An External Archive-Guided Multiobjective Particle Swarm Optimization Algorithm

Qingling Zhu, Qiuzhen Lin, Weineng Chen, *Member, IEEE*, Ka-Chun Wong, Carlos A. Coello Coello, *Fellow, IEEE*, Jianqiang Li, Jianyong Chen, *Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—The selection of swarm leaders (i.e., the personal best and global best), is important in the design of a multiobjective particle swarm optimization (MOPSO) algorithm. Such leaders are expected to effectively guide the swarm to approach the true Pareto optimal front. In this paper, we present a novel external archive-guided MOPSO algorithm (AgMOPSO), where the leaders for velocity update are all selected from the external archive. In our algorithm, multiobjective optimization problems (MOPs) are transformed into a set of subproblems using a decomposition approach, and then each particle is assigned accordingly to optimize each subproblem. A novel archive-guided velocity update method is designed to guide the swarm for exploration, and the external archive is also evolved using an immune-based evolutionary strategy. These proposed approaches speed up the convergence of AgMOPSO. The experimental results fully demonstrate the superiority of our proposed AgMOPSO in solving most of the test problems adopted, in terms of two commonly used performance measures. Moreover, the effectiveness of our proposed archive-guided velocity update method and immune-based evolutionary strategy is also experimentally validated on more than 30 test MOPs.**

***Index Terms*—Evolutionary algorithm (EAs), multiobjective optimization problems (MOPs), particle swarm optimization (PSO).**

I. Introduction

IN many real-world engineering applications, we normally face problems in which we aim to simultaneously optimize multiple (possibly conflicting) objectives [1]. They are termed multiobjective optimization problems (MOPs). Due to the natural conflicts arising among the objectives, the improvement of one objective may deteriorate the others. As a consequence, a set of tradeoff solutions is generated (i.e., solutions in which it is not possible to improve one objective without worsening another). This is called the Pareto optimal set (POS) and their corresponding mapping in objective space is termed Pareto optimal front (POF). In order to provide solutions that are of practical use, it is desirable to obtain a set of uniformly distributed solutions that are as close as possible to the true POF.

Multiobjective evolutionary algorithms (MOEAs) have been substantially studied to tackle MOPs in recent years. They have been found to provide a very promising performance in solving different types of MOPs [2]–[12]. Based on the selection mechanisms they adopt, most of the existing MOEAs can be classified into the following three classes. The first class consists of Pareto-based MOEAs, which incorporate the Pareto optimality concept into their selection process. Two representative MOEAs are NSGA-II [2] and SPEA2 [3]. The second class consists of indicator-based MOEAs, which use a performance indicator (e.g., hypervolume [4]) as their density estimator to guide the search. Two MOEAs that are representative of this category are IBEA [5] and SMS-EMOA [6]. The last class consists of decomposition-based MOEAs, which transform an MOP into a set of subproblems and then optimize them in a collaborative manner. Approaches in this category include MOEA/D [7] and MOEA/D-IR [8]. More recently, some hybridized algorithms based on both Pareto dominance and decomposition approach have also been proposed, such as MMOPSO [9], ND/DPP [10], MOEA/DD [11], and BCE [12]. A survey of decomposition-based MOEAs recently published can be found in [13].

Particle swarm optimization (PSO) has also been studied to tackle MOPs in recent years. Almost all types of PSO approaches are designed by mimicking the social cooperative and competitive behavior of bird flocking and fish schooling [14]. In its origins, PSO was mostly applied to solve single-objective optimization problems (SOPs), due to its fast convergence speed and easy implementation [15], [16]. The promising results of PSO in solving SOPs validated its effectiveness and efficiency of locating the optima, especially in a large and complex problem landscape. This also motivated researchers to extend PSO for tackling MOPs. However, when designing a multiobjective PSO (MOPSO) algorithm, there are two particular issues to be addressed.

Manuscript received February 23, 2017; accepted May 18, 2017. Date of publication June 12, 2017; date of current version August 16, 2017.This work was supported in part by the National Natural Science Foundation of China under Grant 61402291, Grant 61672358, and Grant 61622206, and in part by CONACyT under Grant 221551. This paper was recommended by Associate Editor P. N. Suganthan. *(Corresponding author: Qiuzhen Lin.)*

Q. Zhu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Q. Lin, J. Li, and J. Chen are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: qiuzhlin@szu.edu.cn).

W. Chen and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China.

K.-C. Wong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

C. A. Coello Coello is with the Department of Computer Science, CINVESTAV-IPN (Evolutionary Computation Group), Mexico City 07300, Mexico.

