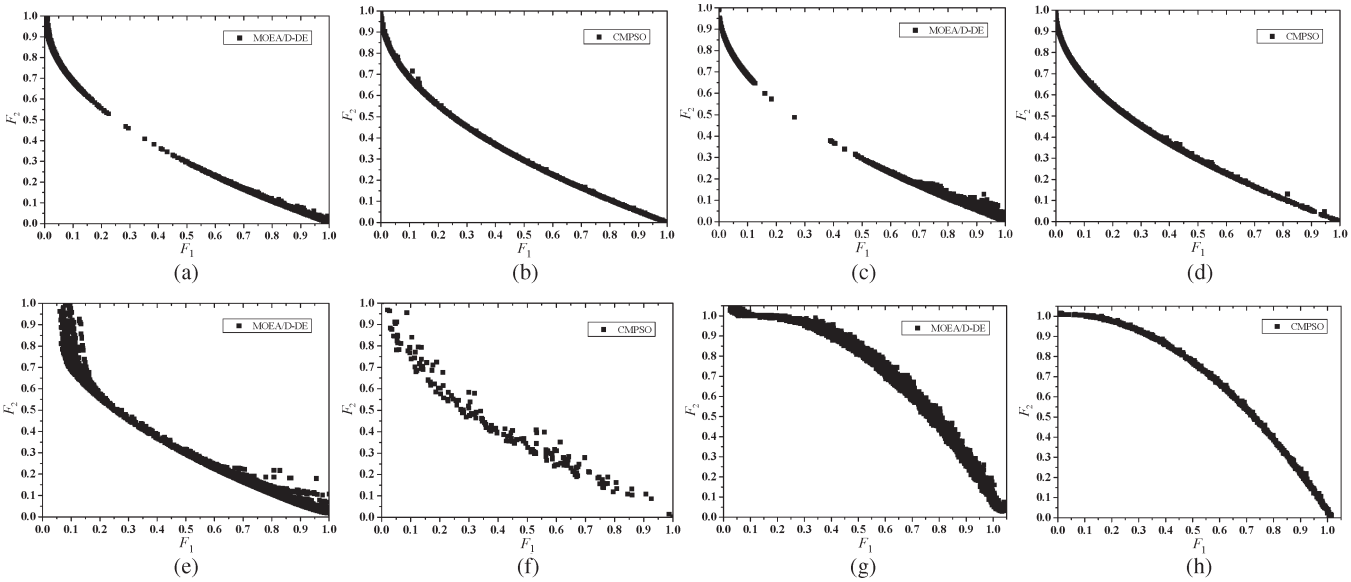


Fig. 6. Final nondominated solutions of the UF problems in all the 30 runs. (a) UF1 (MOEA/D-DE). (b) UF1 (CMPSO). (c) UF2 (MOEA/D-DE). (d) UF2 (CMPSO). (e) UF3 (MOEA/D-DE). (f) UF3 (CMPSO). (g) UF4 (MOEA/D-DE). (h) UF4 (CMPSO).

ELS can help CMPSO to avoid local PFs in MOPs. The comparisons in Table VI show that CMPSO significantly outperforms its variant without using ELS in the archive update (denoted as CMPSO-non-ELS) on all the 18 test problems. The advantages of CMPSO are more evident when solving MOPs with multimodal objective functions for that CMPSO-non-ELS is easy to be trapped into local PFs. We further compare the results obtained by CMPSO and CMPSO-non-ELS on WFG2 (with disconnected PF and one multimodal objective function), WFG4 (with concave PF and two multimodal objective functions), and UF2 (with convex PF and two multimodal objective functions) in Fig. 7. These figures clearly show that CMPSO can approximate the true PFs of these problems, while CMPSO-non-ELS is totally trapped into local PFs.

The benefit of using the shared-archive information to guide the particle update is also summarized in Table VI. The shared-archive information is expected to be beneficial for accelerating the convergence speed to approximate the PF. For the

TABLE VI
COMPARISONS BETWEEN CMPSO AND ITS VARIANTS CMPSO-NON-ELS (CMPSO WITHOUT ELS IN THE ARCHIVE UPDATE) AND CMPSO-NON-ABEST (CMPSO WITHOUT USING ARCHIVE INFORMATION FOR PARTICLE UPDATE)

Problem	CMPSO	CMPSO-non-ELS	CMPSO-non-aBest
ZDT1	**4.13×10^{-3}**	1.09×10^{-2}	
ZDT2	**4.32×10^{-3}**	1.81×10^{-2}	
ZDT3	**1.39×10^{-2}**	2.42×10^{-2}	
ZDT4	0.79	26.09	**0.78**
ZDT6	**3.72×10^{-3}**	0.18	6.63×10^{-2}
DTLZ1	**5.67×10^{-2}**	1.43×10^{2}	6.36×10^{-2}
DTLZ2	**4.62×10^{-3}**	0.12	4.64×10^{-3}
WFG1	**1.23**	2.19	1.66
WFG2	0.11	0.57	**0.10**
WFG3	**1.47×10^{-2}**	0.42	1.48×10^{-2}
WFG4	**1.37×10^{-2}**	0.31	**1.37×10^{-2}**
UF1	6.64×10^{-2}	0.38	**5.36×10^{-2}**
UF2	1.69×10^{-2}	0.16	**1.61×10^{-2}**
UF3	8.90×10^{-2}	0.57	**7.76×10^{-2}**
UF4	2.38×10^{-2}	0.19	**2.35×10^{-2}**
UF5	0.20	2.53	**0.19**
UF6	**0.14**	1.20	0.16
UF7	**0.12**	0.46	**0.12**

MOP with unimodal PF, CMPSO is observed to remarkably outperform its variant without using archived information in the particle update (denoted as CMPSO-non-aBest), e.g., on most of the ZDT, DTLZ, and WFG problems. However, CMPSO-non-aBest seems to be better than CMPSO on some of the UF problems. The reason may be that these UF problems have complicated Pareto sets, and therefore, algorithms with too fast convergence speed will cause premature convergence and cannot search the whole space more efficiently.

Fig. 8 compares the convergence characteristics of the IGD indicator on ZDT1 (with uniform convex PF and unimodal objective functions), ZDT6 (with nonuniform concave PF and multimodal objective functions), WFG1 (with mixed convex PF and unimodal objective functions), and UF2 (with convex PF and multimodal objective functions) during the CMPSO and CMPSO-non-aBest search processes. The figures further show that the utilization of the archived information to guide the particle update remarkably accelerates the convergence speed for the algorithm to approximate the PF, particularly on problems with unimodal objective functions. However, when the objective functions are multimodal, such as Fig. 8(d) for UF2, CMPSO is faster in the early evolutionary phase but is taken over by CMPSO-non-aBest in the late evolutionary phase. This is because too fast convergence speed may cause an adverse effect on prematurity and make the algorithm not search the whole space sufficiently when the Pareto sets are complicated. Therefore, in the late evolutionary phase, CMPSO may perform slightly poorly than CMPSO-non-aBest does on some UF problems. In order to show more clearly how CMPSO can approximate the PF faster than CMPSO-non-aBest does, we plot the finial nondominated solutions of the two algorithms in Fig. 9 for UF1, UF2, and UF3. These solutions are obtained after 1000 FEs and are obtained by the run that has the minimal IGD value among the 30 runs. The figures confirm that CMPSO approximates the PF faster than CMPSO-non-aBest in the early evolutionary phase.

