European Journal of Operational Research 247 (2015) 732–744

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Continuous Optimization

A novel multi-objective particle swarm optimization with multiple search strategies

Qiuzhen Lin, Jianqiang Li*, Zhihua Du, Jianyong Chen, Zhong Ming

College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, PR China

ARTICLE INFO

Article history:
Received 23 August 2014
Accepted 30 June 2015
Available online 8 July 2015

Keywords:
Multiple objective programming
Particle swarm optimization
Evolutionary computation
Multiple search strategies

ABSTRACT

Recently, multi-objective particle swarm optimization (MOPSO) has shown the effectiveness in solving multi-objective optimization problems (MOPs). However, most MOPSO algorithms only adopt a single search strategy to update the velocity of each particle, which may cause some difficulties when tackling complex MOPs. This paper proposes a novel MOPSO algorithm using multiple search strategies (MMOPSO), where decomposition approach is exploited for transforming MOPs into a set of aggregation problems and then each particle is assigned accordingly to optimize each aggregation problem. Two search strategies are designed to update the velocity of each particle, which is respectively beneficial for the acceleration of convergence speed and the keeping of population diversity. After that, all the non-dominated solutions visited by the particles are preserved in an external archive, where evolutionary search strategy is further performed to exchange useful information among them. These multiple search strategies enable MMOPSO to handle various kinds of MOPs very well. When compared with some MOPSO algorithms and two state-of-the-art evolutionary algorithms, simulation results show that MMOPSO performs better on most of test problems.

1. Introduction

In many real-world engineering applications, the problem that needs to optimize multiple objectives simultaneously is often encountered, which is called multi-objective optimization problems (MOPs) (Deb, Pratap, Agarwal, & Meyarivan, 2002; Ishibuchi & Murata, 1998; Samanlioglu, 2013). For example, the goals in job shop scheduling are commonly required to minimize the makespan, total workload, and critical workload, while the targets in product design are certainly needed to minimize the cost of product and optimize its quality. Since the conflicts exist among the objectives, the improvement of one objective may deteriorate other objectives and resultantly it generates a set of equally-optimal solutions, which is termed *Pareto-optimal set* (*PS*). The corresponding mapping of *PS* in objective space is termed *Pareto-optimal front* (*PF*). As the size of *PF* may be infinite, it is impractical to find out all the Pareto-optimal solutions. Thus, an important job of MOPs is to obtain a finite size of *PS* that is distributed uniformly along the *PF*, which supports the decision maker to select the appropriate solutions for different practical cases (Lin & Chen, 2013; Zhang & Li, 2007).

Currently, nature-inspired metaheuristic algorithms have been recognized to be well suitable for solving MOPs since they can handle some complex problems that are characterized with multimodality, nonlinearity, and discontinuity (Jones, Mirrazavi, & Tamiz, 2002). Among them, particle swarm optimization (PSO) is an interesting nature-inspired algorithm that mimics the social cooperative and competitive behavior of bird flocking and fish schooling (Kennedy & Eberhart, 1995). Due to the fast convergence speed and easy implementation, it has attracted a great interest of researchers and been designed for solving many single-objective optimization problems (SOPs) and various engineering applications (Dang et al., 2013; Nayeri, Yang, & Elsherbeni, 2013; Unler & Murat, 2010). The promising results provided by PSO for solving SOPs validate its effectiveness and efficiency to locate the optimal results in a large and complex problem landscape. This motivates the researchers to extend PSO for MOPs and plenty of multi-objective PSO (MOPSO) algorithms are presented accordingly (Moubayed, Pertovski, & McCall, 2014; Coello Coello, Pulido, & Lechuga, 2004; Goh, Tan, Liu, & Chiam, 2010; Zhan et al., 2013). Generally, most of the existing MOPSO algorithms can be classified into two categories. The first class embeds the Pareto dominance relationship into PSO, which is used to determine the personal best and global best particles (Nebro et al., 2009; Sierra & Coello Coello, 2005; Wang & Yang, 2010). The second kind adopts decomposition approach to transform MOPs into a set of SOPs, where traditional PSO can be directly applied to solve MOPs (Moubayed, Pertovski, & McCall, 2010; Martinez & Coello Coello, 2011; Peng & Zhang, 2008). These MOPSO algorithms perform very well in solving

* Corresponding author. Tel.: +86 75526534407; fax: +86-75526534078.
E-mail address: lijq@szu.edu.cn (J. Li).

http://dx.doi.org/10.1016/j.ejor.2015.06.071
0377-2217/

some MOPs. However, when tackling the complex MOPs characterized with multimodality and the existence of many local *PFs*, *e.g.*, WFG1 (Huband, Barone, While, & Hingston, 2005) and DTLZ3 (Deb, Thiele, Laumanns, & Zitzler, 2005), most MOPSO algorithms fail to effectively approach the true *PF*. This is mainly because they only adopt a single search strategy to update the velocity of each particle, which may lack the capabilities to tackle some kinds of complex MOPs.

To repair this weakness, multiple search strategies may be an alternative technology as it has been studied experimentally in PSO for solving SOPs and proven to be an effective and efficient approach to enhance the capabilities of PSO when handling various types of SOPs (Hu, Wu, & Weir, 2013; Li, Yang, & Nguyen, 2012; Zuo, Zhang, & Tan, 2014). Inspired by the reported multiple search strategies for SOPs, it is reasonable to believe that multiple search strategies can be applied in MOPSO to further improve its convergence speed and the robustness when dealing with different kinds of MOPs. Therefore, a novel MOPSO algorithm with multiple search strategies is presented in this paper, called MMOPSO. Decomposition approach is adopted in MMOPSO to decompose MOPs into a set of SOPs and then each particle is assigned to optimize each SOP. Two search strategies for updating the particle's velocity are designed to accelerate the convergence speed and maintain the population diversity respectively. Their cooperation is controlled by a pre-defined threshold. All the non-dominated solutions visited by the particles are stored in a finite-size external archive. Once the external archive is full, only the non-dominated solutions with bigger crowding-distance values will be remained, which are considered to be the elitist solutions and good representatives of the entire *PF*. To let the elitist information be shared among the external archive, an evolutionary search strategy, composed by simulated binary crossover (SBX) and polynomial mutation (PM), is performed, which enhances the exploratory capabilities of MMOPOS. When compared with the existing MOPSO algorithms, the novelty of MMOPSO can be described as follows.

(1) Different from the single search pattern adopted in most MOPSO algorithms, two search strategies are designed in MMOPSO for updating the velocity of each particle, which are aimed at accelerating the convergence speed and maintaining the population diversity respectively. Their executions are determined by a pre-defined threshold to retain the balance of exploitation and exploration.
(2) An evolutionary search strategy is run on the external archive of PSO, which is beneficial for the information exchange among the elitist individuals. The evolutionary operators can provide another search power for PSO and remedy the weaknesses of PSO-based search when handling some difficult MOPs.
(3) New definitions of personal-best and global-best particles are given in MMOPSO. Traditionally, personal-best and global-best particles are the best ones visited by each particle and the swarm respectively. Whereas, in MMOPSO, as decomposition approach is adopted to transform MOPs into a set of SOPs, personal-best and global-best particles are respectively considered to be the best values of each aggregation problem and all SOPs. Therefore, MMOPSO can focus on optimizing each aggregation problem by using PSO search.

The advantages of multiple search strategies will be investigated and validated by the experimental studies. Total 24 standard benchmark problems, including Fonseca (Fonseca & Flemming, 1998), Kursawe (1990), Schaffer (1985), ZDT (Zitzler, Deb, & Thiele, 2000), WFG (Huband et al., 2005) and DTLZ (Deb et al., 2005) series test problems, are utilized to evaluate the comprehensive performance of MMOPSO. When compared with some MOPSO algorithms and two state-of-the-art multi-objective evolutionary algorithms (MOEAs), *e.g.*, DDMOPSO (Moubayed et al., 2014), CMPSO (Zhan et al., 2013), SMPSO (Nebro et al., 2009), dMOPSO (Martinez & Coello Coello, 2011), OMOPSO (Sierra & Coello Coello, 2005), NSGA-II (Deb et al., 2002) and MOEA/D (Li & Zhang, 2009), MMOPSO performs better on most of test problems when considering both of the convergence speed and population diversity.

The rest of this paper is organized as follows. Section 2 introduces the related background, including some important terms of MOPs, decomposition approach, traditional PSO and the existing MOPSO algorithms. In Section 3, the details of MMOPSO are described, where the framework of MMOPSO and multiple search approaches are illustrated. The experimental studies are given in Section 4, which compare the performance of MMOPSO with various multi-objective optimization algorithms and analyze the advantages of multiple search strategies in MMOPSO. At last, conclusions are summarized in Section 5.

2. Related work

2.1. Multi-objective optimization problems

A continuous and unconstrained multi-objective optimization problem can be formulated as follows.

$$\min_{x\in\Omega} F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \tag{1}$$

where $x = (x_1, x_2, \ldots, x_n)$ is a n-dimensional decision vector bounded in the *decision space* Ω, m is the number of objective functions and the mapping function F: $\Omega \to R^m$ defines m objective functions bounded in the *objective space* R^m. Since the objectives often contradict each other, the improvement of one objective may deteriorate other objectives. Therefore, the output of MOPs is generally a set of equally-optimal solutions, which can be determined by Pareto optimality (Bosman & Thierens, 2003).