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

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/TCYB.2017.2710133

The first issue is the selection of the global best (*g*best) and the personal best (*p*best) in an MOPSO algorithm. This is mainly due to the fact that, no a single best solution but rather a set of Pareto optimal solutions exist in tackling MOPs. In single-objective PSO (SOPSO), the swarm leaders (i.e., *g*best and *p*best) can be easily marked, since *g*best and *p*best are the best values, respectively visited by the entire swarm and each particle so far. However, in an MOPSO algorithm, multiple candidates (i.e., all the nondominated solutions) can be nominated as *g*best and *p*best. As the search direction of each particle is simultaneously guided by *g*best and *p*best, the selection of them has a significant impact on the performance of an MOPSO algorithm. The second one is the rapid loss of diversity due to its fast convergence speed, as pointed out in [17]. Such behavior may lead to premature convergence or get stuck in local optima, not only in SOPSO, but even more seriously in an MOPSO algorithm. In order to address this issue, some existing MOPSOs have adopted a perturbation operator on each particle [9], [18], as well as the adaptive control of the acceleration coefficients in the velocity update formula [19], and different selection mechanisms for *p*best and *g*best, with the aim to better guide the swarm without experiencing a quick loss of population diversity [20].

Based on the above issues, one key problem in an MOPSO algorithm is to choose the swarm leaders, i.e., *g*best and *p*best, in order to provide a correct search direction for all the particles. This helps to speed up the convergence, and also to maintain the population diversity if properly selected. Inspired from the direction-guided search approaches in [21]–[24], useful direction information can be extracted from the external archive to better guide the search of a particle swarm optimizer. We believe that such approaches may be very suitable to select the swarm leaders in MOPSOs. Therefore, in this paper, we propose an external archive guided MOPSO algorithm (AgMOPSO) that uses the information of external archive to guide the particle swarm to search. In our approach, all the swarm leaders (i.e., *p*best and *g*best) are appropriately selected from the external archive. To maintain diversity, a decomposition approach [7] is used in AgMOPSO to transform an MOP into a set of SOPs and then each SOP is accordingly optimized by using one particle. Each particle will be guided by three swarm leaders, i.e., *p*best, local best (*l*best), and *g*best, taken from the external archive. To promote the convergence speed, the individuals in the external archive are first evolved by an immune-based evolutionary strategy, which is helpful to guide the particles using a PSO-based search. Comparing to the existing MOPSO algorithms, the novel aspects of our proposed AgMOPSO are listed as follows.

1) An archive-guided velocity update approach is designed in AgMOPSO, which is aimed to exploit information related to defining a search direction from the external archive. As a decomposition approach is used to transform MOPs into a set of subproblems, each particle is guided by three leaders selected from the external archive, in order to optimize the corresponding subproblem.
2) An immune-based evolutionary strategy is run on the external archive. It helps to speed up the convergence using the clonal selection paradigm, as the swarm leaders are all taken from the external archive. Therefore, the improvement of individuals in the external archive will be conducive to guide the PSO-based search, thus providing a fast approximation to the true POF.
3) The selection of *p*best, *l*best, and *g*best is redefined in AgMOPSO. Generally, *p*best, *l*best, and *g*best are respectively, the best values visited by each particle, the local swarm, and the entire swarm. However, in AgMOPSO, as a decomposition approach is exploited to transform MOPs into a set of subproblems, our purpose is to optimize all the subproblems simultaneously. Therefore, *p*best, *l*best, and *g*best are regarded to be the best values in each subproblem, the neighboring subproblems and all the subproblems, respectively. In this way, AgMOPSO is devoted to optimizing each subproblem by using the proposed velocity update approach.

The rest of this paper is organized as follows. Section II introduces the related background, including the basic concepts related to MOPs, decomposition approaches, multiobjective immune algorithms (MOIAs), some existing MOPSOs, and direction-guided evolutionary algorithms (EAs). In Section III, the details of AgMOPSO are given, where the immune-based evolutionary strategy, PSO-based search and archive update are, respectively, described in detail. Our experimental studies are presented in Section IV, which compares AgMOPSO to two current MOPSOs and three state-of-the-art MOEAs. Moreover, the advantages of our proposed immune-based evolutionary strategy and archive-guided velocity update approach are also validated in Section IV. Section V presents an extension of AgMOPSO to handle constraints and to solve a real word engineering problem. Finally, our conclusions and future work are provided in Section VI.

II. Related Background

In this section, the related background of our work is introduced. First, a brief introduction to MOPs and decomposition approaches is provided. Since our external archive is further evolved by an immune-based evolutionary strategy, some relevant MOIAs are introduced with their advantages to speed up the convergence. Finally, some representative MOPSOs and direction-guided EAs are also briefly reviewed to illustrate the novelties of our proposed approach.

A. MOPs

Generally, an MOP can be formulated as follows:

$$\begin{aligned} \text{minimize}: \ & F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \\ \text{subject to} \ & g_j(x) \geq 0, j = 1, \ldots, J \\ & h_k(x) = 0, k = 1, \ldots, K \end{aligned} \quad (1)$$

where J and K are the numbers of inequality and equality constraints, respectively. $x = (x_1, x_2, \ldots, x_n)$ is an n-dimensional decision vector bounded in decision space Ω. The mapping function $F : \Omega \rightarrow R^m$ defines m objective functions and R^m

is called the objective space. Due to the conflicts among the objectives, no single solution can optimize all the objectives simultaneously. The best tradeoff solutions can be found using the definitions of Pareto dominance. A solution x is said to dominate another solution y (denoted as $x \succ y$) if and only if $\forall i \in \{1, 2, \ldots, m\}, f_i(x) \leq f_i(y)$ and at least $\exists j \in \{1, 2, \ldots, m\}$, $f_j(x) < f_j(y)$. A solution x is said to be Pareto optimal if and only if $\neg\exists y \in \Omega : y \succ x$.