H. Impacts of Parameter Settings

1) Population Size for Each Swarm: As each swarm optimizes only one objective in CMPSO, we set a relative small population size of 20 particles for each swarm. In this subsection, the population size for each swarm is set to be 40, 60, 80, and 100, respectively, to investigate the impact of population size on the algorithm's performance. The investigations are conducted on ZDT1 whose objective functions are umimodal and on UF1 whose objective functions are multimodal. For CMPSO with different population size, the other parameter settings are still the same as that in Section IV-C, and the maximal number of FEs is still 25 000 for ZDT1 and 3×10^5 for UF1.

The average values of 30 independent runs on the IGD indicator are compared in Fig. 10. The comparisons show that a small population size is efficient enough for CMPSO to obtain good performance, specifically for simple MOPs. This may be due to the contribution of the MPMO technique in reducing the search complexity for each swarm because only one objective is optimized by each swarm. When the population size increases to be large, CMPSO performs even worse, e.g., on the ZDT1 problem. This may be caused by the fact that with the fixed value of maximal number of FEs, a larger population size reduces the evolutionary generation and at last affects the algorithm's performance. However, when solving complicated MOPs, increasing the population size can increase the diversity of the algorithm and can therefore lead to better results, e.g., on the UF1 problem. Nevertheless, a too large population size costs much computational burden in each generation and may therefore weaken the performance when the maximal number of FEs is fixed. By considering both the computational burden and the algorithm's performance, this paper adopts the population size of 20 for each swarm in CMPSO. In general, a population size of 20–60 for each swarm may be promising. The small population size that can bring good performance is the advantage of the CMPSO algorithm.

2) Maximal Number of FEs: As shown in Section IV-D, CMPSO and other MOPSOs perform poorly on ZDT4. It could be due to the fact that ZDT4 is with multimodal objective functions while CMPSO does not have sufficient number of FEs to converge to the true PF. Herein, we keep the other parameters the same as in Section IV-C and set different maximal number of FEs (e.g., 1×10^5, 2×10^5, and 3×10^5) for CMPSO to solve ZDT4. The final nondominated solutions found with different maximal number of FEs are shown in Fig. 11.

When compared with Fig. 4(c), it is clear that by increasing the maximal number of FEs, CMPSO can search the space sufficiently to approximate the true PF well. Moreover, we show the solutions obtained by OMOPSO in all the 30 runs in Fig. 11. When compared with CMPSO, it is clear that CMPSO has stronger global search ability than OMOPSO to approximate the PF. This indicates that the MPMO technique is helpful to enhance MOPSOs' performance.

3) Inertia Weight ω and Acceleration Coefficients c_i: To investigate the impact of ω on MOPSOs' performance, we test different values of ω (e.g., 0.1, 0.3, 0.5, 0.7, and 0.9) on CMPSO, MOCLPSO, and OMOPSO. The investigations are conducted on DTLZ2 with umimodal objective functions and on UF1 with multimodal objective functions.

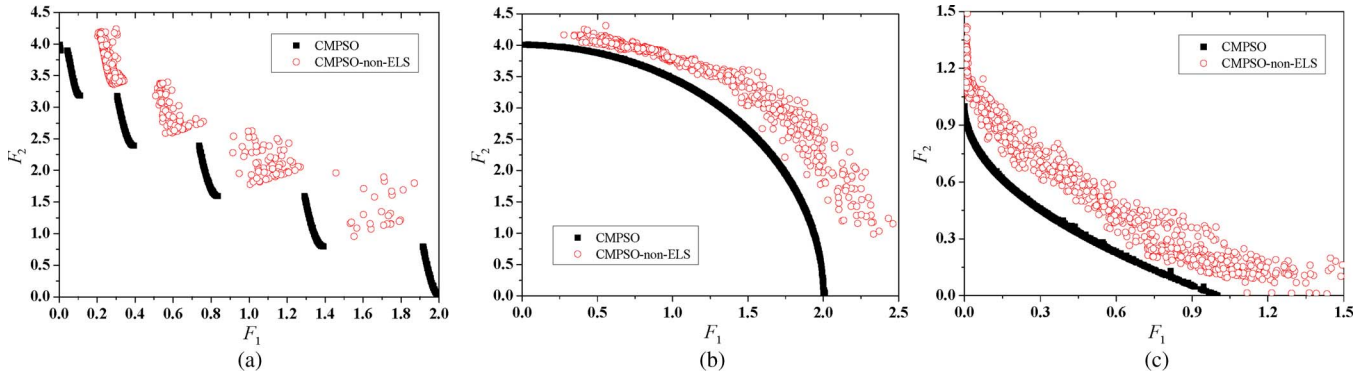


Fig. 7. Final nondominated solutions found by CMPSO and CMPSO-non-ELS in all the 30 runs. (a) WFG2. (b) WFG4. (c) UF2.

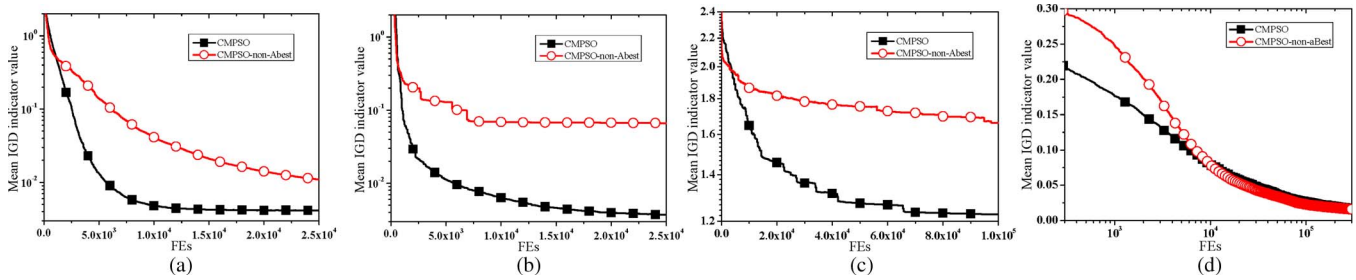


Fig. 8. Mean IGDs of CMPSO and CMPSO-non-aBest during the evolutionary process. (a) ZDT1. (b) ZDT6. (c) WFG1. (d) UF2.