Definition 1. *(Pareto-dominance)*: A decision vector x is said to dominate another decision vector y (noted as $x \succ y$) if and only if

$$(\forall i \in \{1, 2, \ldots, m\} : f_i(x) \le f_i(y)) \wedge (\exists j \in \{1, 2, \ldots, m\} : f_j(x) < f_j(y)) \tag{2}$$

Definition 2. *(Pareto-optimal)*: A solution x is said to be Pareto-optimal if and only if

$$\neg\exists y \in \Omega : y \succ x. \tag{3}$$

Definition 3. *(Pareto-optimal set)*: The set ***PS*** includes all Pareto-optimal solutions, defined as

$$\boldsymbol{PS} = \{x | \neg\exists y \in \Omega : y \succ x\}. \tag{4}$$

Definition 4. *(Pareto-optimal front)*: The set ***PF*** includes the values of all the objective functions corresponding to the Pareto-optimal solutions in ***PS***.

$$\boldsymbol{PF} = \{F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T | x \in \boldsymbol{PS}\}. \tag{5}$$

2.2. Decomposition approach

Recently, decomposition approach is widely embedded into nature-inspired metaheuristic for solving MOPs (Gong et al., 2014; Liu, Gu, & Zhang, 2014). It is based on the facts that a Pareto-optimal solution for MOPs, under some mild conditions, could be an optimal solution of a scalar optimization problem, whose optimization target is an aggregation of all the objectives. Therefore, the finding of *PF* can be decomposed into a set of SOPs (Li & Zhang, 2009; Zhang & Li, 2007). Currently, the popular decomposition approaches include the weighted sum, Tchebycheff and boundary intersection approaches. Among them, boundary intersection method has shown certain advantages over the other two approaches as discussed in (Martinez & Coello Coello, 2011; Zhang & Li, 2007). Thus, boundary intersection method is adopted in MMOPSO, which uses the pre-defined weighted vectors λ and a penalty value θ to minimize the distance d_1 to the

utopia vector and the direction error to the weighted vector d_2 from the solution $F(x)$ in the objective space, as defined by

$$\min_{x\in\Omega} g(x|\lambda, z^*) = d_1 + \theta d_2 \tag{6}$$

where

$$d_1 = \frac{\|(F(x) - z^*)^T\lambda\|}{\|\lambda\|} \tag{7}$$

$$d_2 = \left\| (F(x) - z^*) - d_1 \frac{\lambda}{||\lambda||} \right\| \tag{8}$$

and $z^* = (z_1^*, z_2^*, \ldots, z_m^*)$ is the reference point, *i.e.*, $z_i^* = \min\{f_i(x)|x \in \Omega\}$ for each $i = 1, 2, \ldots, m$. In practical implementation, as z^* is unavailable in advance, it is usually replaced by the minimal value of each objective found by the algorithms so far. The generation approaches for uniform weighted vectors λ have been introduced in (Li & Zhang, 2009; Zhang & Li, 2007).

2.3. Particle swarm optimization

Particle swarm optimization is an interesting nature-inspired metaheuristic originally proposed by Kennedy and Eberhart (1995) for dealing with global optimization problems. By simulating the movement rules of bird flocking and fish schooling, it is very capable for locating the optimal value in a large searching space. In PSO, a swarm is composed by a certain number of particles. Each particle represents a potential solution for the optimization problem, which is characterized by its position and moving velocity. Here, it is assumed that there are N particles in a swarm. When searching an n-dimensional hyperspace, the position of particle i ($i = 1, 2, \ldots, N$) indicates the solution location in search space, as represented by $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The positional movement of particle i is recorded using its velocity, as described by $v_i = (v_{i1}, v_{i2}, \ldots, v_{in})$. Each particle i will memorize its historically best position as noted by $pbest_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ and the best one among all $pbest_i$ in a swarm is acknowledged as the globally best position $gbest$. Each particle i is evolved by exploiting positional information from the selected global leader and its own personal best to update its velocity and position values, as expressed in Eqs. (9) and (10).

$$v_i(t+1) = wv_i(t) + c_1r_1(x_{pbest_i} - x_i(t)) + c_2r_2(x_{gbest} - x_i(t)) \tag{9}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{10}$$

where t is the iteration number, w is the inertial weight, c_1 and c_2 are two learning factors from the personal and global best particles respectively, r_1 and r_2 are two random numbers generated uniformly in the range [0, 1].

2.4. Existing MOPSO algorithms

Particle swarm optimization is originally designed for solving SOPs. To extend PSO for tackling MOPs, Pareto ranking method or decomposition approach is embedded into PSO. Thus, the existing MOPSO algorithms can be generally classified into two categories. The first class uses Pareto ranking to determine the personal best and global best particles. The global best particles are generally the non-dominated solutions found during the particle movement and they can be exploited to guide the particle swarm to approach the entire *PF*. The reported MOPSOs, such as OMOPSO (Sierra & Coello Coello, 2005) and SMPSO (Nebro et al., 2009), belong to this category. The second type adopts decomposition approach for transforming MOPs into a set of SOPs and then PSO can be directly applied to solve each SOP. This kind of MOPSOs can exploit the reported technologies of PSO to better solve MOPs. The representatives of these MOPSO algorithms include SDMOPSO (Moubayed et al., 2010), dMOPSO (Martinez & Coello Coello, 2011), CMPSO (Zhan et al., 2013) and DDMOPSO (Moubayed et al., 2014). All of these representative MOPSOs are introduced briefly as follows.

OMOPSO is proposed by Sierra and Coello Coello (2005), which uses Pareto dominance and crowding-distance information to identify the list of leader solutions. To enhance the search capability, two mutation operators, *i.e.*, uniform and non-uniform mutations, are respectively executed to balance the abilities of exploration and exploitation. Moreover, an external archive is exploited to collect all the non-dominated solutions visited by the swarm and the concept of ε-dominance is utilized to limit the size of this archive.

SMPSO is an improved version of OMOPSO as designed by Nebro et al. (2009), which embeds a velocity construction procedure in the movement of particles to prevent the so-called "swarm explosion" effect (Clerc & Kennedy, 2002) in OMOPSO. Thus, SMPSO is able to produce new effective particles in the cases that the velocity of particle becomes too high. Besides that, polynomial mutation is performed after PSO search as a turbulence factor and an external archive is used to preserve a number of the historically found non-dominated solutions.

MOPSO/D is reported by Peng and Zhang (2008), which may be the first attempt to embed decomposition approach into MOPSO. It follows the framework of MOEA/D (Zhang & Li, 2007) and replaces the genetic search method with the traditional PSO search approach. The updates of personal and global particles are fully decided by the aggregation values of all objectives. After that, a turbulence operator is performed and an external archive based on ε-dominance is used to collect a number of non-dominated solutions that are historically found during the PSO search.

SDMOPSO is an enhanced algorithm from MOPSO/D as designed by Moubayed et al. (2010), which tackles the drawback of MOPSO/D by fully exploiting the salient properties of neighborhood relations in PSO. The particle's global best is only picked from the neighboring solutions and each particle is only associated with a unique weight vector that gives the best scalar aggregated fitness value. Moreover, a crowding archive is also adopted in SDMOPSO to maintain the diversity of the swarm leaders.

dMOPSO is presented by Martinez and Coello Coello (2011), which is fully dependent on decomposition approach to solve MOPs. The position of each particle is updated using a set of global particles, which are determined based on the scalar aggregated values. The distinct feature of dMOPSO is that a memory re-initialization procedure is used when the particle exceeds a certain age, which is aimed at maintaining the diversity of the swarm and avoiding the trap in local *PFs*. However, as pointed out by Moubayed et al. (2014), the absence of dominance relation in dMOPSO may lead to the fail to cover the entire *PF* in some complex MOPs.

CMPSO is designed by Zhan et al. (2013), which is a novel coevolutionary technique for PSO to solve MOPs. It provides a simple and straightforward way to solve MOPs by letting each swarm correspond with each objective. An external shared archive is used to store all the visited non-dominated solutions and allow the information exchange among the elitist individuals. Two novel approaches are presented to enhance its performance. The first method embeds the elitist information from the shared archive to update the particle's velocity, while the second approach presents an elitist learning strategy for archive update to improve the swarm diversity and avoid the trap in local *PFs*.

The original DDMOPSO is proposed by Moubayed, Pertovski, and McCall (2012), which integrates both of dominance and decomposition approaches for solving MOPs. Afterward, an improved version is also presented by the same authors (Moubayed et al., 2014), which can fast converge to the true *PF* without using the genetic operators. It proposes a new mechanism for the selection of the particle leaders and a novel archiving technique that collects the non-dominated particles based on the crowding-distance values in both objective and solution spaces.

Inspired by the above MOPSO algorithms, the proposed MMOPSO algorithm also decomposes MOPs into a set of aggregation problems and adopts a crowding archive to preserve a number of the non-dominated solutions. However, when compared with the above MOPSO algorithms, the distinct features of MMOPSO include two search strategies that are designed to update the velocity of each particle, an evolutionary search strategy that is performed on the archive to exchange beneficial information among the elitist individuals, a new definition of the personal best and global best particles. All of the new features improve MMOPSO on both of the convergence speed and swarm diversity, which will be analyzed and discussed in the experimental studies.

3. The proposed MMOPSO algorithm

The proposed MMOPSO algorithm is based on decomposition approach to transform MOPs into a set of scalar aggregation problems, which adopts boundary intersection method as introduced in Section 2.2. Each particle in MMOPSO is aimed at optimizing each aggregation problem by updating its flight velocity and then all the non-dominated solutions visited by the particles are maintained in an external finite-size archive. Once the archive is full, the non-dominated solutions with bigger crowding-distance values will be remained. In the following subsections, the main procedures of MMOPSO, such as two search strategies for velocity update, evolutionary search on the external archive and archive update, are respectively described. At last the complete MMOPSO algorithm is illustrated.

3.1. Two search strategies for velocity update

In traditional PSO algorithm, the velocity and position values of the particles are usually updated using the positional information of the personal-best and global-best particles, as defined in Eqs. (9) and (10). However, as pointed out above, this single search pattern may cause some difficulties when solving some complex MOPs. Therefore, inspired by the multiple search patterns reported for solving SOPs (Hu et al., 2013; Li et al., 2012; Zuo et al., 2014), two velocity update equations are incorporated into MMOPSO, which are respectively used for exploitation and exploration in search space. They are cooperated with the decomposition approach, attempting to optimize each aggregation problem. Assuming that there are N particles in a swarm, for each particle i ($i = 1, 2, \ldots, N$), it associates with a unique weight vector λ_i used in Eqs. (6)–(8). In MMOPSO, the velocity of particle i ($i = 1, 2, \ldots, N$) is updated as defined in Eq. (11) or (12).

$$v_i(t+1) = wv_i(t) + c_1 r_1 (x_{pbest_i} - x_i(t)) \tag{11}$$

$$v_i(t+1) = wv_i(t) + c_2 r_2 (x_{gbest} - x_i(t)) \tag{12}$$

where t is the iteration number, w is the inertial weight, c_1 and c_2 are two learning factors, and r_1 and r_2 are two uniformly distributed random numbers in [0, 1]. It is noted that x_{pbest_i} in Eq. (11) is picked from the solutions among the external archive $\mathbf{A} = \{A_1, A_2, \ldots, A_{|\mathbf{A}|}\}$, which gives the best value of each aggregation problem corresponding with the weight vector λ_i (Moubayed et al., 2014). The pseudo-code for the selection of each x_{pbest_i} is described in Fig. 1.

On the other hand, as the solutions in external archive **A** are all non-dominated, they can be considered to be the global-best values for MOPs. Thus, x_{gbest} is randomly selected from the external archive **A**. Therefore, when using Eq. (11) to update the velocity, it will quickly guide the corresponding particle to approach the neighboring region around the optimal aggregated value, which enhances the ability of exploitation and resultantly accelerates the convergence speed. Otherwise, the velocity is renewed by Eq. (12); it will lead the targeted particle to search the intermediate region between x_{gbest} and itself. This is beneficial for the enhancement of exploration and simultaneously improves the swarm diversity. In MMOPSO, the advantages of

Algorithm 1 Selection for x_{pbest}

```
for i =1 to N
    x_pbest_i = A_1;
    for j = 2 to |A|
        if g(x_pbest_i | λ_i, z*) > g(A_j | λ_i, z*)
            x_pbest_i = A_j;
        end if
    end for
end for
```

Fig. 1. The pseudo-code for the selection of personal-best particles.