B. Decomposition Approach

Decomposition approaches adopted in MOEAs include the weighted sum, Tchebycheff and boundary intersection approaches. As discussed in [7], the boundary intersection method has shown certain advantages over the other two approaches, so it is used in our algorithm for decomposing MOPs. This approach uses the predefined weighted vectors λ and a penalty value θ to minimize the distance d_1 to the utopian vector and the direction error to the weighted vector d_2 from the solution in objective space, as defined by

$$\text{minimize} : g\big(x|\lambda, z^*\big) = d_1 + \theta d_2 \tag{2}$$

where z^* is the vector including the minimum value of each objective, and d_1, d_2 are calculated as follows:

$$d_1 = \frac{\left\|(F(x) - z^*)^T\lambda\right\|}{\|\lambda\|} \text{ and } d_2 = \left\|(F(x) - z^*) - d_1\frac{\lambda}{\|\lambda\|}\right\|.$$

C. Relevant MOIAs

The fact that the external archive in our algorithm is evolved using the clonal selection mechanism helps to speed up its convergence [25], [26]. Thus, some representative MOIAs that incorporate clonal selection are briefly reviewed.

In [27], a multiobjective immune system algorithm was proposed based on the clonal selection principle to produce the clones of the individuals with high affinities. In [28], an immune dominance clonal multiobjective algorithm was reported, by using the concept of the antibody–antibody affinity to reflect the similarity among individuals. It was also applied to solve dynamic MOPs in [29] and was further improved by using a novel nondominated neighbor-based selection mechanism in [30]. In [31], a novel evolutionary MOIA was presented with a novel clonal selection scheme based on the diversity of the evolving population.

Recently, several MOIAs have been reported with a faster convergence and better mechanisms to maintain diversity. For example, a hybrid immune multiobjective optimization algorithm [32] was designed to combine Gaussian and polynomial-based mutations (PMs). It was further enhanced by an adaptive mutation operator [33] and a novel adaptive differential evolution (DE) operator [34] with a fine-grained selection mechanism. A novel immune clonal algorithm [26] was put forward to solve complex MOPs, using a full cloning scheme and a novel antibody population updating operation. A novel MOIA called IMADE was designed in [35], presenting a novel DE-based recombination as the search paradigm used after clonal selection. In [25], a hybrid evolutionary framework for MOIAs was reported to evolve subpopulations using multiple evolutionary strategies.

Some experimental results with the above MOIAs have validated that the clonal selection mechanism used for evolution helps to speed up the convergence, particularly in some simple MOPs without variable dependence. These results motivated us to use an immune-based evolutionary strategy for evolving the individuals in the external archive of our proposed approach. These enhanced individuals in the external archive will in turn help to effectively guide the PSO-based search.

D. Existing MOPSOs

PSO was originally designed to tackle SOPs and showed a fast convergence speed. Most multiobjective extensions of PSO rely on Pareto ranking and a few on decomposition methods. Thus, most existing MOPSOs can be classified into these two categories: 1) Pareto-based MOPSOs and 2) decomposition-based MOPSOs.

The first type of MOPSOs adopts Pareto ranking to determine *p*best and *g*best. The *g*best particle is generally assigned with one of the nondominated solutions found from the entire swarm and it is used to guide the swarm to approach the POF. MOPSO [18], OMOPSO [36], SMPSO [37], MOCLPSO [38], 2LB-MOPSO [39], CMPSO [40], and pccsAMOPSO [17], belong to this category. In [18], MOPSO was designed by using the Pareto dominance relationship and an adaptive grid to update the external archive. OMOPSO was reported in [36] to adopt the concept of ε dominance and crowding-distance information to identify the list of leaders. To avoid the so-called "swarm explosion" effect in OMOPSO, SMPSO was presented with a velocity constriction procedure during the particles movement [37]. In [38], Pareto dominance concept and external archive technique were integrated in MOCLPSO to handle MOPs, and 2LB-MOPSO [39] was proposed to run a fine-grained search around the vicinity of the best found fronts. In [40], CMPSO was reported with a novel coevolutionary technique for PSO to solve MOPs and pccsAMOPSO [17] was designed based on a parallel cell coordinate system.

The second kind of MOPSOs adopts a decomposition approach to transform an MOP into a set of SOPs and then a SOPSO can be directly applied for each SOP. Such MOPSOs include MOPSO/D [41], SDMOPSO [42], and dMOPSO [43]. MOPSO/D may be the first attempt to embed a decomposition approach into an MOPSO. To tackle the drawback of MOPSO/D, SDMOPSO was proposed to fully exploit the salient properties of neighborhood relations in PSO. In [43], dMOPSO was presented as an approach that fully relies on decomposition. A set of *g*best particles, which can give the best scalar aggregated values for all subproblems, are used to update the position of each particle. However, as pointed out by Moubayed *et al.* [20], the absence of Pareto dominance in dMOPSO may lead to a failure in covering the entire POF in some complex MOPs.