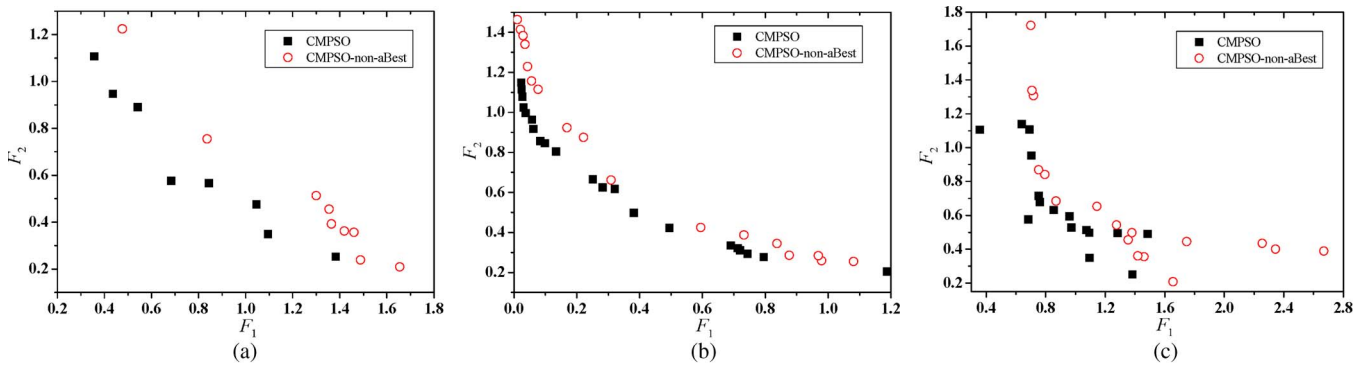


Fig. 9. Final nondominated solutions with minimal IGD value found by CMPSO and CMPSO-non-aBest after 1000 FEs. (a) UF1. (b) UF2. (c) UF3.

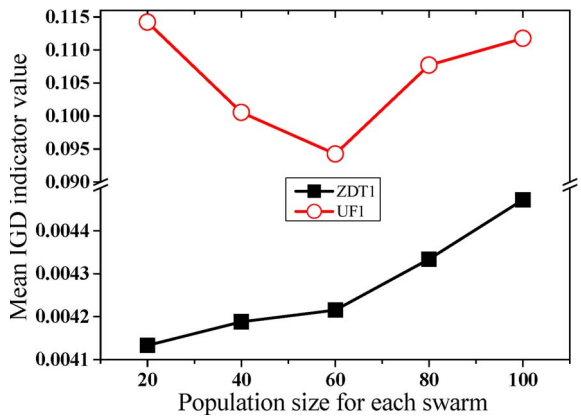


Fig. 10. Mean IGD of CMPSO with different population size.

Fig. 12(a) and (b) shows the mean IGD value of DTLZ1 and UF1 when MOPSOs use different inertia weight values. For DTLZ2 whose objective functions are unimodal, a relatively small ω value would be preferred, while for UF1 whose objective functions are multimodal, a relatively large ω value seems to be preferred. This may be due to the fact that a large ω is helpful for global search, while a small ω is beneficiary for local fine tuning [15]. The results are also compared with the ones obtained by MOPSOs using ω values as in Table II. The figure shows that the parameters in Table II are promising. Moreover, it is evident from Fig. 12(a) and (b) that CMPSO is less sensitive to the ω value, while the other two MOPSOs, particularly MOCLPSO, are affected by the ω value significantly. This is another advantage of the CMPSO algorithm.

The impact of the acceleration coefficients c_i on CMPSO's performance is also investigated on DTLZ2 and UF1, with the results being shown in Fig. 12(c) and (d). The results further confirm that the parameters in Table II are promising and CMPSO is much less sensitive to the c_i value when compared with MOCLPSO and OMOPSO, showing the advantage of CMPSO.

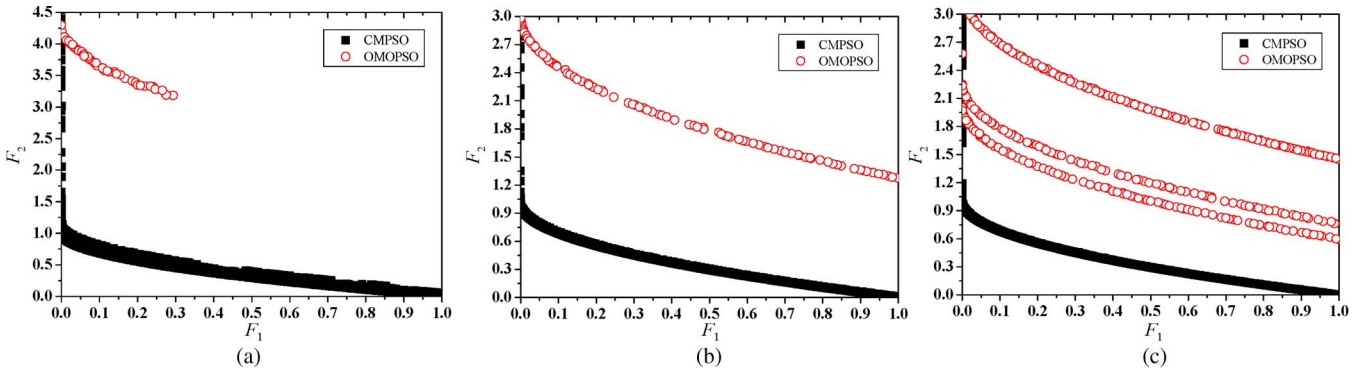


Fig. 11. Final nondominated solutions found by CMPSO and OMOPSO with different maximal FEs on ZDT4. (a) FEs $= 1 \times 10^5$. (b) FEs $= 2 \times 10^5$. (c) FEs $= 3 \times 10^5$.