Algorithm 2 Evolutionary Search Strategy

```
for i =1 to |A|
    generate a random integer j in [1,|E|];
    {C_1, C_2} =SBX(A_i, E_j);
    generate a random integer k in [1, 2];
    S_i =PM(C_k);
end for
```

Fig. 2. The pseudo-code of evolutionary search strategy.

the two search patterns are combined by using a pre-defined threshold δ, as follows:

$$\begin{cases} v_i(t+1) = wv_i(t) + c_1 r_1 (x_{pbest_i} - x_i(t)) & \text{if } r_3 < \delta \\ v_i(t+1) = wv_i(t) + c_2 r_2 (x_{gbest} - x_i(t)) & \text{else} \end{cases} \tag{13}$$

where r_3 is a uniformly distributed random number in [0, 1]. The appropriate setting of δ can keep the balance between exploitation and exploration. Indicated by the experimental studies, δ is generally set in [0.5, 0.9] to put more attention on the exploitation of the current search region.

3.2. Evolutionary search on the archive

After the PSO-based optimization, the visited non-dominated solutions with bigger crowding-distance values are preserved in external archive **A**, which are considered to be good representatives of the entire *PF*. To allow the beneficial information exchange among the archive, MMOPSO performs evolutionary search on each non-dominated solution in the archive. The embedded evolutionary search power can repair the potential vulnerability of PSO search. This is supported by the recent research studies in evolutionary algorithms that the hybridized search power can enhance the search capability and the robustness to tackle various kinds of MOPs (Chen, Lin, & Ji, 2010; Sindhya et al., 2013; Tang & Wang, 2013). In MMOPSO, the evolutionary operators, such as simulated binary crossover (SBX) and polynomial mutation (PM), are performed, as they are widely adopted in multi-objective optimization algorithms (Chen et al., 2010; Deb et al., 2002; Gong, Jiao, Du, & Bo, 2008; Lin & Chen, 2013). SBX operator allows the elitist solutions to exchange useful gene segments while PM operation injects a small turbulence to search the local region. To perform SBX and PM operators on external archive **A**, an elitist subset **E** is firstly selected from **A**, which contains a number of non-dominated solutions with bigger crowding-distance values in **A**. The size of **E** is generally smaller than **A** and set to be half of |**A**| in this paper. For each solution A_i ($i = 1, 2, \ldots, |\mathbf{A}|$), a random integer j is generated in [1,|**E**|]. Then, A_i and E_j are used as parent solutions to execute SBX operator. One of the child solutions from SBX operator is randomly selected and then further to perform PM operator. The implementations of SBX and PM operators can be found in (Chen et al., 2010; Gong et al., 2008; Lin & Chen, 2013). The pseudo-code of this evolutionary search strategy is described in Fig. 2, where SBX(A_i, E_j) means to perform SBX operator on parent solutions A_i and E_j, C_1 and C_2 are the resultant child solutions generated from SBX operator, PM(C_k) indicates the execution of PM operator on C_k. After that, a new solution set **S** is generated, which will be added into the external

Algorithm 3 Archive Update

```
1:  for i =1 to |S|
2:    for j =1 to |A|
3:      State = CheckDominance( S_i , A_j );
4       if State == 1
5:        mark A_j as a dominated solution;
6       else
7:        break;
8:      end if
9:    end for
10:   delete the marked dominated solutions from A;
11:   if State! = –1
12:     add S_i to A;
13:     if |A| > N
14        CrowdingDistanceAssignment(A);
15:       delete the most crowded one;
16:     end if
17:   end if
18: end for
```

Fig. 3. The pseudo-code of archive update.

archive by archive update operation as introduced in the following subsection.

3.3. Archive update

After the execution of PSO search or evolutionary search, the new generated non-dominated solutions are collected into the external archive. As the size of archive is finite, whereas the number of non-dominated solutions may be infinite, it is necessary to use a proper selection mechanism for archive update, which can help to guide the search direction toward the true *PF*. Here, the popular archive update mechanism used in (Nebro et al., 2009; Zhan et al., 2013) is also adopted, which is based on both of Pareto dominance and crowding distance. Assuming that the new generated solution set is **S** and the solution set in external archive is **A**, the pseudo-code of the archive update procedures can be briefly described in Fig. 3, where N is the maximum size of **A**. In Fig. 3, the function **CheckDominance**(x, y) returns the Pareto dominance relationship between solutions x and y. If the function returns 1, it means that x dominates y. Otherwise, the function returns -1 when y dominates or is equal with x. Another function **CrowdingDistanceAssignment**(**A**) will calculate the crowding distance value (Deb et al., 2002) for each solution in **A**.

3.4. The complete MMOPSO algorithm

The above subsections have described the procedures of velocity update, evolutionary search and archive update, which compose the main components of MMOPSO. Besides that, the other parts are presented in the pseudo-code of MMOPSO, as illustrated in Fig. 4, where N is the size of population and external archive, ev represents the number of function evaluations, max_ev indicates the maximum number of function evaluations, r is a uniformly distributed random number in [0, 1] and δ is a predefined threshold to control the velocity update.

In the initialization phase, N weight vectors are firstly initialized and then a swarm with N particles is randomly generated, where each particle associates with a unique weight vector. The external archive **A** is initialized to be empty. After evaluating the objectives of each particle, the archive update procedures are performed to preserve the non-dominated solutions in archive **A**. Then, MMOPSO turns into the loop of evolutionary process until the function evaluation times ev reaches the predefined maximum times max_ev.

Algorithm 4 MMOPSO

```
1:  A={};
2:  ev=0;
3:  initialize the N weight vectors used in Eqs. (6-8);
4   for i = 1 to N
5:    randomly initialize the position x_i of particle i;
6:    set the velocity v_i of particle i to 0;
7:    evaluate the objectives of x_i ;
8:  end for
9:  ev = ev + N;
10: archive update (Algorithm 3);
11: while ev < max_ev
12:   selection for x_pbest (Algorithm 1);
13    for i = 1 to N
14:     if r < δ
15:       update the velocity using Eq. (11);
16:     else
17:       randomly select a solution from A as x_gbest ;
18:       update the velocity using Eq. (12);
19:     end if
20:     update the position using Eq. (10);
21:     evaluate the objectives of new particles;
22:     update the reference point z* in Eq. (6);
23:   end for
24:   ev = ev + N;
25:   archive update (Algorithm 3);
26:   evolutionary search strategy (Algorithm 2) on A;
27:   evaluate the objectives of new solutions;
28:   update the reference point z* in Eq. (6);
29:   ev = ev + |A|;
30    archive update (Algorithm 3);
31  end while
32  report the solutions in archive A;
```

Fig 4. The pseudo-code of complete MMOPSO algorithm.

During the evolutionary phase, the PSO search is first executed. The velocity of each particle is updated by using Eqs. (11) or (12), which is determined by the threshold δ. Once the random number r is smaller than δ, Eq. (11) is used to update the velocity, where the selection for $\boldsymbol{x}_{pbest}$ as introduced in **Algorithm 1** is run to find the personal-best particle that can give the best aggregation value. Otherwise, the velocity is updated using Eq. (12), where x_{gbest} is randomly picked from external archive **A**. After the positional information for each particle is renewed, the objectives of new particles are evaluated. Then, the archive update procedure as described in **Algorithm 3** is executed to gather the new non-dominated solutions with bigger crowding-distance values. After that, the evolutionary search process is run to allow the information exchange among the archive **A**, the detailed implementation of which is illustrated in **Algorithm 2**. Evolutionary operators, such as SBX and PM, are operated accordingly. Then, the objectives of the mutant solutions are computed and the archive update process in **Algorithm 3** is activated again. The above evolutionary phase will repeat until the predefined maximum function evaluation times are achieved. At the end of algorithm, the non-dominated solutions in archive **A** are reported as the final approximated *PF*.

4. Experimental studies

In this section, several experimental studies are performed to examine the performance of MMOPSO. Firstly, the related background about the simulations is introduced, including the standard benchmark problems, performance metric and the corresponding parameter settings. Secondly, the performance of MMOPSO is compared with some MOPSO algorithms and two state-of-the-art MOEAs,

Table 1
The parameter settings for all the algorithms.

Algorithms	Parameter settings
DDMOPSO	$N = 200$, $\omega \in [0.1, 0.5]$, $c_1, c_2 \in [1.5, 2.0]$
CMPSO	$N_p = 20$, $N_a = 200$ $\omega \in 0.9 \rightarrow 0.4$, $c_1 = c_2 = c_3 = 4.0/3$
SMPSO	$N = 200$, $\omega \in [0.1, 0.5]$, $c_1, c_2 \in [1.5, 2.5]$, $p_m = 1/n$, $\eta_m = 20$
dMOPSO	$N = 200$, $\omega \in [0.1, 0.5]$, $c_1, c_2 \in [1.5, 2.0]$
OMOPSO	$N = 200$, $\omega \in [0.1, 0.5]$, $c_1, c_2 \in [1.5, 2.0]$
NSGA-II	$N = 200$, $p_c = 0.9$, $p_m = 1/n$, $\eta_c = 20$, $\eta_m = 20$
MOEA/D	$N = 200$, $CR = 1.0$, $F = 0.5$, $p_m = 1/n$, $\eta_m = 20$, $T = 20$, $\delta = 0.9$, $n_r = 2$
MMOPSO	$N = 200$, $\omega \in [0.1, 0.5]$, $c_1, c_2 \in [1.5, 2.0]$, $p_c = 0.9$, $p_m = 1/n$, $\eta_c = 20$, $\eta_m = 20$, $\delta = 0.9$

e.g., DDMOPSO (Moubayed et al., 2014), CMPSO (Zhan et al., 2013), SMPSO (Nebro et al., 2009), dMOPSO (Martinez & Coello Coello, 2011), OMOPSO (Sierra & Coello Coello, 2005), NSGA-II (Deb et al., 2002) and MOEA/D (Li & Zhang, 2009). Thirdly, in order to validate the advantages of multiple search strategies in MMOPSO, the performance of MMOPSO is further compared with the two variants of MMOPSO, *i.e.*, MMOPSO-I and MMOPSO-II. MMOPSO-I replaces the velocity update equation in Eq. (13) with the traditional one in Eq. (9), while MMOPSO-II removes the evolutionary search strategy, making it a pure PSO algorithm. At last, the time complexity analysis of MMOPSO is provided.