Recently, some MOPSOs, such as D^{2}MOPSO [20] and MMOPSO [9], have been designed by combining Pareto dominance and decomposition approach. The original version of D^2 MOPSO was proposed in [44], attempting to use a hybridization of Pareto dominance and decomposition approach for solving MOPs. An enhanced version of D^{2}MOPSO was also presented by the same authors [20]. This approach introduces a new mechanism to select leaders and a novel archiving technique to maintain the nondominated particles based on the crowding-distance values [2] in both objective and decision spaces. Following the framework of D^{2}MOPSO, MMOPSO was introduced in [9] to use two search strategies for velocity update, aiming to concurrently promote the convergence speed and maintain the population diversity. Moreover, an evolutionary search strategy is further applied to each individual of the external archive for speeding up the convergence.

E. Direction-Guided EAs

Steering the optimization process is an effective and efficient way to design an MOEA. Among the numerous search strategies currently available, direction-guided search, which drives the population to explore some interesting areas, has been found to be quite promising [24]. To tackle SOPs, a neighbor guided selection scheme and a direction induced mutation strategy were designed in [21] to respectively exploit the neighborhood and direction information from the population. In [22], the reproduction mechanism was also enhanced through an evolutionary path. This evolutionary path can be cumulatively learned during the evolutionary process and it is exploited to produce a new solution among its central area.

To tackle MOPs, a direction-based MOEA (DMEA) [45] was designed to guide a population for evolving using the directions of improvement. Then, several new features were further embedded into DMEA to make it more robust (named DMEA-II [46]), and a new niching method was designed for DMEA in [47]. In [48], an evolutionary multiobjective simulated annealing algorithm with a two-phase strategy was designed to maintain the diversity of the search directions. This strategy uses fixed and adaptive search directions, respectively, in the first and second search phases, in order to search the objective space more effectively. Recently, a new DE variant for MOPs was studied in [24]. This approach uses the information across generations to model the search directions as guidance, such that both the convergence and the diversity during the evolution can be steered.

According to our survey, although additional archives are used in some direction-guided MOEAs, little attention has been paid to study the use of direction information to guide the particle swarm in MOPSOs. Therefore, in each iteration of AgMOPSO, the archive is first evolved by an immune-based evolutionary strategy, and then this evolved archive can be used to guide the particle swarm for further exploration.

III. Proposed Algorithm

In this section, the details of our proposed AgMOPSO algorithm are introduced. The distinct feature of this algorithm is the use of an external archive containing all the elitist individuals as the swarm leaders for the particles. To speed up the convergence, this external archive is first evolved using an immune-based evolutionary strategy. The algorithmic framework of AgMOPSO is shown in Fig. 1, where P, A, and S, respectively, denote the particle swarm, the external archive,

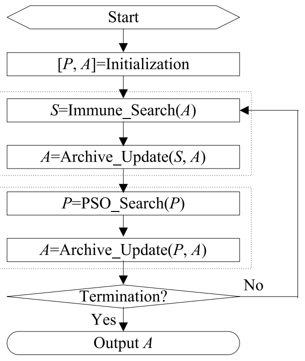


Fig. 1. Algorithmic framework of AgMOPSO.

and a temporary population after immune-based evolutionary search. After initialization, three main procedures are iteratively run in AgMOPSO, i.e., an immune-based evolutionary search, a PSO-based search, and an archive update. Please note that the immune-based evolutionary search is only performed on the external archive, and then the renewed individuals in the external archive are used to guide the swarm for exploration. Once the termination condition is satisfied, the individuals in the external archive are reported as the final result.

A. Initialization

The pseudo-code for the initialization is described in **Algorithm 1**, where *fes* denotes the count for function evaluation. As no prior knowledge of the search landscape is available, an initial swarm $P = \{x_1, \dots, x_N\}$ is randomly sampled in decision space Ω. To decompose an MOP into a set of SOPs using (2), a set of weight vectors $W = \{\lambda^1, \dots, \lambda^N\}$ is uniformly sampled from a unit simplex using the approach in [49]. Each weight vector is associated with a subproblem, thus the number of weight vectors is equal to the population size. After that, the Euclidean distances between any two weight vectors are computed and the neighborhood set $B_i = \{i_1, \dots, i_T\}$ for each weight vector λ^i ($i \in \{1, \dots, N\}$) is built. Moreover, in (2), the ideal objective vector used in this paper is approximated by using the minimum value of each objective in current swarm, i.e., $z_i^* = \min\{f_i(x)|x \in P\}$, for all $i \in \{1, \dots, m\}$. At last, the external archive A is updated by adding all the nondominated solutions in P to A.