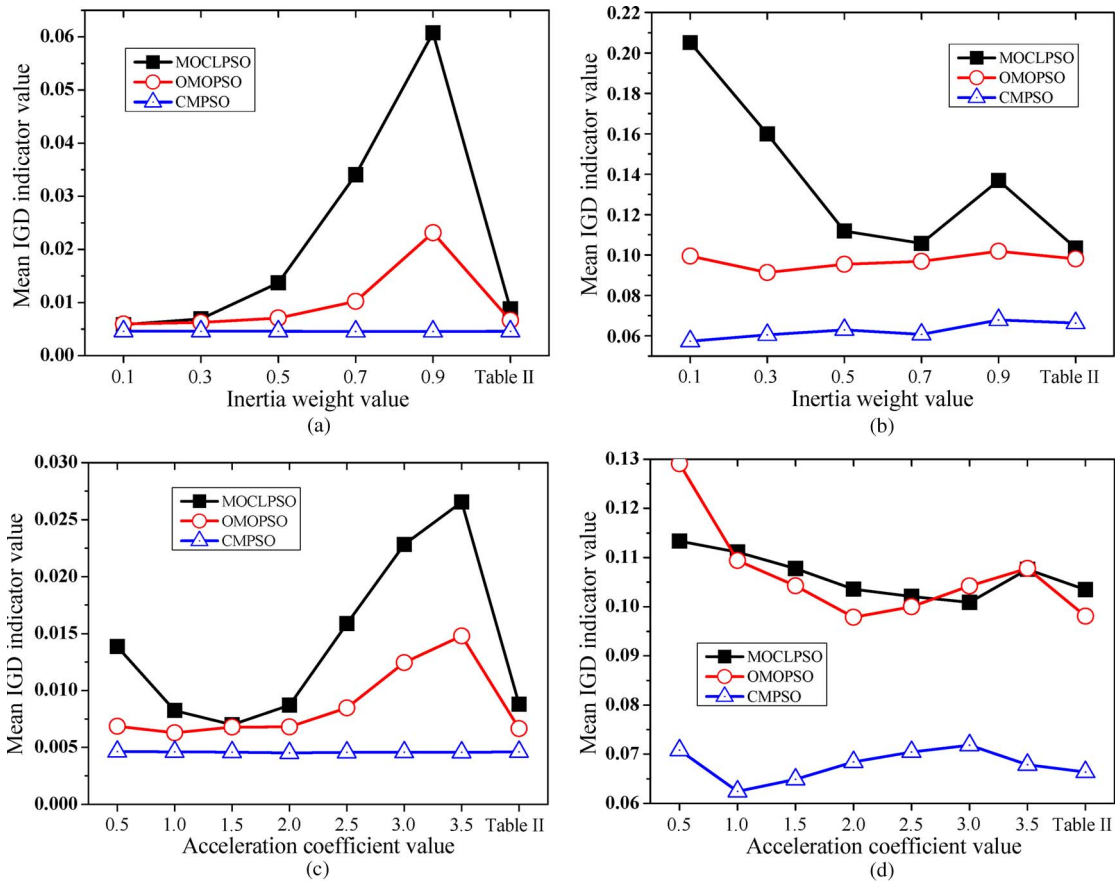


Fig. 12. Mean IGDs on DTLZ2 and UF1 of MOPSOs with different inertia weight ω and different acceleration coefficients c_i. (a) Different ω on DTLZ2. (b) Different ω on UF1. (c) Different c_i on DTLZ2. (d) Different c_i on UF1.

I. Comparisons on More Performance Metrics

In this subsection, we compare our CMPSO with other algorithms on three more performance metrics [57]. The first one is the computational time, the second one is the size of the space covered (SSC) [58], and the third one is the coverage of two sets $C(A, B)$ [58]. For the computational time, all the algorithms are implemented on the Microsoft Visual C++ environment and run on the same machine with four-core (2.4-GHz) CPU, 2.0-GB RAM, and Windows 7 operation system. The SSC and $C(A, B)$ metrics were both proposed by Zitzler and Thiele [58]. The SSC indicator can measure the portion of the hypervolume dominated by the obtained nondominated-solution set A to the hypervolume dominated by the true PF solution set P. Therefore, the larger SSC value reflects a better set of solutions. If an algorithm obtains a set of nondominated solutions spread along the true PF, the SSC value will be one. For the $C(A, B)$ metric, it compares the quality of two sets of nondominated solutions A and B. The calculation is defined as

$$C(A,B) = \frac{|\{b \in B, \exists a \in A : a \text{ dominates } b \text{ or } a \text{ is equal to } b\}|}{|B|}. \quad (10)$$

The value of $C(A, B)$ is within [0, 1]. The larger the $C(A, B)$ value, the better the quality of set A than set B. If $C(A, B) = 1.0$, it means that each solution in set B is

TABLE VII
COMPARISONS ON MORE PERFORMANCE METRICS

Problems		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO
ZDT1	CPU Time (s)	2.560	0.805	1.471	3.833	4.084	1.757	1.390
	SSC (%)	99.9976	99.9046	98.4949	99.9976	99.9240	99.2931	99.9972
	C(CMPSO, −) (%)	0.27	96.51	99.35	9.57	44.50	97.37	
	C(−, CMPSO) (%)	0.17	0	0	3.10	1.00	1.60	
ZDT2	CPU Time (s)	2.093	0.636	1.105	3.957	4.629	1.710	1.500
	SSC (%)	98.0816	99.5240	96.7376	95.9770	99.9861	98.8855	99.9980
	C(CMPSO, −) (%)	7.10	99.24	99.96	24.23	26.97	97.40	
	C(−, CMPSO) (%)	0.93	0.03	0	2.17	1.10	1.80	
ZDT3	CPU Time (s)	2.505	0.800	1.404	3.093	3.403	1.896	0.971
	SSC (%)	99.5489	99.9033	97.2747	99.9886	98.4870	93.4598	99.9842
	C(CMPSO, −) (%)	0	44.55	99.95	1.80	13.03	100	
	C(−, CMPSO) (%)	42.09	7.61	0	39.67	19.31	0	
ZDT4	CPU Time (s)	1.907	1.895	0.802	0.380	1.248	1.536	0.555
	SSC (%)	97.7566	97.1766	97.1227	76.5240	7.2230	0.1614	93.7530
	C(CMPSO, −) (%)	0	0	15.36	0	100	100	
	C(−, CMPSO) (%)	96.20	96.84	78.04	10.72	0	0	
ZDT6	CPU Time (s)	3.013	0.610	0.774	3.974	5.205	2.002	1.379
	SSC (%)	99.9521	99.0670	86.5647	99.9985	99.9980	98.8994	99.9983
	C(CMPSO, −) (%)	68.10	97.49	97.04	2.17	3.53	46.40	
	C(−, CMPSO) (%)	1.60	1.10	0.87	3.60	3.70	2.17	
DTLZ1	CPU Time (s)	18.12	132.0	5.403	2.781	4.276	3.340	3.42
	SSC (%)	99.9991	99.9654	99.9940	0.0368	0	0	99.9777
	C(CMPSO, −) (%)	0	0	3.60	100	100	100	
	C(−, CMPSO) (%)	99.62	16.92	93.58	0	0	0	
DTLZ2	CPU Time (s)	18.53	184.9	5.826	3.915	24.76	3.391	4.004
	SSC (%)	99.9978	99.8804	99.9935	99.9949	99.9968	99.9585	99.9984
	C(CMPSO, −) (%)	4.07	0	1.50	74.97	25.57	100	
	C(−, CMPSO) (%)	4.37	0.27	1.73	0.07	0.93	0	
WFG1	CPU Time (s)	15.60	1.685	3.020	4.407	6.692	3.887	4.483
	SSC (%)	79.4607	76.9657	76.7392	84.0004	82.9965	81.0764	85.0785
	C(CMPSO, −) (%)	99.76	100	100	95.97	99.33	96.90	
	C(−, CMPSO) (%)	0	0	0	0.23	0	0.33	
WFG2	CPU Time (s)	14.34	3.868	3.159	3.512	4.648	3.784	4.050
	SSC (%)	89.5679	80.7958	82.7887	90.7469	90.2127	90.1851	95.1738
	C(CMPSO, −) (%)	100	100	100	100	100	100	
	C(−, CMPSO) (%)	0	0	0	0	0	0	
WFG3	CPU Time (s)	14.30	19.84	3.615	3.452	6.349	4.108	4.609
	SSC (%)	91.7197	75.4405	86.2241	93.2002	92.7822	92.3682	99.7297
	C(CMPSO, −) (%)	100	100	100	100	100	100	
	C(−, CMPSO) (%)	0	0	0	0	0	0	
WFG4	CPU Time (s)	14.14	3.745	3.493	3.632	6.006	4.320	4.677
	SSC (%)	89.2546	71.6783	82.4283	91.0879	90.6234	89.7718	99.5843
	C(CMPSO, −) (%)	100	100	100	100	100	100	
	C(−, CMPSO) (%)	0	0	0	0	0	0	
UF1	CPU Time (s)	64.71	12.32	19.29	10.27	10.46	11.55	9.879
	SSC (%)	99.4065	99.3455	99.8440	98.8283	98.2819	98.2923	99.2738
	C(CMPSO, −) (%)	8.53	16.10	13.93	35.77	35.57	82.20	
	C(−, CMPSO) (%)	2.11	0.93	1.85	0	0	0	
UF2	CPU Time (s)	66.24	81.54	22.18	10.49	25.47	12.17	16.90
	SSC (%)	99.8854	99.6819	99.8812	98.5332	98.8847	99.2368	99.7582
	C(CMPSO, −) (%)	39.83	13.13	8.27	100	99.87	96.23	
	C(−, CMPSO) (%)	2.33	10.54	7.04	0	0	0	
UF3	CPU Time (s)	75.77	9.240	20.76	9.728	13.41	12.01	9.958
	SSC (%)	99.1350	96.5094	99.2635	93.8287	96.0476	98.8900	99.9241
	C(CMPSO, −) (%)	6.30	39.96	9.50	100	100	98.90	
	C(−, CMPSO) (%)	9.20	1.78	23.30	0	0	0.48	
UF4	CPU Time (s)	74.69	14.39	22.81	11.41	17.49	12.87	11.05
	SSC (%)	99.6102	99.7180	99.4867	99.1434	98.5565	99.3828	99.8398
	C(CMPSO, −) (%)	90.70	65.67	91.90	100	100	100	
	C(−, CMPSO) (%)	0	0.27	0.67	0	0	0	
UF5	CPU Time (s)	67.96	7.616	18.46	7.492	8.063	11.80	4.830
	SSC (%)	24.1232	24.3676	23.6859	23.5035	22.1288	20.4741	24.2002
	C(CMPSO, −) (%)	43.40	49.37	64.70	39.51	93.93	99.80	
	C(−, CMPSO) (%)	7.13	6.42	4.54	3.02	0	0	
UF6	CPU Time (s)	67.83	8.052	20.76	8.587	9.151	12.13	4.906
	SSC (%)	97.6132	96.7606	96.4714	94.7340	93.2846	87.5348	96.8186
	C(CMPSO, −) (%)	0.93	79.87	21.61	53.25	77.23	98.83	
	C(−, CMPSO) (%)	23.18	0	28.37	0	0	0	
UF7	CPU Time (s)	62.81	9.881	23.89	9.351	15.76	10.87	7.657
	SSC (%)	98.1942	98.8910	99.9110	98.4841	97.0056	98.1461	98.6551
	C(CMPSO, −) (%)	7.80	14.93	12.23	44.62	32.13	93.13	
	C(−, CMPSO) (%)	3.39	2.42	4.02	0	0.18	0	
Mean CPU Time of 18 Problems		32.62 s	27.43 s	9.90 s	5.79 s	9.51 s	6.40 s	5.35 s
Mean SSC of 18 Problems		0.9241	0.8975	0.9016	0.8548	0.8147	0.8033	0.9399
# of {C(CMPSO,−)>C(−,CMPSO)}		13	15	14	15	16	18	
Algorithms		NSGA-II	GDE3	MOEA/D-DE	MOCLPSO	OMOPSO	VEPSO	CMPSO