4.1. Standard benchmark problems

In this study, twenty four standard benchmark problems without any inequality or equality constraints are used to evaluate the performance of MMOPSO. They can be classified into three categories. The first kind is low-dimensional bi-objective problems, such as Schaffer (1985), Fonseca and Flemming (1998), and Kursawe (1990). They are shortly written as SFK test problems. The second class is high-dimensional bi-objective problems, including ZDT1∼ZDT4 and ZDT6 (Zitzler et al., 2000). The third type is scalable objective problems, covering WFG1∼WFG9 (Huband et al., 2005) and DTLZ1∼DTLZ7 (Deb et al., 2005). In our experimental studies, the WFG and DTLZ family problems are respectively scaled to two and three objectives. It is noted that for ZDT1-ZDT3, the number of decision variables is 30, while the sizes of decision variables in ZDT4, ZDT6 and all DTLZ problems are 10. The number of decision variables in all WFG problems is 12, which is consisted by 4 position parameters and 8 distance parameters. These test problems are characterized with convexity, concavity, discontinuity, non-uniformity and the trap of many local *PFs*. Therefore, they are widely applied in the experimental studies to test the comprehensive performance of multi-objective optimization algorithms (Moubayed et al., 2014; Gong et al., 2008; Lin & Chen, 2013; Zhan et al., 2013).

4.2. Performance metric

One important job of MOPs is to find a uniformly distributed subset that approximates the true *PF* as close as possible, which can be provided to the decision maker as the alternative solutions for various practical cases. Since the inverted generational distance (*IGD*) metric (Li & Zhang, 2009) can examine both of the convergence and diversity, it is adopted in our experimental studies to assess the optimization performance.

Let S be a uniformly distributed subset selected from the true *PF* and S' is the approximated set that is obtained by a multi-objective optimization algorithm. The *IGD* value of S to S', *i.e.*, $IGD(S, S')$ is defined as

$$IGD(S, S') = \frac{\sum_{i=1}^{|S|} d(S_i, S')}{|S|} \tag{14}$$

where $|S|$ returns the number of solutions in set S and $d(S_i, S')$ computes the minimum Euclidean distance from S_i to the solutions of S' in objective space. When acquiring this *IGD* value, the true *PF* has to be available in advance. Generally, a lower value of $IGD(S, S')$ is preferred as it indicates that S' is distributed more uniformly and closer to the true *PF*.

4.3. Experimental settings

In this study, in order to validate the optimization performance of MMOPSO in a convincible way, MMOPSO is compared with some MOPSO algorithms, including DDMOPSO, CMPSO, SMPSO, dMOPSO and OMOPSO. Moreover, MMOPSO is also compared with two state-of-the-art MOEAs, *i.e.*, MOEA/D and NSGA-II. It is noted that the source codes of SMPSO, dMOPSO, OMOPSO, MOEA/D and NSGA-II can be found in jMetal (Durillo & Nebro, 2011) and the source code of DDMOPSO is provided by the authors that is also implemented in jMetal. Besides that, CMPSO and MMOPSO are realized by us in the framework of jMetal. All the above algorithms have shown the promising performance when tackling various kinds of MOPs. Therefore, the comparisons of MMOPSO with them can make the results more convincible.

The parameter settings of all the algorithms are summarized in Table 1. For the compared algorithms, these parameter settings are all recommended by their authors. As most of parameters in MMOPSO also exist in the compared algorithms, they are set the same with the compared algorithms for fair comparison. For MMOPSO, DDMOPSO, dMOPSO and OMOPSO, the control parameters c_1, c_2 are randomly generated from [1.5, 2.0] and the inertial weight ω is selected from [0.1, 0.5] randomly. In SMPSO, the control parameters c_1, c_2 are randomly chosen from [1.5, 2.5] and the inertial weight ω is also randomly selected from [0.1, 0.5]. For CMPSO, the control parameters c_1, c_2, and c_3 are all set to 2.0 and the inertial weight ω is linearly decreasing from 0.9 to 0.4. N is the sizes of swarm and external archive for all the algorithms except CMPSO. As multiple populations are respectively evolved to optimize multiple objectives in CMPSO, a small population size is recommended by the authors. Thus, the swarm size N_p in CMPSO is set to 20 while the external archive N_a is also set to 200. p_c and p_m are respectively the crossover and mutation probabilities used in evolutionary operators. η_c and η_m are the distribution indexes of SBX and PM respectively. For MOEA/D, T defines the size of the neighborhood in the weight coefficients, δ controls the probability that parent solutions are chosen from T neighbors and n_r is the maximal number of parent solutions that are replaced by each child solution.

It is noted that the setting of N listed in Table 1 is only for the bi-objective problems. For the triple-objective test problems, the sizes of population and external archive are all set to 595 except for CMPSO. The swarm size N_p and the external archive N_a in CMPSO are respectively set to 60 and 595 for triple-objective test problems. The expected maximal generation is 300. Therefore, the maximal numbers of function evaluations (*FEs*) are 60,000 and 178,500 for bi-objective and triple-objective problems, respectively. All the algorithms are run by 30 times in jMetal using a personal computer with a 3.20 Giga Hertz CPU, 2 Giga Byte memory and windows 7 operating system. Their mean values and standard deviations (std) for each test problem are collected for comparison, where the best results are

Table 2
IGD results on the FKS and ZDT problems.

Problems		Algorithms							
		NSGA-II	MOEA/D	OMOPSO	dMOPSO	SMPSO	CMPSO	DDMOPSO	MMOPSO
Fonseca	Mean	2.78E-03	**1.77E-03**	1.85E-03	1.82E-03	1.85E-03	2.43E-03	1.88E-03	1.86E-03
	Std	6.30E-05	**3.65E-06**	1.87E-05	1.30E-05	1.50E-05	7.17E-05	5.42E-05	1.37E-05
	p-value	9.78E-36	1.84E-25	0.0054	1.76E-16	0.0015	1.22E-27	0.0747	–
Kursawe	Mean	2.16E-02	2.07E-02	1.97E-02	2.49E-02	1.82E-02	1.95E-02	1.95E-02	**1.63E-02**
	Std	8.40E-04	1.73E-04	6.65E-04	6.70E-04	4.98E-04	5.83E-04	1.29E-03	**3.00E-04**
	p-value	3.03E-25	2.31E-33	4.49E-21	8.75E-33	4.43E-18	4.07E-21	1.97E-14	–
Schaffer	Mean	4.46E-01	4.84E-01	**7.97E-03**	1.67E-01	8.33E-03	5.15E-02	8.33E-03	8.00E-03
	Std	1.46E-01	2.03E-01	**7.89E-04**	3.09E-05	5.55E-04	9.06E-03	5.89E-04	6.29E-04
	p-value	3.15E-16	1.77E-13	0.8756	1.63E-71	0.0353	9.19E-22	0.0601	–
ZDT1	Mean	2.33E-03	5.33E-03	1.86E-03	2.25E-03	**1.82E-03**	2.03E-03	3.42E-03	1.87E-03
	Std	6.14E-05	1.16E-03	2.11E-05	9.58E-06	**1.53E-05**	3.94E-05	7.26E-04	1.38E-05
	p-value	1.30E-26	3.69E-16	0.3050	5.89E-41	4.13E-13	1.42E-18	1.65E-12	–
ZDT2	Mean	2.39E-03	3.98E-03	1.92E-03	1.96E-03	**1.89E-03**	2.14E-03	2.26E-01	1.91E-03
	Std	7.71E-05	8.76E-04	1.76E-05	1.75E-05	**1.47E-05**	3.67E-05	2.98E-01	2.10E-05
	p-value	1.26E-24	1.42E-13	0.0283	3.10E-10	2.81E-4	4.91E-24	2.89E-4	–
ZDT3	Mean	2.60E-03	6.16E-03	2.24E-03	6.64E-03	2.14E-03	4.64E-03	6.06E-03	**2.10E-03**
	Std	8.07E-05	6.64E-04	8.61E-05	9.90E-05	7.86E-05	8.46E-04	1.78E-03	**4.49E-05**
	p-value	1.04E-22	1.40E-24	2.06E-8	3.76E-48	0.0729	2.22E-16	6.14E-13	–
ZDT4	Mean	2.48E-03	4.94E-02	4.44E+00	2.27E-03	1.87E-03	5.06E-02	1.66E+00	**1.84E-03**
	Std	2.57E-04	6.02E-02	2.02E+00	1.41E-05	2.18E-05	3.94E-02	1.33E+00	**1.87E-05**
	p-value	4.53E-14	1.63E-4	8.28E-13	9.38E-41	1.89E-5	1.94E-7	1.67E-7	–
ZDT6	Mean	2.57E-03	**1.17E-03**	1.56E-03	1.18E-03	1.46E-03	1.80E-03	3.75E-03	1.56E-03
	Std	1.93E-04	**3.02E-06**	8.72E-05	4.64E-07	7.92E-05	1.84E-04	2.16E-03	4.72E-05
	p-value	2.09E-23	1.87E-28	0.8586	3.79E-28	4.54E-6	4.60E-8	5.44E-6	–
Better/Similar/Worse		8/0/0	6/0/2	4/3/1	6/0/2	3/1/4	8/0/0	6/2/0	–

identified with bold font in comparison tables. Moreover, the *t-test* with significant level $\alpha = 0.05$ is also performed to examine that whether the IGD mean values obtained by MMOPSO are statistically different from that obtained by the other algorithms. The *p-value* returned by the *t-test* is also collected in the comparison table, where the *p-value* bigger than 0.05 means that the compared IGD mean values are statistically similar. It is noted that the underlined IGD results for the compared algorithms indicate that they are statistically similar with that obtained by MMOPSO under the *t-test*.

4.4. Comparisons of MMOPSO with other multi-objective algorithms

4.4.1. Comparisons on the FKS and ZDT test problems

Table 2 summarizes the results of all the algorithms on the FKS and ZDT test problems. Our MMOPSO algorithm obtains the best results on Kursawe, ZDT3 and ZDT4, while SMPSO performs best on ZDT1 and ZDT2. Moreover, MOEA/D gets the best results on Fonseca and ZDT6, and OMOPSO performs best on Schaffer. Since the corresponding test problems are not so difficult, it is observed that the compared algorithms perform well on most of test problems. However, it is important to point out that some of the compared algorithms are lack of capabilities in handling test problems with specific characteristics. For example, NSGA-II, MOEA/D and dMOPSO can't effectively approach the true *PF* of Schaffer; DDMOPSO gives the worst result on ZDT2; OMOPSO and DDMOPSO are unable to find the true *PF* of ZDT4 due to the existence of many local *PFs*.