B. Immune-Based Evolutionary Search

The immune-based evolutionary search is composed of two steps. First, the less crowded solutions in archive A are proportionally cloned to get the mating population E. As shown in Fig. 2(a), the selected solutions in $PC = \{a_1, a_2, a_3, a_4, a_5\}$ represent the sparse areas that need to be searched. Second, the mating population undergoes recombination and mutation to get the child population S. This proportional clonal principle will lead to the result that the less crowded area in the archive is assigned with more clones and of course more computational resources are allocated to these areas. As shown in Fig. 2(b), the solutions a_1, a_2, a_3, a_4, a_5 with different crowding degrees will, respectively, get 8, 3, 4, 2, and

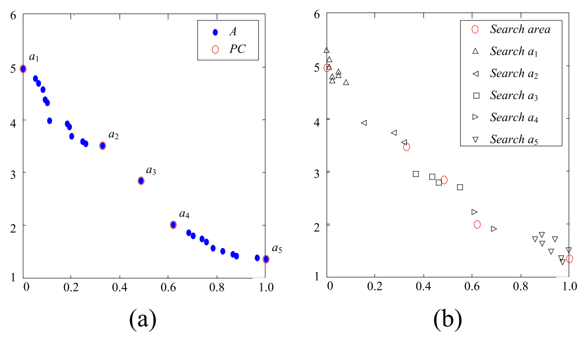


Fig. 2. Procedure of immune-based evolutionary search. (a) Before cloning operator. (b) After cloning operator.

8 clones of solutions. The details of the cloning operator and the evolutionary operators are respectively, introduced below.

1) Cloning Operator: It is assumed that the population after cloning is E with size N and the elitist population used for cloning is PC with size NC. Please note that NC is smaller than N, usually set as $N/5$ [34]. At first, NC individuals with the largest values of crowding degree are selected from the external archive A, to build the elitist population $PC = \{a_1, \ldots, a_{NC}\}$. Then, cloning is activated and the cloned population E is generated, as follows:

$$E = \bigcup_{i=1}^{NC} \{q_i \otimes a_i\} \tag{3}$$

where operation $q_i \otimes a_i$ means to duplicate a_i with the number of q_i, and q_i stands for the number of clones corresponding to each individual $a_i (i = 1, 2, \ldots, NC)$, calculated by

$$q_i = \left\lceil N \times \frac{\mathrm{CD}(a_i)}{\sum_{j=1}^{NC} \mathrm{CD}(a_j)} \right\rceil \tag{4}$$

where $\mathrm{CD}(a_i)$ is the crowding-degree value of individual $a_i (i = 1, 2, \ldots, NC)$. The estimation of the crowding degree can use the crowding distance method in NSGA-II [2] or the niching method in BCE [12]. As studied in [52], the crowding distance method is more suitable to estimate the crowding status in bi-objective problems, while the niching method may be more effective to reflect the crowding degree for MOPs with more than two objectives. Note that once the individuals are located in the boundary of objective space, their crowding degrees are set as twice of the maximal crowding degree except for the boundary individuals.

The pseudo-code of this operator is shown in **Algorithm 2**, where NC is the size of the elite population for cloning and the function **CrowdingDegreeAssignment**(P) calculates the value of crowding degree for each individual in P using (4). Another function $PC =$ **SelectforClone**(PC) will return NC individuals with the largest values of crowding degree in PC.

2) Evolutionary Operators: As shown in [25], the cloning operator helps to speed up the convergence, especially on some simple MOPs without variable dependence. After cloning, this cloned population E will undergo two evolutionary operators, i.e., simulated binary crossover (SBX) and PM [50]. To clearly introduce the immune-based evolutionary strategy, its pseudo-code is illustrated in **Algorithm 3**, where SBX(E_i, E_j) means to apply SBX on parent solutions E_i and E_j; C_1 and C_2 are the resultant children solutions generated from SBX; PM(C_k) indicates the execution of PM on C_k. Note that there is a small

Algorithm 1 Initialization

```
1:  $A = \{\}, fes = 0$
2:  for $i = 1$ to $N$
3:      randomly generate a particle $x_i$ and evaluate the objectives of $x_i$
4:      add $x_i$ to the population $P$
5:  end for
6:  initialize $N$ weight vectors $\lambda^1, \ldots, \lambda^N$
7:  set $z_i^* = \min\{f_i(x)|x \in P\}$ for $i \in \{1, \ldots, m\}$
8:  for $i = 1$ to $N$
9:      $B_i = \{i_1, \ldots, i_T\}$ // where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the $T$ closest to $\lambda^i$
10: end for
11: copy all the non-dominated solutions from $P$ to $A$
```

Algorithm 2 $E =$ CloneOperator(A)

```
1:  $PC = A$
2:  if ($|PC| > NC$)
3:      CrowdingDegreeAssignment($PC$)
4:      $PC =$ Sort($PC$) //sort $PC$ according to crowding distance
5:      $PC =$ SelectforClone($PC$)
6:  end if
7:  for $i$=1 to $|PC|$
8:      calculate $q_i$ according to (4)
9:      clone $q_i$ individuals of $a_i$ and add them to $E$
10: end for
11: return $E$
```

Algorithm 3 $S =$ **Immune_Search**(A)

```
1: $E =$ CloneOperator($A$) (Algorithm 2)
2: for $i$=1 to $|E|$
3:     generate a random integer $j$ in $[1, |E|]$
4:     $\{C_1, C_2\} =$ SBX($E_i$, $E_j$)
5:     generate a random integer $k$ in [1,2]
6:     $S_i =$ PM($C_k$)
7: end for
8: return $S$
```