dominated by or equal to at least one solution in set A, while $C(A, B) = 0$ means that all the solutions in set B are not dominated by any solution in set A.

The experimental results presented in Table VII indicate the general better performance of CMPSO when compared with other algorithms on these performance metrics. The CPU

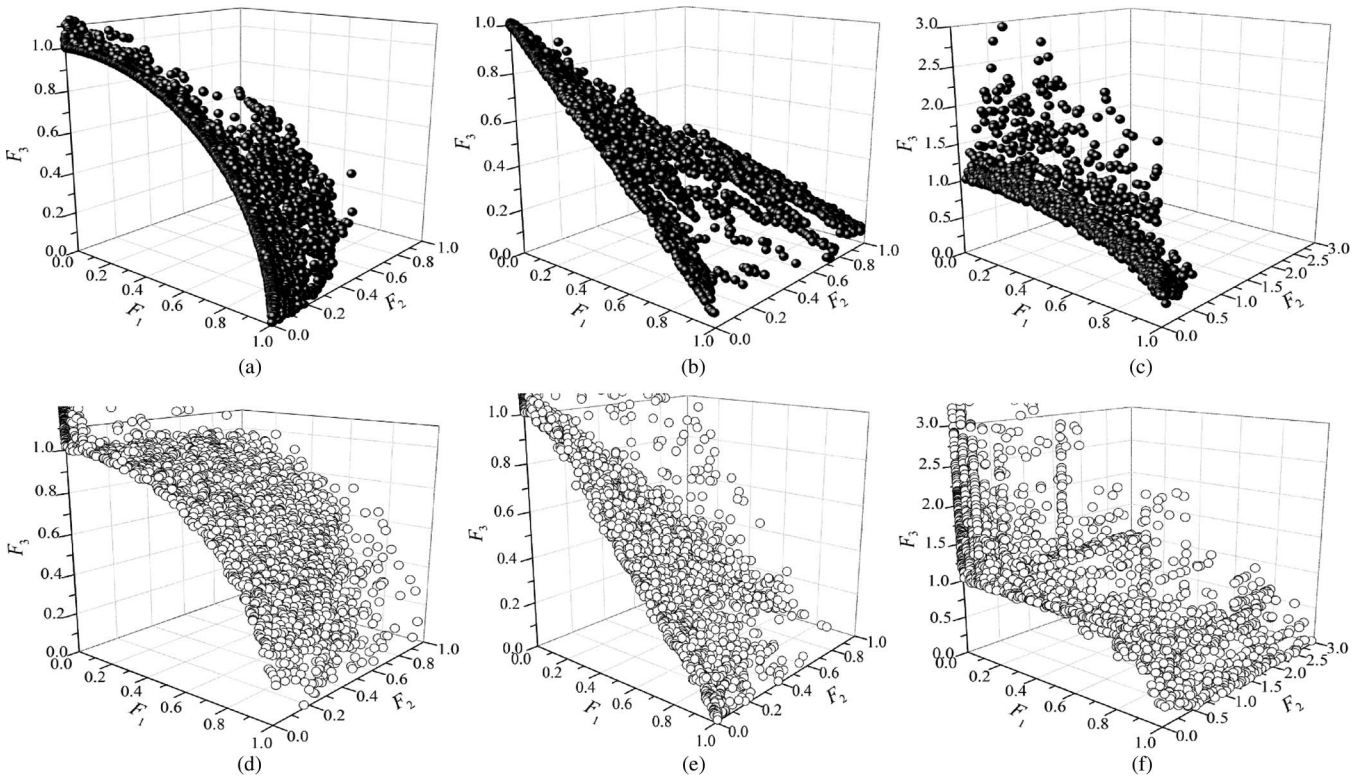


Fig. 13. Final nondominated solutions of the UF problems found by CMPSO and NSGA-II in all the 30 runs. (a) CMPSO (UF8). (b) CMPSO (UF9). (c) CMPSO (UF10). (d) NSGA-II (UF8). (e) NSGA-II (UF9). (f) NSGA-II (UF10).