The *t-test* results indicate that MMOPSO performs similarly with OMOPSO on Schaffer, ZDT1 and ZDT6, with SMPSO on ZDT3, and with DDMOPSO on Fonseca and Schaffer. Moreover, the final comparison results of MMOPSO with the compared algorithms are clearly concluded in the last row of Table 2, where Better/Similar/Worse indicates that the number of test problems that the results obtained by MMOPSO are better than, similar with or worse than that of the compared algorithms. It is quite obvious that MMOPSO performs better than or similarly with NSGA-II, CMPSO and DDMOPSO on all the test problems. For MOEA/D and dMOPSO, MMOPSO obtains the better results on 6 out of 8 test problems. Besides that, MMOPSO performs better than OMOPSO, and worse than SMPSO. Actually, both of MMOPSO and SMPSO are able to solve the corresponding test problems well. To visually show the optimization performance, the best results of MMOPSO on these test problems are plotted in Fig. 5, where the true *PFs* are identified with the red lines and the approximated *PFs* are marked with black diamonds. It is evident that the found approximated *PF* is distributed uniformly on the true *PF*.

4.4.2. Comparisons on the WFG test problems

Table 3 presents the simulation results obtained by all the algorithms on the WFG test problems. Our proposed MMOPSO algorithm achieves the best performance on WFG1, WFG3, WFG4, WFG7 and WFG9, while OMOPSO performs best on WFG2 and WFG6. MOEA/D and CMPSO get the best results on WFG8 and WFG5, respectively. It is noted that most of the compared algorithms fail to approach the true *PF* of WFG1, whereas MMOPSO performs better. Actually, observed from Table 3, MMOPSO is able to deal with most of WFG problems quite well.

The *t-test* results show that MMOPSO gets the statistically similar results with MOEA/D on WFG2, with OMOPSO on WFG5, with SMPSO on WFG5 and WFG6, with CMPSO on WFG8, and with DDMOPSO on WFG2. The comparison summary in the last row of Table 3 illustrates that MMOPSO performs better than the compared targets on most of WFG test problems. Fig. 6 plots the best results obtained by MMOPSO on the WFG test problems, which can visually show the promising performance of MMOPSO. Except for WFG8, MMOPSO is able to effectively approach the true *PFs*.

4.4.3. Comparisons on the DTLZ test problems

Table 4 gives all the experimental results on the DTLZ test problems. Our proposed MMOPSO algorithm performs best on DTLZ1, DTLZ5 and DTLZ6, while dMOPSO obtains the best results on DTLZ2 and DTLZ4. MOEA/D and CMPSO get the best performance on DTLZ3 and DTLZ7, respectively. The simulations show that some MOPSO algorithms, such as OMOPSO, CMPSO and DDMOPSO, can't effectively

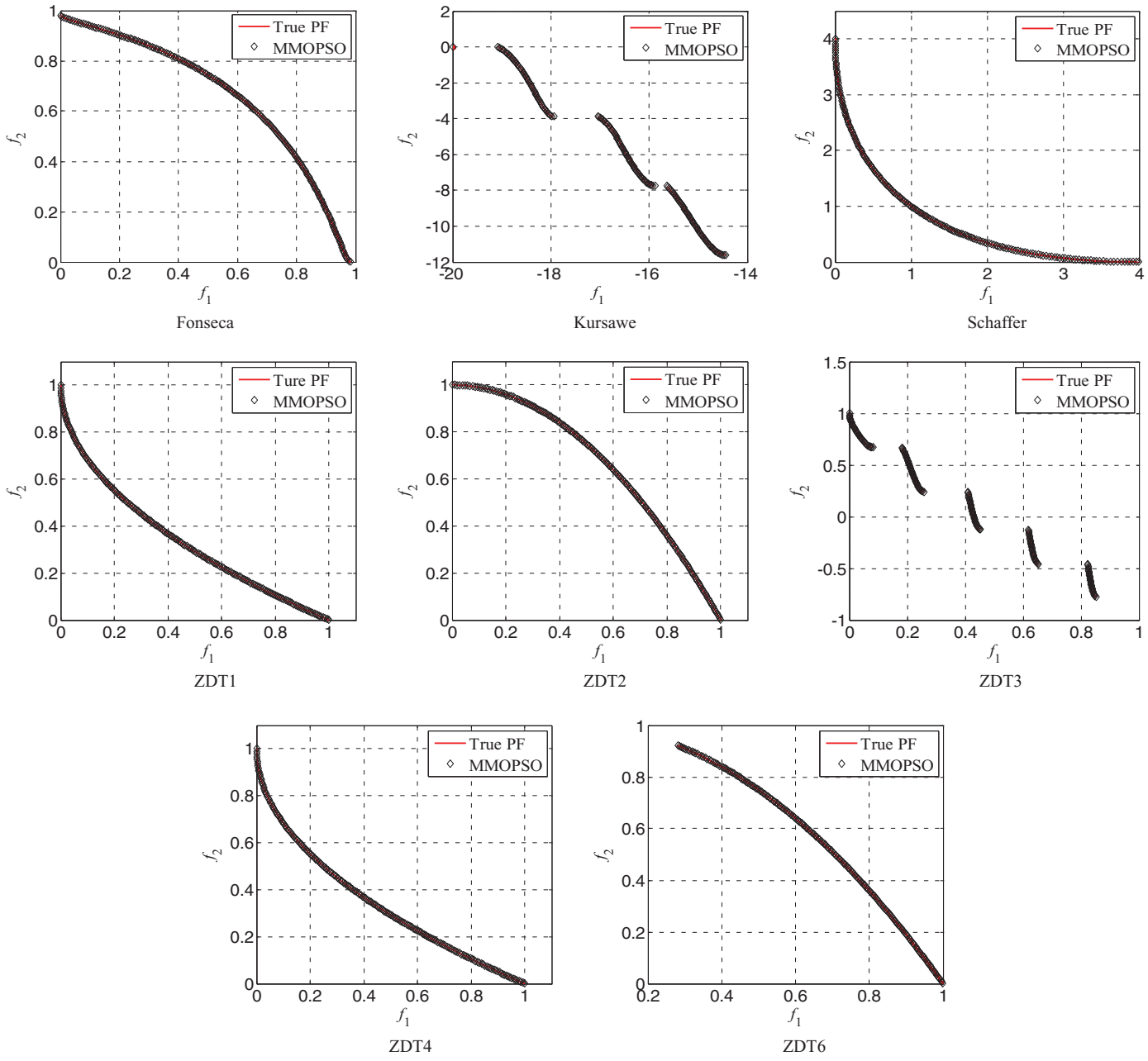

Fig. 5. The plots of best performance obtained by MMOPSO on the FKS and ZDT test problems.

approach the true *PF* of DTLZ3 as it is easy to get trapped in local *PFs*, while MMOPSO and SMPSO can handle it quite well.

The *t-test* results show that MMOPSO, NSGA-II, OMOPSO, SMPSO and CMPSO obtain the similar results on DTLZ4. Moreover, MMOPSO performs similarly with SMPSO and OMOPSO on DTLZ6. The comparison conclusion on the last row of Table 4 indicates that MMOPSO performs better than NSGA-II, OMOPSO, SMPSO and DDMOPSO. When compared with MOEA/D, dMOPSO and CMPSO, our proposed algorithm MMOPSO still have some advantages on these DTLZ problems. Fig. 7 gives the plots of best results obtained by MMOPSO on the DTLZ problems, where the true *PFs* are marked with red surface and the approximated *PFs* are identified with black diamonds. These plots further confirm that MMOPSO can find the approximated *PF* that is distributed uniformly and very close to the true *PF*.

At last, all the comparison summaries on the last rows of Tables 2–4 are collected in Table 5, which gives the comprehensive performance of MMOPSO when compared with the other algorithms on all the test problems. It is obvious that MMOPSO performs better than or similarly with NSGA-II, OMOPSO, dMOPSO, CMPSO and DDMOPSO on at least 20 out of 24 test problems. When compared with MOEA/D and SMPSO, MMOPSO also obtains the better or similar results on 17 and 18 out of 24 test problems, respectively. These experimental results justify the advantages of MMOPSO when handling various kinds of test problems.

4.5. Advantages of multiple search strategies in MMOPSO

In order to investigate the advantages of multiple search strategies in MMOPSO, two variants of MMOPSO are included for comparison, *i.e.*, MMOPSO-I and MMOPSO-II. They have the similar components with MMOPSO, except that MMOPSO-I replaces the velocity update equation in Eq. (13) with the traditional one in Eq. (9) and MMOPSO-II removes the evolutionary search component on the archive. In

Table 3
IGD results on the WFG test problems.

Problems		Algorithms							
		NSGA-II	MOEA/D	OMOPSO	dMOPSO	SMPSO	CMPSO	DDMOPSO	MMOPSO
WFG1	Mean	6.47E-01	7.51E-01	1.15E+00	1.20E+00	1.19E+00	1.04E+00	5.11E-01	**1.22E-02**
	Std	3.22E-01	1.38E-01	2.57E-02	5.57E-03	1.38E-02	8.89E-02	1.07E-01	**4.06E-03**
	p-value	1.18E-11	3.28E-23	1.27E-49	1.19E-66	1.53E-57	1.20E-32	1.73E-21	–
WFG2	Mean	6.24E-02	2.72E-02	**6.47E-03**	2.00E-01	1.09E-02	1.72E-02	3.03E-02	3.31E-02
	Std	4.47E-04	1.62E-02	**5.48E-04**	1.96E-02	1.55E-03	1.08E-02	4.53E-02	2.87E-02
	p-value	4.96E-6	0.3095	1.96E-5	9.79E-21	2.22E-4	0.0100	0.7868	–
WFG3	Mean	8.26E-03	7.13E-03	6.34E-03	2.20E-02	8.06E-03	1.57E-02	6.20E-03	**5.47E-03**
	Std	4.32E-04	1.02E-04	6.15E-04	1.58E-03	6.62E-04	3.21E-03	5.01E-04	**7.36E-04**
	p-value	1.04E-17	4.28E-13	2.75E-5	5.41E-31	4.75E-14	6.65E-17	5.11E-5	–
WFG4	Mean	7.15E-03	2.40E-02	4.32E-02	5.34E-02	4.78E-02	7.71E-03	1.45E-02	**5.74E-03**
	Std	2.75E-04	4.93E-03	2.20E-03	2.14E-03	6.06E-03	6.03E-04	3.17E-03	**4.53E-04**
	p-value	7.72E-14	1.39E-18	1.51E-37	7.32E-41	4.64E-26	4.67E-14	0.0198	–
WFG5	Mean	6.59E-02	6.55E-02	6.53E-02	6.61E-02	6.52E-02	**6.26E-02**	6.56E-02	6.47E-02
	Std	1.76E-04	6.11E-05	1.23E-04	2.61E-04	9.91E-05	**2.88E-03**	1.03E-04	2.01E-03
	p-value	0.0034	0.0392	0.1142	5.48E-4	0.1581	0.0044	0.0198	–
WFG6	Mean	7.92E-02	8.39E-03	**7.15E-03**	2.65E-02	1.72E-02	3.05E-02	2.19E-02	1.42E-02
	Std	2.89E-02	1.51E-03	**4.45E-04**	7.18E-03	1.98E-02	1.57E-02	1.30E-02	1.10E-02
	p-value	1.41E-11	0.0066	0.0015	4.48E-7	0.5087	1.23E-4	0.0256	–
WFG7	Mean	8.02E-03	8.81E-03	6.26E-03	2.29E-02	7.16E-03	9.40E-03	6.19E-03	**5.93E-03**
	Std	3.55E-04	8.44E-05	6.40E-05	1.75E-03	3.34E-04	5.81E-04	8.95E-05	**7.35E-05**
	p-value	1.16E-24	4.62E-43	3.75E-18	1.92E-30	1.63E-18	2.59E-24	4.28E-13	–
WFG8	Mean	1.95E-01	**1.68E-01**	2.33E-01	2.49E-01	2.00E-01	2.23E-01	2.32E-01	2.24E-01
	Std	8.59E-03	**4.97E-02**	6.46E-03	6.07E-03	2.02E-02	2.53E-02	1.74E-02	3.47E-03
	p-value	8.76E-17	1.19E-6	2.32E-7	1.19E-18	7.41E-7	0.7934	0.0134	–
WFG9	Mean	1.18E-02	3.52E-02	1.56E-02	2.42E-02	1.74E-02	1.51E-02	1.21E-02	**8.86E-03**
	Std	1.56E-03	6.84E-02	6.17E-04	1.65E-03	2.90E-04	3.52E-03	1.51E-03	**1.52E-03**
	p-value	5.00E-8	0.0421	4.44E-19	2.95E-25	2.22E-23	3.84E-9	2.63E-10	–
Better/Similar/Worse		8/0/1	6/1/2	6/1/2	9/0/0	5/2/2	6/1/2	8/1/0	–