Algorithm 4 $A =$ **Archive_Update**(S, A)

```
1:  for $i$=1 to $|S|$
2:      for $j$=1 to $|A|$
3:          $flag =$ CheckDominance($S_i$,$A_j$)
4:          if $flag$ == 1 // $A_j$ is dominated by $S_i$
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 $flag$ != -1 // if any individual in $A$ does not dominate $S_i$
12:         add $S_i$ to $A$
13:         if $|A| > N$
14:             CrowdingDegreeAssignment($A$)
15:             delete the most crowded one
16:         end if
17:     end if
18: end for
19: return $A$
```

Algorithm 5 Complete Algorithm of AgMOPSO

```
1: Initialization (Algorithm 1)
2: while $fes < max\text{-}fes$
3:     $S$ = Immune_Search ($A$) (Algorithm 3)
4:     evaluate $S$ and set $fes = fes + |S|$
5:     $A$ = Archive_Update($S$, $A$) (Algorithm 4)
6:     update the reference point $z^*$
7:     for $i$=1 to $N$
8:         $p\text{best}_i = A_i$        //Selection for $p\text{best}_i$
9:         for $j$=2 to $|A|$
10:            if $g(p\text{best}_i|w^i, z^*) > g(A_j|w^i, z^*)$
11:                $p\text{best}_i = A_j$
12:            end if
13:        end for
14:        select $l\text{best}_i$ from the neighbors of sub-problem $i$
15:        randomly select $g\text{best}_i$ from archive $A$
16:        calculate $v_i$ use (5)
17:        calculate $x_i$ use (6)
18:        evaluate the new particle $x_i$ and set $fes = fes + 1$
19:    end for
20:    $A$ = Archive_Update($P$, $A$) (Algorithm 4)
21:    update the reference point $z^*$
22: end while
23: Output $A$
```

Algorithm 6 A = Archive_Update_Constraint(S, A)

```
1: $U=A \bigcup S$
2: $I=\{\}$, $O=\{\}$
3: for each $x \in U$
4:     if $CV(x) > 0$ // $x$ is infeasible solution
5:         $I = I \bigcup x$
6:     else
7:         $O=O \bigcup x$
8:     end if
9: end for
10: if $|O|> N$
11:    Archive_Update($O$, $A$); // use $O$ to update $A$ (Algorithm 4)
12: else
13:    for each $i$=1 to $|O|$// add the feasible solutions
14:        add $O_i$ to $A$
15:    end for
16:    sort $I$ in descending order according to $CV$
17:    while ($|A|< N$) //add the infeasible solutions
18:        for each $i$=1 to $|I|$
19:            find the subproblem $k$ that is nearest to $I(i)$
20:            if subproblem $k$ is not marked
21:                add $I(i)$ to $A$
22:            end if
23:            mark subproblem $k$
24:        end for
25:    end while
26: end if
27: return $A$
```

probability to select two same individuals for running SBX. In this case, the PM operator will further perturb the parent to produce a new offspring, even though SBX does nothing. After that, a new solution set S is generated, which will be added into the external archive using the archive update operation as introduced in Section III-D.

C. PSO-Based Search

In original PSO algorithm, the velocities of the particles are usually updated using the positional information of the $p$best and $g$best particles. However, the selection of $p$best and $g$best is particularly difficult when using PSO to tackle MOPs, as multiple equally optimal solutions (i.e., nondominated solutions) are available. Here, a novel velocity update approach is presented, aiming to optimize all the subproblems, as follows:

$$v_i(t+1) = w \cdot v_i(t) + F_1 \cdot (p\text{best}_i - x_i(t)) + F_2 \cdot (l\text{best}_i - g\text{best}_i) \quad (5)$$

where x_i is the current evolved particle in P, and t indicates the iteration number; $p\text{best}_i$ is the individual in the external archive A that can give the best result for the subproblem i, $l\text{best}_i$ is the individual in the external archive A that can give the best result for a random subproblem selected from the neighboring set B_i (as defined in line 9 of **Algorithm 1**), and $g\text{best}_i$ is randomly selected from the external archive A.

After the update of velocity, the position of particle x_i is renewed as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (6)$$

There are three parts in our new PSO search method. First, $w \cdot v_i(t)$ is the "inertial" part same as other PSO search methods. The second part is $F_1 \cdot (p\text{best}_i - x_i(t))$, which guides the current particle to approach the best individual for the current subproblem i. The step size F_1 is set as d_1 in (2), which indicates the distance of $p$best from the current subproblem. That is to say, if $p$best is far away from the current subproblem, the step size is set to be large; otherwise, it is set to be small. The last part is $F_2 \cdot (l\text{best}_i - g\text{best}_i)$, which is the differential vector similar to "DE/rand/1" [51]. In this way, the proposed velocity update method in (5) not only has the search pattern of PSO, but also inherits the effective search behaviors of DE.