time is measured in seconds. GDE3, CMPSO, MOEA/D-DE, and VEPSO are fast on the ZDT problems. However, GDE3 costs very large computational burden on the DTLZ and UF problems when compared with CMPSO. In general, GDE3 is fast on the ZDT problems, MOCLPSO is fast on the DTLZ problems, MOEA/D-DE is fast on the WFG problems, and CMPSO is fast on the UF problems. The results show that CMPSO is the fastest algorithm, as indicated by the mean CPU time over all the 18 problems. Moreover, the results show that CMPSO has the largest SSC value on ZDT2, DTLZ2, WFG1, WFG2, WFG3, WFG4, UF2, UF3, and UF6, i.e., 9 out of the 18 problems. In general, the mean SSC value of CMPSO over all the 18 problems is 0.9399, which is the largest (best) result of all the seven algorithms. At last, all the algorithms are compared with CMPSO by both the $C(\text{CMPSO}, -)$ and $C(-, \text{CMPSO})$ values, which are also the mean values of 30 runs. The results show that the nondominated solutions obtained by CMPSO tend to dominate those generated by other algorithms. Totally, CMPSO does better than NSGA-II, GDE3, MOEA/D-DE, MOCLPSO, OMOPSO, and VEPSO on 13, 15, 14, 15, 16, and 18 problems out of all the 18 problems, respectively, indicated by the number of $C(\text{CMPSO}, -) > C(-, \text{CMPSO})$.

J. Experimental Results on MOPs With Three Objectives

In this subsection, the performance of CMPSO on some three-objective MOPs like UF8, UF9, and UF10 [53] is tested, and the results are shown in Fig. 13. In order to make a comparison, the results obtained by NSGA-II are also shown in the figures.

Fig. 13(a)–(c) shows the final obtained nondominated solutions in 30 runs by CMPSO on UF8, UF9, and UF10, respectively, while Fig. 13(d)–(f) shows those obtained by NSGA-II. The figures clearly show that CMPSO outperforms NSGA-II on these problems with more objectives. The solutions obtained by CMPSO can approximate the PF curve surfaces of UF8 and UF9 very well, while most of the solutions obtained by NSGA-II seem to scatter in the area that is dominated by the PF curve surfaces. Therefore, CMPSO not only is demonstrated to be efficient in solving MOP with two objectives but also is promising in solving MOP with three objectives. For MOP with higher objectives such as larger than five, future work will be conducted, and the CMPSO performance will be further tested.

V. Conclusion

A coevolutionary technique named MPMO has been proposed for solving MOPs, and CMPSO with shared archive based on the MPMO technique has been designed in this paper. CMPSO uses as many swarms as the objectives number and lets each swarm focus on optimizing one objective. These swarms work cooperatively and communicate through an external shared archive. CMPSO benefits from the following three aspects when solving MOP. 1) As each swarm focus on optimizing one objective, it can use conventional or any other improved PSO to solve a single-objective problem. Importantly,

the difficulty of fitness assignment can be avoided. 2) As an external shared archive is used to store the nondominated solutions found by different swarms and the shared-archive information is used to guide the particle update, the algorithm can use the whole search information to approximate the whole PF fast. 3) As ELS is performed on the archived solutions in the update process, the algorithm is able to avoid local PFs. This is helpful for MOPs with multimodal objective functions or with complicated Pareto sets.

The performance of the proposed CMPSO has been tested on different sets of MOPs with various objective functions, PFs, and Pareto sets. CMPSO is compared with some state-of-the-art and modern MOEAs and MOPSOs based on the IGD indicator. The experimental results show that CMPSO not only generally outperforms the compared algorithms on ZDT problems but also generally performs remarkably better than all the other algorithms on DTLZ and WFG problems. When dealing with UF problems with complicated Pareto sets, CMPSO is also one of the most promising algorithms. Moreover, the CMPSO performance is tested and compared with other algorithms on more metrics such as CPU time, SSC, and coverage of two sets. Experimental results support the good performance of CMPSO on these metrics. Furthermore, the benefit of the shared archive used in CMPSO is investigated, to demonstrate both the benefit of ELS in bringing in diversity to avoid local PFs and the benefit of the archived information in guiding the particles to approximate the PF fast. At last, the results indicate that the MPMO technique may contribute to reducing the search complexity for each swarm, and a small population size is efficient enough to obtain good performance.

The CMPSO algorithm proposed in this paper is an instantiation of the MPMO technique by using CMPSO. However, as a new and general technique to use multiple populations to tackle with the multiple objectives in MOP, there are still many interesting issues worth further studying in MPMO. Future research work will include the following aspects: 1) applying other algorithms like GA [59], [60], ant colony optimization [61]–[63], DE [64]–[67], EDA, MA [68]–[72], and their improved variants (like enhanced PSO variants [73]–[76]) to realize MPMO for further evaluation of performance of the multiple-population algorithms in solving MOP. Nevertheless, the information-sharing and population communication strategy should be redesigned when new optimization algorithms are used. For example, we can let individuals perform crossover among different populations for GA information sharing and communication; 2) using the MPMO-based multiobjective algorithms to solve MOP with many objectives (e.g., up to or larger than five), and those problems in dynamic [77], [78], noisy [79], and uncertain [80] environments; and 3) applying the new MPMO-based multiobjective algorithms to real-world problems.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and reviewers for their valuable comments and suggestions that have improved the paper's quality.

REFERENCES

[1] E. Masazade, R. Rajagopalan, P. K. Varshney, C. K. Mohan, G. K. Sendur, and M. Keskinoz, "A multiobjective optimization approach to obtain decision thresholds for distributed detection in wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 2, pp. 444–457, Apr. 2010.

[2] S. Y. Shin, I. H. Lee, Y. M. Cho, K. A. Yang, and B. T. Zhang, "EvoOligo: Oligonucleotide probe design with multiobjective evolutionary algorithms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1606–1616, Dec. 2009.

[3] C. K. Ting, C. N. Lee, H. C. Chang, and J. S. Wu, "Wireless heterogeneous transmitter placement using multiobjective variable-length genetic algorithm," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 945–958, Aug. 2009.

[4] M. Delgado, M. P. Cuellar, and M. C. Pegalajar, "Multiobjective hybrid optimization and training of recurrent neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 2, pp. 381–403, Apr. 2008.

[5] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, U.K.: Wiley, 2001.

[6] M. Reyes-Sierra and C. A. Coello Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *Int. J. Comput. Intell. Res.*, vol. 2, no. 3, pp. 287–308, 2006.