Table 4
IGD results on the DTLZ test problems.

Problems		Algorithms							
		NSGA-II	MOEA/D	OMOPSO	dMOPSO	SMPSO	CMPSO	DDMOPSO	MMOPSO
DTLZ1	Mean	1.07E-02	1.11E-02	2.47E+01	1.78E-02	1.16E-02	5.83E-02	3.22E+00	**1.01E-02**
	Std	3.57E-04	2.50E-04	8.01E+00	3.89E-03	2.26E-04	1.99E-02	2.95E+00	**2.02E-04**
	p-value	7.16E-8	2.65E-18	1.54E-16	8.41E-12	2.34E-22	7.55E-14	1.80E-6	–
DTLZ2	Mean	2.81E-02	2.61E-02	2.81E-02	**2.24E-02**	2.84E-02	2.56E-02	2.63E-02	2.74E-02
	Std	5.64E-04	1.12E-04	4.88E-04	**2.00E-04**	7.22E-04	4.51E-04	3.91E-04	4.27E-04
	p-value	1.91E-6	2.32E-16	1.03E-7	1.26E-31	2.65E-7	1.28E-15	1.54E-12	–
DTLZ3	Mean	2.85E-02	**2.68E-02**	8.65E+01	4.32E-02	2.83E-02	1.18E-01	1.26E+01	2.75E-02
	Std	9.97E-04	**3.27E-04**	2.15E+01	5.59E-03	4.97E-04	3.28E-02	1.19E+01	5.08E-04
	p-value	1.30E-5	2.25E-5	1.14E-19	1.12E-15	8.23E-7	2.61E-15	3.04E-6	–
DTLZ4	Mean	2.87E-02	1.98E-02	2.78E-02	**1.73E-02**	2.91E-02	6.78E-02	3.01E-02	2.85E-02
	Std	2.05E-03	8.61E-04	4.93E-03	**8.02E-04**	4.32E-03	1.24E-01	2.56E-03	1.81E-03
	p-value	0.7344	2.77E-20	0.5013	2.60E-23	0.5213	0.0929	0.0036	–
DTLZ5	Mean	8.81E-04	2.64E-03	6.81E-04	7.31E-03	6.76E-04	9.66E-04	1.07E-03	**6.61E-04**
	Std	3.87E-05	9.92E-06	2.18E-05	2.45E-04	2.39E-05	3.72E-05	1.53E-04	**2.27E-05**
	p-value	3.01E-23	5.09E-58	7.28E-4	3.11E-43	0.0098	5.14E-26	3.84E-14	–
DTLZ6	Mean	9.35E-04	2.37E-03	6.62E-04	1.25E-02	6.52E-04	7.62E-04	4.08E-03	**6.44E-04**
	Std	5.84E-05	4.26E-06	3.47E-05	1.11E-05	3.46E-05	4.02E-05	8.75E-04	**3.27E-05**
	p-value	6.43E-22	1.41E-51	0.0537	6.47E-75	0.3950	2.24E-14	2.96E-19	–
DTLZ7	Mean	2.77E-02	3.64E-02	2.93E-02	5.21E-02	3.02E-02	**2.72E-02**	3.64E-02	2.86E-02
	Std	6.16E-04	2.08E-03	8.02E-04	1.65E-04	1.21E-03	**7.43E-04**	2.08E-03	9.63E-04
	p-value	1.66E-4	4.39E-18	0.0128	6.00E-42	3.45E-6	3.77E-8	4.39E-18	–
Better/Similar/Worse		5/1/1	4/0/3	5/2/0	5/0/2	5/2/0	4/1/2	6/0/1	–

Table 6, the comparison results of MMOPSO, MMOPSO-I and MMOPSO-II on all the test problems are listed.

Observed from Table 6, MMOPSO performs best on 18 out of 24 test problems, which validates the advantages of MMOPSO when compared with MMOPSO-I and MMOPSO-II. The *t-test* results also reveal that MMOPSO obtains better results than MMOPSO-I and MMOPSO-II on 12 and 10 test problems, respectively. Moreover, it has the similar performance with MMOPSO-I on 11 test problems and with MMOPSO-II on 13 test problems. In other words, MMOPSO performs better than or similarly with MMOPSO-I and MMOPSO-II on

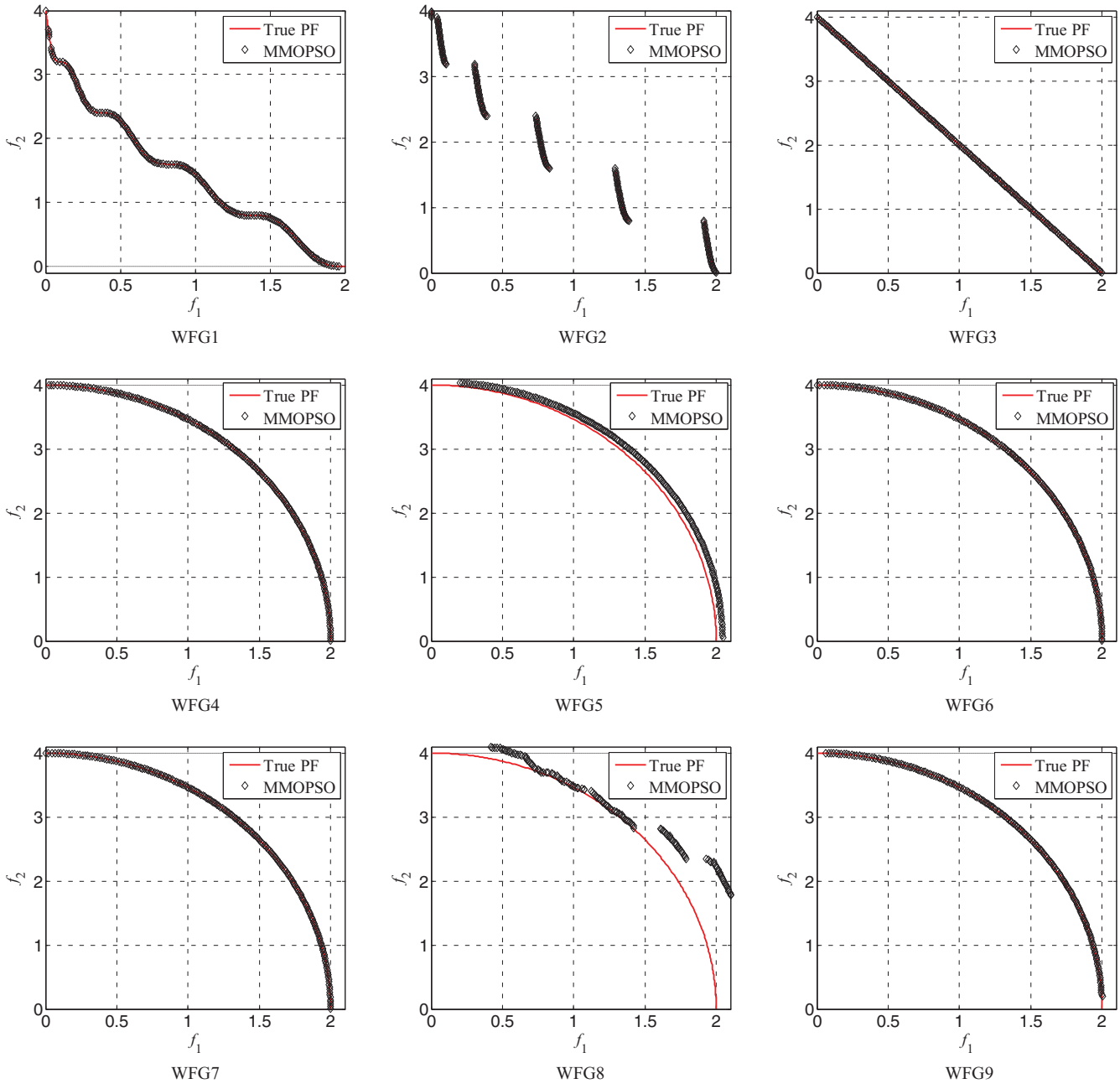

Fig. 6. The plots of best performance obtained by MMOPSO on the WFG test problems.

Table 5
Comparison summary on all the test problems.

Problems	Algorithms						
	NSGA-II	MOEA/D	OMOPSO	dMOPSO	SMPSO	CMPSO	DDMOPSO
FKS and ZDT	8/0/0	6/0/2	4/3/1	6/0/2	3/1/4	8/0/0	6/2/0
WFG	8/0/1	6/1/2	6/1/2	9/0/0	5/2/2	6/1/2	8/1/0
DTLZ	5/1/1	4/0/3	5/2/0	5/0/2	5/2/0	4/1/2	6/0/1
Total (Better/Similar/Worse)	21/1/2	16/1/7	15/6/3	20/0/4	13/5/6	18/2/4	20/3/1

23 out of 24 test problems. This justifies the effectiveness of multiple search strategies in MMOPSO.