D. Archive Update

After finishing the immune-based evolutionary search or the PSO-based search, the newly found nondominated solutions are collected into the external archive. Given the finite size of the external archive, a proper selection mechanism is necessary for updating it since the number of nondominated solutions may be very large. Such selection mechanism has significant impact on performance, as it helps to guide the search toward the true POF. In this paper, a popular archive update mechanism [9], [20], [52] is used. This selection mechanism is designed based on both Pareto dominance and the crowding degree. Assuming that, the newly generated solution set is S and the external archive is A, the pseudo-code of the archive updating mechanism can be briefly described in **Algorithm 4**. 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 to x. Another function **CrowdingDegreeAssignment**(A) will calculate the value of crowding degree for each solution in A.

E. Complete Algorithm of AgMOPSO

The above sections have introduced the main components of AgMOPSO, i.e., immune-based evolutionary strategy, PSO-based search, and archive update operator. Other implementation details are further clarified in the pseudo-code of AgMOPSO, as illustrated in **Algorithm 5**, where *fes* and *max-fes*, respectively, denote the count of current function evaluation and the maximum number of function evaluation.

At first, the initialization is processed in line 1 of AgMOPSO, as described in **Algorithm 1**. After that, AgMOPSO starts the loop of evolutionary process in lines 2–22. In the first search phase, the immune-based evolutionary strategy is operated on the external archive A (in line 3), and a children population S is produced. The pseudo-code of this procedure was described in **Algorithm 3**. After that, the objective function values of all the individuals in S are computed in line 4, and they are coupled with the external archive A to run **Algorithm 4** for the archive update procedure (in line 5). Then, the reference point z^* used in (2) is updated (in line 6). In the PSO-based search phase, the selection of $p\text{best}_i$, $l\text{best}_i$, and $g\text{best}_i$ from the external archive A is performed in lines 7–15, as introduced in Section III-C. After that, each particle x_i is evolved by the PSO-based search pattern, as illustrated in lines 16 and 17, and its new objective functions are evaluated in line 18. Also, the updated particle swarm P and external archive A are used to run **Algorithm 4** (line 20), and the reference point z^* is updated again in line 21. The above evolutionary phase will be repeated until the predefined maximum number of function evaluation, *max-fes*, is reached. At the end of this algorithm, the external archive A is reported as the final POF.

F. Discussion

Based on the implementation of AgMOPSO described above, this section discusses the differences between AgMOPSO and some existing algorithms, such as D^{2}MOPSO [20], MMOPSO [9], and BCE–MOEA/D [12], as they all adopt the similar idea of combining Pareto dominance and decomposition approaches. Please note that, as described in Section I, the novel aspects of AgMOPSO include an immune-based evolutionary strategy to enhance the solution's quality in external archive and an archive-guided velocity update approach to guide the PSO-based search. These two search patterns are mutually co-operated in AgMOPSO to speed up the convergence, which is the essential difference with D^{2}MOPSO, MMOPSO, and BCE-MOEA/D. Their detail differences are further clarified as follows.

1) Differences Between AgMOPSO and D^{2}MOPSO:

- In D^{2}MOPSO, evolutionary search is not run to enhance the archive. However, immune-based search is performed in AgMOPSO to evolve the archive, which aims to search the least crowded areas and results in a better diversity and convergence in the archive as discussed in Section III-B.
- The PSO-based search behavior in D^{2}MOPSO is different from that driven by (5) in Section III-C. That is to say, the $p$best and $l$best in D^{2}MOPSO are both selected dependent on the current subproblem, which may ignore some useful information from the local and global neighbors. As discussed in Section III-C, the PSO-based search of AgMOPSO in (5) is driven by three parts, i.e., the inertial part, the subproblem guided part and the differential vector part. In this way, $p$best, $l$best, and $g$best in AgMOPSO are, respectively, selected based on the different subproblems in the archive.
- For D^{2}MOPSO, two archives, i.e., the leaders archive and the external archive, are adopted. The leaders archive is used to collect the elite particles with larger crowding distances based on both objective and decision spaces. However, in AgMOPSO, only the external archive is used to keep the swarm leaders.

2) Differences Between AgMOPSO and MMOPSO:

- Although both algorithms evolve the archive, they use different evolutionary operators. MMOPSO only adopts a general evolutionary operator (SBX+PM), while AgMOPSO performs an additional cloning operator. Such operator emphasizes the search on sparse areas in objective space, as illustrated in Fig. 2.
- In MMOPSO, the velocity update formulations are composed by two aspects, i.e., local search and global search, which are controlled by a parameter δ. This search pattern is totally different from the archive-guided PSO-based search method in (5).