[7] K. Deb, S. Agrawal, A. Pratap, 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.

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

[9] M. J. Wade, "The co-evolutionary genetics of ecological communities," *Nat. Rev. Genet.*, vol. 8, no. 3, pp. 185–195, Mar. 2007.

[10] J. Zhang, W. L. Lo, and H. Chung, "Pseudo-coevolutionary genetic algorithms for power electronic circuits optimization," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 36, no. 4, pp. 590–598, Jul. 2006.

[11] C. K. Goh, K. C. Tan, D. S. Liu, and S. C. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *Eur. J. Oper. Res.*, vol. 202, no. 1, pp. 42–54, Apr. 2010.

[12] X. D. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[13] Z. H. Zhan and J. Zhang, "Co-evolutionary differential evolution with dynamic population size and adaptive migration strategy," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2011, pp. 211–212.

[14] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, Australia, 1995, vol. 4, pp. 1942–1948.

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

[16] 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.

[17] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method in multiobjective problems," in *Proc. ACM Symp. Appl. Comput.*, 2002, pp. 603–607.

[18] X. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1677–1681.

[19] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. the 5th Int. Conf. Genet. Algorithms*, 1993, pp. 416–423.

[20] J. Horn, N. Nafploitis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput.*, 1994, pp. 82–87.

[21] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.

[22] X. Li, "A non-dominated sorting particle swarm optimizer for multiobjective optimization," in *Proc. Genetic Evol. Comput. Conf.*, 2003, pp. 37–48.

[23] J. Branke and S. Mostaghim, "About selecting the personal best in multiobjective particle swarm optimization," in *Proc. Conf. Parallel Probl. Solving Nature*, Reykjavik, Iceland, 2006, pp. 523–532.

[24] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, 2003, pp. 26–33.

[25] L. Rachmawati and D. Srinivasan, "Incorporating the notion of relative importance of objectives in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 530–546, Aug. 2010.

[26] I. Karahan and M. Koksalan, "A territory defining multiobjective evolutionary algorithms and preference incorporation," *IEEE Trans. Evol. Comput.*, vol. 14, no. 4, pp. 636–664, Aug. 2010.

[27] E. Zitzler, L. Thiele, and J. Bader, "On set-based multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 58–79, Feb. 2010.

[28] Y. Wang, Z. X. Cai, G. Q. Guo, and Y. R. Zhou, "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 560–575, Jun. 2007.

[29] S. F. Adra, T. J. Dodd, I. A. Griffin, and P. J. Fleming, "Convergence acceleration operator for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 4, pp. 825–847, Aug. 2009.

[30] G. Avigad and A. Moshaiov, "Interactive evolutionary multiobjective search and optimization of set-based concepts," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 4, pp. 1013–1027, Aug. 2009.

[31] A. Lara, G. Sanchez, C. A. Coello Coello, and O. Schütze, "HCS: A new local search strategy for memetic multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 14, no. 1, pp. 112–132, Feb. 2010.

[32] Z. Song and A. Kusiak, "Multiobjective optimization of temporal processes," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 40, no. 3, pp. 845–856, Jun. 2010.

[33] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. Evol. Comput.*, vol. 14, no. 3, pp. 456–474, Jun. 2010.

[34] D. S. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 1, pp. 42–50, Feb. 2007.

[35] B. B. Li and L. Wang, "A hybrid quantum-inspired genetic algorithm for multiobjective flow shop scheduling," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 37, no. 3, pp. 576–591, Jun. 2007.

[36] S. Kukkonen and J. Lampinen, "Performance assessment of generalized differential evolution 3 with a given set of constrained multi-objective test problems," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1943–1950.

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

[38] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.

[39] J. D. Knowles and D. W. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 100–116, Apr. 2003.

[40] J. Branke, H. Schmeck, K. Deb, and R. S. Reddy, "Parallelizing multiobjective evolutionary algorithms: Cone separation," in *Proc. Congr. Evol. Comput.*, Portland, OR, 2004, pp. 1952–1957.

[41] 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.

[42] G. Ewald, W. Kurek, and M. A. Brdys, "Grid implementation of a parallel multiobjective genetic algorithm for optimized allocation of chlorination stations in drinking water distribution systems: Chojnice case study," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 4, pp. 497–509, Jul. 2008.

[43] W. F. Leong and G. G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1270–1293, Oct. 2008.

[44] G. G. Yen and W. F. Leong, "Dynamic multiple swarms in multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 39, no. 4, pp. 890–911, Jul. 2009.

[45] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genet. Algorithms*, 1985, pp. 93–100.

[46] K. E. Parsopoulos, D. K. Tasoulis, and M. N. Vrahatis, "Multiobjective optimization using parallel vector evaluated particle swarm optimization," in *Proc. Int. Conf. AIA*, 2004, pp. 823–828.

[47] C. K. Chow and H. T. Tsui, "Autonomous agent response learning by a multi-species particle swarm optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2004, pp. 778–785.

[48] R. A. Krohling and L. dos Santos Coelho, "Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1407–1416, Dec. 2006.

[49] 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.

[50] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, Jun. 2000.

[51] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 825–830.

[52] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multiobjective test problem toolkit," in *Lecture Notes in Computer Science*, vol. 3410. Berlin, Germany: Springer-Verlag, Mar. 2005, pp. 280–295.

[53] Q. Zhang, A. Zhou, S. Z. Zhao, P. N. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 1–30.

[54] V. L. Huang, P. N. Suganthan, and J. J. Liang, "Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems," *Int. J. Intell. Syst.*, vol. 21, no. 2, pp. 209–226, Feb. 2006.

[55] M. R. Sierra and C. A. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance," in *Lecture Notes in Computer Science*, vol. 3410. Berlin, Germany: Springer-Verlag, Mar. 2005, pp. 505–519.

[56] J. J. Durillo, J. Garcéa-Nieto, A. J. Nebro, C. A. Coello Coello, F. Luna, and E. Alba, "Multi-objective particle swarm optimizers: An experimental comparison," in *Proc. 5th Int. Conf. Evol. Multi-Criterion Optim.*, 2009, pp. 495–509.

[57] J. Molina, M. Laguna, R. Marté, and R. Caballero, "SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization," *INFORMS J. Comput.*, vol. 19, no. 1, pp. 91–100, Jan. 2007.

[58] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[59] J. Zhang, H. Chung, and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 326–335, Jun. 2007.

[60] X. M. Hu, J. Zhang, Y. Yu, Y. L. Li, X. N. Luo, and Y. H. Shi, "Hybrid genetic algorithm using a forward encoding scheme for lifetime maximization of wireless sensor networks," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 766–781, Oct. 2010.

[61] Z. H. Zhan, J. Zhang, Y. Li, O. Liu, S. K. Kwok, W. H. Ip, and O. Kaynak, "An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 2, pp. 399–412, Jun. 2010.

[62] W. N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 29–43, Jan. 2009.

[63] Y. Lin, J. Zhang, H. Chung, Y. Li, and Y. H. Shi, "An ant colony optimization approach for maximizing the lifetime of heterogeneous wireless sensor networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 3, pp. 408–420, May 2012.