To investigate the effect of evolutionary search on the external archive, it can be observed from the performance of MMOPSO-II that it is unable to approach the true *PFs* of some test problems, such as ZDT4, WFG1, DTZL1 and DTZL3. These test problems are mostly characterized with the trap of many local *PFs*. Thus, it indicates that the PSO search pattern in MMOPSO is easy to get trapped in local *PFs*, while the embedding of evolutionary search power on archive can remedy this shortcoming, which provides the capability to jump out of the local *PFs*. On the other hand, although MMOPSO-I performs much better than MMOPSO-II on ZDT4, WFG1, DTZL1 and DTLZ3, it

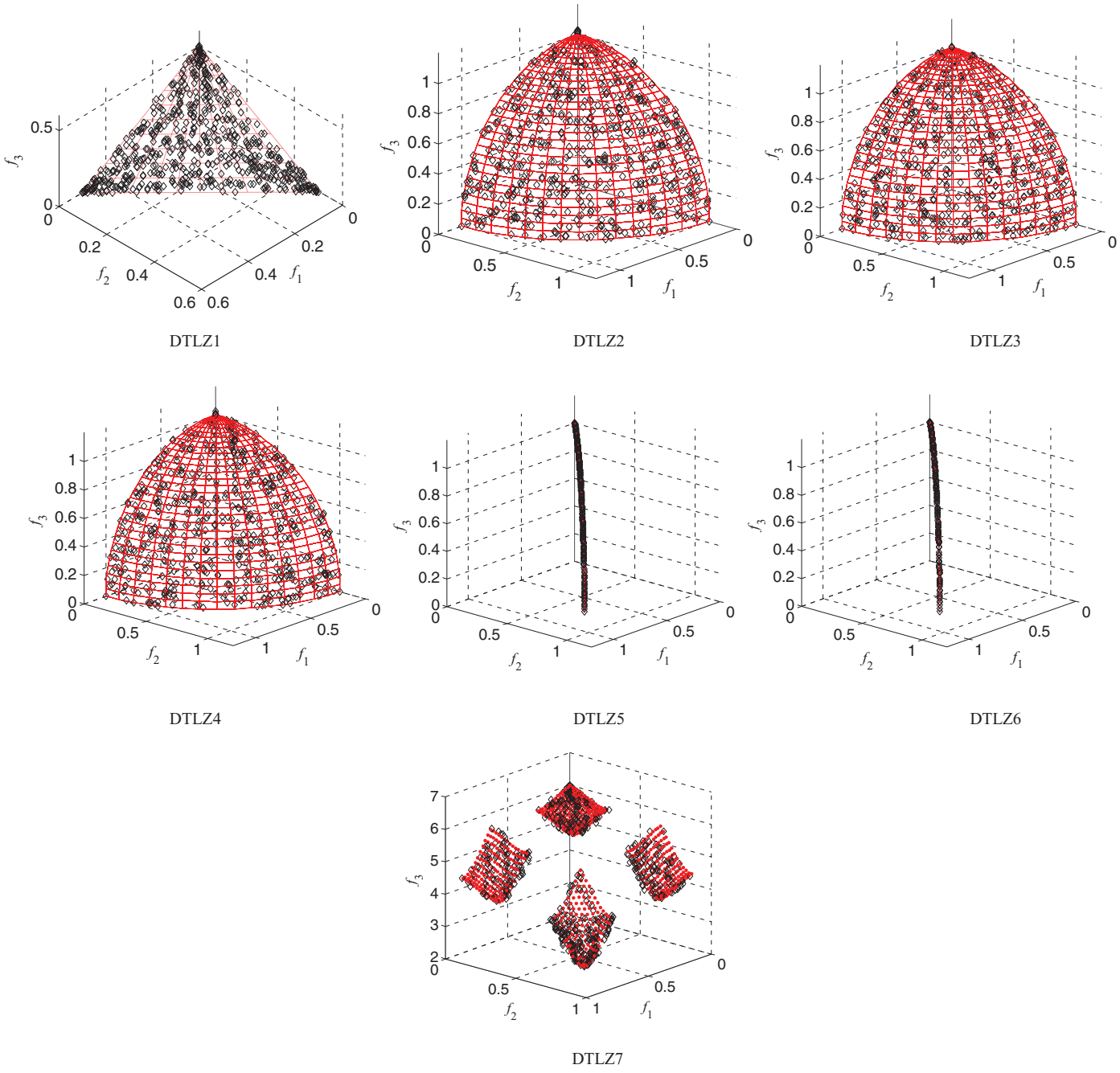

Fig. 7. The plots of best performance obtained by MMOPSO on the DTLZ test problems.

still cannot gain the satisfactory results as obtained by MMOPSO. In most cases, MMOPSO-I performs worse than MMOPSO. This justifies that the two search strategies for velocity update are beneficial for enhancing the PSO search capability to handle different MOPs.

4.6. Time complexity analysis

In this subsection, the time complexity analysis of MMOPSO is provided and compared with the other algorithms. Based on the pseudo-code of MMOPSO in Fig. 4, the time complexity of MMOPSO is determined by the evolutionary loop in lines 12–30. It is noted that when computing the following time complexity, the impact of decision variables and objectives are generally ignored as they are much smaller than the sizes of population and external archive N. In line 12, the time complexity of **Algorithm 1** in Fig. 1 is $O(N^2)$; in lines 13–23, two search strategies are executed to update the velocity of each particle and the corresponding time complexity is $O(N)$. **Algorithm 3** and **Algorithm 2** are serially operated in lines 25–26, and their time complexity are respectively $O(N^2)$ and $O(N)$ as observed in Figs. 2 and 3. In lines 27–28, the time complexity is obviously $O(N)$ as it only evaluates the objectives of N new solutions and then update the reference point z^*. At last, **Algorithm 3** is activated again with the time complexity $O(N^2)$. Thus, based on the above analysis, the time complexity of MMOPSO is $O(3N^2+3N)\sim O(N^2)$. As discussed in Moubayed et al. (2014), the computational complexities of MOEA/D, dMOPSO, OMOPSO and DDMOPSO are all $O(N^2)$ when the size of external archive is set the same with the population size N. Besides that, SMPSO shares the similar structure with OMOPSO and thus its computational complexity is also $O(N^2)$, while the time complexity of NSGA-II is $O(N^2)$ (Deb et al., 2002). For CMPSO, its time complexity is dependent on the size of sub-swarm. By simply assuming the sub-swarm size is also N, the time complexity of CMPSO is also

Table 6
IGD results of MMOPSO, MMOPSO-I and MMOPSO-II on all the test problems.

Problems		Algorithms			Problems		Algorithms		
		MMOPSO -I	MMOPSO -II	MMOPSO			MMOPSO-I	MMOPSO -II	MMOPSO
Fonseca	Mean	**1.86E-03**	**1.86E-03**	**1.86E-03**	WFG5	Mean	**6.43E-02**	6.54E-02	6.47E-02
	Std	**1.10E-05**	**1.77E-05**	**1.37E-05**		Std	**2.47E-03**	3.93E-04	2.01E-03
	p-value	0.9941	0.7686	–		*p-value*	0.5248	0.0867	–
Kursawe	Mean	1.64E-02	1.64E-02	**1.63E-02**	WFG6	Mean	3.38E-02	**9.59E-03**	1.42E-02
	Std	2.82E-04	3.52E-04	**3.00E-04**		Std	2.33E-02	**7.50E-03**	1.10E-02
	p-value	0.3543	0.4621	–		*p-value*	6.66E-4	0.0748	–
Schaffer	Mean	8.52E-03	8.07E-03	**8.00E-03**	WFG7	Mean	5.99E-03	**5.92E-03**	5.93E-03
	Std	5.19E-04	8.48E-04	**6.29E-04**		Std	8.14E-05	**7.51E-05**	7.35E-05
	p-value	2.65E-4	0.7467	–		*p-value*	0.0054	0.9570	–
ZDT1	Mean	1.89E-03	2.32E-03	**1.87E-03**	WFG8	Mean	2.26E-01	2.25E-01	**2.24E-01**
	Std	1.63E-05	8.35E-05	**1.38E-05**		Std	7.20E-03	1.18E-02	**3.47E-03**
	p-value	5.75E-5	1.76E-23	–		*p-value*	0.2218	0.5533	–
ZDT2	Mean	1.94E-03	5.47E-01	**1.91E-03**	WFG9	Mean	1.10E-02	1.84E-02	**8.86E-03**
	Std	3.43E-05	2.20E-01	**2.10E-05**		Std	1.76E-03	5.07E-02	**1.52E-03**
	p-value	0.0011	4.15E-14	–		*p-value*	1.31E-5	0.3138	–
ZDT3	Mean	**2.10E-03**	2.23E-03	**2.10E-03**	DTLZ1	Mean	3.32E-02	1.45E+01	**1.01E-02**
	Std	**6.05E-05**	5.89E-05	**4.49E-05**		Std	4.74E-02	5.70E+00	**2.02E-04**
	p-value	0.9769	7.67E-10	–		*p-value*	0.0124	2.47E-14	–
ZDT4	Mean	2.65E-02	1.24E+01	**1.84E-03**	DTLZ2	Mean	2.74E-02	**2.71E-02**	2.74E-02
	Std	7.79E-02	6.07E+00	**1.87E-05**		Std	5.33E-04	**4.63E-04**	4.27E-04
	p-value	0.0934	4.85E-12	–		*p-value*	0.8975	0.0057	–
ZDT6	Mean	1.64E-03	3.50E-02	**1.56E-03**	DTLZ3	Mean	1.44E-01	4.86E+01	**2.75E-02**
	Std	1.37E-04	1.79E-01	**4.72E-05**		Std	2.84E-01	1.63E+01	**5.08E-04**
	p-value	0.0037	0.3142	–		*p-value*	0.0320	3.44E-16	–
WFG1	Mean	1.03E-01	1.89E-01	**1.22E-02**	DTLZ4	Mean	2.85E-02	**2.80E-02**	2.85E-02
	Std	5.89E-02	1.33E-01	**4.06E-03**		Std	1.47E-03	**1.83E-03**	1.81E-03
	p-value	1.73E-9	4.65E-8	–		*p-value*	0.9939	0.3951	–
WFG2	Mean	**1.65E-02**	3.82E-02	3.31E-02	DTLZ5	Mean	6.67E-04	6.66E-04	**6.61E-04**
	Std	**2.29E-02**	4.35E-02	2.87E-02		Std	2.20E-05	1.71E-05	**2.27E-05**
	p-value	0.0189	0.6098	–		*p-value*	0.1477	0.3663	–
WFG3	Mean	5.76E-03	5.61E-03	**5.47E-03**	DTLZ6	Mean	6.54E-04	1.98E-03	**6.44E-04**
	Std	6.18E-04	5.81E-04	**7.36E-04**		Std	4.03E-05	4.03E-04	**3.27E-05**
	p-value	0.1725	0.3653	–		*p-value*	0.2725	5.25E-17	–
WFG4	Mean	6.09E-03	1.48E-02	**5.74E-03**	DTLZ7	Mean	2.94E-02	2.95E-02	**2.86E-02**
	Std	4.14E-04	4.62E-03	**4.53E-04**		Std	8.11E-04	1.40E-03	**9.63E-04**
	p-value	0.0053	1.13E-11	–		*p-value*	0.0027	0.0178	–
Better/Similar/Worse		12/11/1	10/13/1	–					

Table 7
The average computational time (in seconds) for the WFG test problems.

Problems	Algorithms				
	dMOPSO	DDMOPSO	MMOPSO-I	MMOPSO-II	MMOPSO
WFG1	2.29	4.08	**0.85**	1.10	1.15
WFG2	2.13	45.54	**1.39**	2.30	1.71
WFG3	**2.13**	91.58	2.50	3.51	2.82
WFG4	2.19	21.09	**1.38**	1.85	1.81
WFG5	**2.14**	63.92	2.41	3.38	2.79
WFG6	2.15	43.89	**1.73**	3.00	2.39
WFG7	**2.18**	96.24	2.62	3.58	2.95
WFG8	2.24	45.75	1.34	1.49	**1.12**
WFG9	2.37	28.57	**1.57**	2.36	2.26
Average	2.20	48.96	1.76	2.51	2.11

$O(N^2)$. Therefore, based on the theoretical analysis, MMOPSO has the similar time complexity with the compared algorithms.

To further study the extra computational burden induced by the designed multiple search strategies, Table 7 gives the average computational time of MMOPSO-I, MMOPSO-II and MMOPSO in solving all the WFG test problems. Besides that, two decomposition-based MOPSOs, *i.e.*, dMOPSO and DDMOPSO, are also included for comparison. It is noted that the average time for each WFG problem is obtained by 30 independent runs and the lowest one is highlighted with boldface. As observed from Table 7, MMOPSO needs 2.11 seconds in average to solve one WFG test problem, which performs slower than MMOPSO-I and faster than MMOPSO-II. This indicates that the proposed multiple search strategies won't bring much computational burden. Even compared with MMOPSO-I, it is still worthy of spending 17 percent extra average time in order to obtain the superior performance as provided by MMOPSO in Table 6. Moreover, MMOPSO performs slightly faster than dMOPSO and much faster than DDMOPSO, which justify the computational efficiency of MMOPSO when compared with other decomposition-based MOPSOs. However, the execution time of dMOPSO is more stable than that of MMOPSO and DDMOPSO. This is because dMOPSO only adopts the aggregated values to update the globally best particles, while MMOPSO and DDMOPSO utilize the crowding-distance metric to store the non-dominated solutions. Thus, the execution times of MMOPSO and DDMOPSO are greatly affected by the number of non-dominated solutions found during the search phase. Especially for DDMOPSO, it is worth noting that its execution time is much longer than that of dMOPSO and MMOPSO. This is mainly due to the fact that the new archiving approach in DDMOPSO needs to compute the crowding-distance values in both objective and decision spaces, and its source code has not been fully optimized by the authors.

5. Conclusions

In this paper, a novel MOPSO algorithm with multiple search strategies is presented, which is based on decomposition approach

to transform MOPs into a set of aggregation problems. Each particle in the swarm is accordingly assigned to optimize each aggregation problem. A novel velocity update approach is designed to renew the particle velocity by using two search strategies, which can concurrently promote the convergence speed and remain the population diversity. Additionally, evolutionary search strategy is further performed on the non-dominated solutions in the external archive, which is able to exchange their beneficial information. This embedded evolutionary search power can repair the weakness of PSO search pattern and resultantly enhance the comprehensive performance of MMOPSO in solving various kinds of MOPs. The effectiveness and efficiency of multiple search strategies are also justified by the experimental studies. When compared with some MOPSO algorithms and two state-of-the-art MOEAs, such as DDMOPSO, CMPSO, SMPSO, dMOPSO, OMOPSO, MOEA/D and NSGA-II, the experimental results illustrate that MMOPSO performs best on most of test problems.

Our future study will further enhance the performance of MMOPSO, and extend it for tackling MOPs with more than three objectives. Furthermore, the applications of MMOPSO for some practical engineering problems will also be investigated in our future work.

Acknowledgments

This work was supported by the National Nature Science Foundation of China under Grant nos. 61402291 and 61170283, National High-Technology Research and Development Program ("863" Program) of China under Grant 2013AA01A212, Ministry of Education in the New Century Excellent Talents Support Program under Grant NCET-12-0649, Foundation for Distinguished Young Talents in Higher Education of Guangdong under Grant 2014KQNCX129, Shenzhen Technology Plan under Grant JCYJ20140418095735608, and Natural Science Foundation of SZU under Grant 201531. The authors are grateful to both the editor and anonymous reviewers for their constructive comments, which greatly improve the quality of this paper.

References

Moubayed, N. AI, Pertovski, A., & McCall, J. (2010). A novel smart multi-objective particle swarm optimisation using decomposition. *Parallel Problem Solving from Nature-PPSN XI, PT II, Lecture Notes in Computer Science, 6239*, 1–10.

Moubayed, N. AI, Pertovski, A., & McCall, J. (2012). d^2mopso: Multi-objective particle swarm optimizer based on decomposition and dominance. *Evolutionary Computation in Combinatorial Optimization, Notes in Computer Science, 7245*, 75–86.

Moubayed, N. AI, Pertovski, A., & McCall, J. (2014). D^2MOPSO: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces. *Evolutionary Computation, 22*, 47–77.

Bosman, P. A. N., & Thierens, D. (2003). The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation, 7*, 174–188.

Chen, J. Y., Lin, Q. Z., & Ji, Z. (2010). A hybrid immune multiobjective optimization algorithm. *European Journal of Operational Research, 204*, 294–302.

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*, 58–73.

Coello Coello, C. A., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*, 256–279.

Dang, D. C., Guibadj, R. N., & Moukrim, A. (2013). An effective PSO-inspired algorithm for the team orienteering problem. *European Journal of Operational Research, 229*, 332–344.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*, 182–197.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable test problems for evolutionary multi-objective optimization. In *Evolutionary multiobjective optimization, advanced information and knowledge processing* (pp. 105–145). London: Springer.

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software, 42*, 760–771.

Fonseca, C. M., & Flemming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms Part II: Application example. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 28*, 38–47.

Goh, C. K., Tan, K. C., Liu, D. S., & Chiam, S. C. (2010). A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research, 202*, 42–54.

Gong, M. G., Jiao, L. C., Du, H. F., & Bo, L. F. (2008). Multi-objective immune algorithm with nondominated neighbor-based selection. *Evolutionary Computation, 16*, 225–255.

Gong, M. G., Cai, Q., Chen, X. W., & Ma, L. J. (2014). Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. *IEEE Transactions on Evolutionary Computation, 18*, 82–97.

Hu, M. Q., Wu, T., & Weir, J. D. (2013). An adaptive particle swarm optimization with multiple adaptive methods. *IEEE Transactions on Evolutionary Computation, 17*, 705–720.

Huband, S., Barone, L., While, L., & Hingston, P. (2005). A scalable multi-objective test problem toolkit. *Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, 3410*, 280–295.

Ishibuchi, H., & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 28*, 392–403.

Jones, D. F., Mirrazavi, S. K., & Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research, 137*, 1–9.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks, 4*, 1942–1948.

Kursawe, F. (1990). A variant of evolution strategies for vector optimization. *Parallel Problem Solving from Nature 1st Workshop, PPSN I, Lecture Notes in Computer Science, 496*, 193–197.

Li, C. H., Yang, S. X., & Nguyen, T. T. (2012). A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 42*, 627–646.

Li, H., & Zhang, Q. F. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation, 13*, 284–302.

Lin, Q. Z., & Chen, J. Y. (2013). A novel micro-population immune multiobjective optimization algorithm. *Computers & Operations Research, 40*, 1590–1601.

Liu, H. L., Gu, F. Q., & Zhang, Q. F. (2014). Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems. *IEEE Transactions on Evolutionary Computation, 18*, 450–455.

Martinez, S. Z., & Coello Coello, C. A. (2011). A multi-objective particle swarm optimizer based on decomposition. In *Proceedings of the 13th annual genetic and evolutionary computation conference* (pp. 69–76).

Nayeri, P., Yang, F., & Elsherbeni, A. Z. (2013). Design of single-feed reflectarray antennas with asymmetric multiple beams using the particle swarm optimization method. *IEEE Transactions on Antennas and Propagation, 61*, 4598–4605.

Nebro, A. J., Durillo, J. J., Garcia-Nieto, J., Coello Coello, C. A., Luna, F., & Alba, E. (2009). SMPSO: A new PSO-based metaheuristic for multi-objective optimization. In *Proceedings of IEEE symposium on computational intelligence in multi-criteria decision-making* (pp. 66–73).

Peng, W., & Zhang, Q. F. (2008). A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In *Proceedings of IEEE international conference on granular computing* (pp. 534–537).

Samanlioglu, F. (2013). A multi-objective mathematical model for the industrial hazardous waste location-routing problem. *European Journal of Operational Research, 226*, 332–340.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated Genetic algorithms. In *Proceedings of the first international conference on genetic algorithms* (pp. 93–100).

Sierra, M. R., & Coello Coello, C. A. (2005). Improving PSO-based multi-objective optimization using crowding, mutation and epsilon-dominance. *Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science, 3410*, 505–519.

Sindhya, K., Miettinen, K., & Deb, K. (2013). A hybrid framework for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation, 17*, 495–511.

Tang, L. X., & Wang, X. P. (2013). A hybrid multiobjective evolutionary algorithm for multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation, 17*, 20–46.

Unler, A., & Murat, A. (2010). A discrete particle swarm optimization method for feature selection in binary classification problems. *European Journal of Operational Research, 206*, 528–539.

Wang, Y. J., & Yang, Y. P. (2010). Particle swarm with equilibrium strategy of selection for multi-objective optimization. *European Journal of Operational Research, 200*, 187–197.

Zhan, Z. H., Li, J. J., Cao, J. N., Zhang, J., Chung, H. S. H., & Shi, Y. H. (2013). Multiple populations for multiple objectives: A coevolutionary technique for solving multiobjective optimization problems. *IEEE Transactions on Cybernetics, 43*, 445–463.

Zhang, Q. F., & Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation, 11*, 712–731.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation, 8*, 173–195.

Zuo, X. Q., Zhang, G. X., & Tan, W. (2014). Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. *IEEE Transactions on Automation Science and Engineering, 11*, 564–573.