[64] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 54–65, May 2010.

[65] J. H. Zhong, M. Shen, J. Zhang, H. S.-H. Chung, Y. H. Shi, and Y. Li, "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Trans. Evol. Comput.*, 2012, to be published.

[66] Z. H. Zhan and J. Zhang, "Self-adaptive differential evolution based on PSO learning strategy," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2010, pp. 39–46.

[67] Z. H. Zhan and J. Zhang, "Enhance differential evolution with random walk," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2012, pp. 1513–1514.

[68] Y. S. Ong, M. Lim, and X. S. Chen, "Memetic computation—Past, present & future," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 24–31, May 2010.

[69] R. Meuth, E. Saad, D. Wunsch, and J. Vian, "Memetic mission management," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 32–40, May 2010.

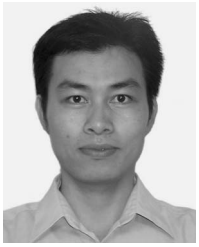
[70] Z. Zhu, S. Jia, and Z. Ji, "Towards a memetic feature selection paradigm," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 41–53, May 2010.

[71] L. C. Jiao, M. G. Gong, S. Wang, B. Hou, Z. Zheng, and Q. D. Wu, "Natural and remote sensing image segmentation using memetic computing," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 78–91, May 2010.

[72] J. Zhang, Z. H. Zhan, Y. Lin, N. Chen, Y. J. Gong, J. H. Zhong, H. Chung, Y. Li, and Y. H. Shi, "Evolutionary computation meets machine learning: A survey," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 68–75, Nov. 2011.

[73] Z. H. Zhan, J. Zhang, Y. Li, and Y. H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[74] W. N. Chen, J. Zhang, H. Chung, W. L. Zhong, and Y. H. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.
[75] W. N. Chen, J. Zhang, Y. Lin, N. Chen, Z. H. Zhan, H. Chung, Y. Li, and Y. H. Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, 2012, to be published.
[76] Y. J. Gong, J. Zhang, H. Chung, W. N. Chen, Z. H. Zhan, Y. Li, and Y. H. Shi, "An efficient resource allocation scheme using particle swarm optimization," *IEEE Trans. Evol. Comput.*, 2012, to be published.
[77] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 425–442, Oct. 2004.
[78] C. K. Goh and K. C. Tan, "A competitive-cooperative coevolutionary paradigm for dynamic multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 1, pp. 103–127, Feb. 2009.
[79] C. K. Goh and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 354–381, Jun. 2007.
[80] K. Deb, S. Gupta, D. Daum, J. Branke, A. K. Mall, and D. Padmanabhan, "Reliability-based optimization using evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1054–1074, Oct. 2009.

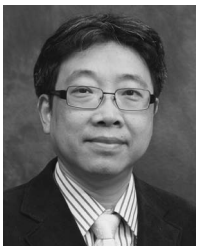


Zhi-Hui Zhan (S'09) received the Bachelor's degree in computer science and technology from Sun Yat-Sen University, Guangzhou, China, in 2007, where he is currently working toward the Ph.D. degree.

His current research interests include particle swarm optimization, ant colony optimization, genetic algorithms, differential evolution, brain storm optimization, and their applications in real-world problems.

Jingjing Li is currently working toward the Ph.D. degree at Hong Kong Polytechnic University, Kowloon, Hong Kong.

Her research interests are mainly focused on evolutionary algorithm, energy efficient routing, and object tracking for wireless sensor networks.



Jiannong Cao (M'93–SM'05) received the B.Sc. degree in computer science from Nanjing University, Nanjing, China, and the M.Sc. and Ph.D. degrees in computer science from Washington State University, Pullman.

He is currently a Chair Professor and the Head of the Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong. He has coauthored three books, coedited nine books, and published over 300 papers in major international journals and conference proceedings. His research interests include parallel and distributed computing, computer networks, mobile and pervasive computing, fault tolerance, and middleware.

Dr. Cao is a Senior Member of China Computer Federation and a member of Association for Computing Machinery. He was the Coordinator in Asia and is currently the Chair of the Technical Committee on Distributed Computing of the IEEE Computer Society. He has served as an Associate Editor and a member of the editorial boards of many international journals, including the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE NETWORK, *Pervasive and Mobile Computing Journal, Wireless Communications and Mobile Computing, Peer-to-Peer Networking and Applications*, and the *Journal of Computer Science and Technology*. He has also served as a Chair and member of organizing/program committees for many international conferences, including PERCOM, INFOCOM, ICDCS, IPDPS, ICPP, RTSS, DSN, ICNP, SRDS, MASS, PRDC, ICC, GLOBECOM, and WCNC.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002.

From 2003 to 2004, he was a Brain Korean 21 Postdoctoral Fellow at the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea. Since 2004, he has been with Sun Yat-Sen University, Guangzhou, China, where he is currently a Professor and a Ph.D. Supervisor with the Department of Computer Science. He has authored seven research books and book chapters, and over 100 technical papers in his research areas. His research interests include computational intelligence, data mining, wireless sensor networks, operations research, and power electronic circuits.

Dr. Zhang received the award of National Science Fund for Distinguished Young Scholars of China from the National Natural Science Foundation of China in 2011. He also received 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 INDUSTRIAL ELECTRONICS, and the IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE. He is the founding and current Chair of the IEEE Guangzhou Subsection and the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapter.

Henry Shu-Hung Chung (M'95–SM'03) received the B.Eng. and Ph.D. degrees in electrical engineering from The Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1991 and 1994, respectively.

Since 1995, he has been with the City University of Hong Kong, Kowloon, where he is currently the Associate Dean of the College of Science and Engineering and a Professor with the Department of Electronic Engineering. He has authored seven research book chapters and over 280 technical papers, including 130 refereed journal papers in his research areas, and is a holder of 16 patents. His current research interests include time- and frequency-domain analysis of power electronic circuits, switched-capacitor-based converters, random switching techniques, control methods, digital audio amplifiers, soft-switching converters, electronic ballast design, and circuit design with evolutionary computation techniques.

Dr. Chung is currently an Associate Editor of the IEEE TRANSACTIONS ON POWER ELECTRONICS and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II.

Yu-Hui Shi (SM'98) received the Ph.D. degree in electronic engineering from Southeast University, Nanjing, China, in 1992.

He is currently a Professor with the Department of Electrical and Electronic Engineering, Xi'an Jiaotong–Liverpool University, Suzhou, China. He also holds the position of the Director of the Research and Postgraduate Office, Xi'an Jiaotong–Liverpool University. Before joining Xi'an Jiaotong–Liverpool University, he was with Electronic Data Systems Corporation, Indianapolis, IN. His main research interests include the areas of computational intelligence techniques (including swarm intelligence) and their applications.

Dr. Shi is the Editor-in-Chief of the *International Journal of Swarm Intelligence Research* and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He is the Chair of the IEEE Chief Information Officer Task Force on Swarm Intelligence.