3) Differences Between AgMOPSO and BCE-MOEA/D:

- In BCE-MOEA/D, although a further search is also launched on the archive (i.e., Pareto criteria (PC) population in BCE-MOEA/D [12]), it behaves differently than the immune-inspired search in AgMOPSO. Both of them are encouraged to search the less crowded area, but the individuals with different crowding degrees in BCE-MOEA/D are all assigned with the same search strength. Whereas, in AgMOPSO, due to the use of the proportional cloning operator, the individuals with lower crowding degrees will be allocated with more clones which will result in a stronger search bias in that area.
- In AgMOPSO, the elite archive is exploited to guide the PSO swarm for searching; whereas, in BCE-MOEA/D, such information from the archive is not exploited at all to guide the search behavior of the Non-PC (NPC) population.
- The decomposition-based method (i.e., MOEA/D algorithm [7]) in BCE-MOEA/D is used to evolve the NPC population as replacement will be activated if the new solution is better. However, in AgMOPSO, the decomposition-based module is only used to select $p$best, $l$best, and $g$best from the archive.

IV. Experimental Studies

A. Test Problems

Comprehensive and diverse test problems were employed in order to assess the performance of AgMOPSO. First, the ZDT test problems were adopted. As some complicated features, such as variable linkages and objective function modality, are absent in the ZDT problems, they are not very challenging for most multiobjective algorithms. Thus, two other kinds of more difficult MOPs, i.e., the bi-objective WFG and the UF test problems were also used in light of their complicated features, including convexity, concavity, discontinuity, nonuniformity, and the existence of many local POFs. To further examine the performance of AgMOPSO in tackling MOPs with three objectives, the DTLZ test problems and UF8-UF10 were used in this paper. Moreover, the DTLZ and WFG test problems with 5 and 10 objectives were also used to further study the scalability of AgMOPSO. For ZDT1–ZDT3 and all the UF test problems, 30 decision variables were used; ZDT4 and ZDT6 were used with 10 decision variables; and WFG1-WFG9 were used with $2 \times (m - 1)$ position parameters and 20 distance parameters. For details on the ZDT, WFG, UF, and DTLZ test problems, please refer to [53]–[56], respectively.

B. Performance Measures

The goal of MOPs is to find a uniformly distributed set that is as close to the true POF as possible. In order to assess the performance among different compared algorithms, two performance measures, i.e., inverted generational distance (IGD) [57] and hyper-volume (HV) [58] were adopted here. It is believed that these two performance indicators can not only account for convergence, but also the distribution of final solutions. The true POFs for computing IGD were downloaded from http://jmetal.sourceforge.net/problems.html. The reference point for HV calculation was set to 1.1 times the nadir point of the true POF, i.e., $1.1 \times (0.5, \dots, 0.5)$ for DTLZ1, $1.1 \times (1.0, \dots, 1.0)$ for DTLZ2–DTLZ4, $1.1 \times (1.0, \dots, 1.0, 2.0 \times m)$ for DTLZ7, $1.1 \times (1.0, \dots, 1.0, 2.0 \times m)$ for WFG1-WFG9 (where m is the number of objectives).

C. Experimental Settings

In the experiments, in order to validate the performance of AgMOPSO in a convincing way, it was compared to three competitive MOEAs, i.e., NSGA-II [2], EAG-MOEA/D [59], BCE-MOEA/D [12], and two current MOPSOs (D^{2}MOPSO [20] and MMOPSO [9]).

To allow a fair comparison, the related parameters in all the compared algorithms were set according to their original references, as summarized in Table I. In Table I, N denotes the population size for all the algorithms; p_c and p_m are, respectively, the crossover and mutation probabilities; η_c and η_m are the distribution indexes of SBX and PM, respectively. w, c_1, and c_2 are the parameters used for the velocity update equations of MOPSO algorithms. For EAG-MOEA/D, BCE-MOEA/D, and AgMOPSO, T defines the size of the neighborhood regarding the weight vectors. The decomposition method in (2) is also adopted in the original paper of D^{2}MOPSO and MMOPSO. As the original EAG-MOEA/D was designed for combinatorial optimization, the evolutionary operators, and decomposition method in AgMOPSO were applied to EAG-MOEA/D, to make it more effective for continuous optimization and to achieve a more fair comparison.

Please note that the settings of N listed in Table I are only applicable for bi-objective test problems. Moreover, N is set to 105 for three-objective test problems, to 210 for five-objective test problems, and to 220 for ten-objective test problems. The maximum numbers of function evaluation were set to $250 \times N$ for ZDT, to $500 \times N$ for WFG and DTLZ, and to $3000 \times N$ for UF (N is the population size). In general, the size of the external archive is set the same as N. All the experiments were independently run 30 times. The mean values and the standard deviations (std) on IGD and HV were collected in the corresponding tables for performance comparison. Moreover, in order to ascertain statistical significances, the Wilcoxon's rank sum test was further performed to examine the statistical significance of the difference between the results obtained by AgMOPSO and those obtained by the other algorithms at the significance level $\alpha = 0.05$.

TABLE I
PARAMETERS SETTINGS OF THE COMPARED ALGORITHMS

Algorithms	Parameter settings
NSGA-II	$N = 100, p_c = 0.9, p_m = 1/n, \eta_c = 20, \eta_m = 20$
EAG-MOEA/D	$N = 100, p_c = 0.9, p_m = 1/n, \eta_c = 20, \eta_m = 20, T = 20$
BCE-MOEA/D	$N = 100, p_c = 1.0, p_m = 1/n, \eta_c = 20, \eta_m = 20, T = N/10$ $nr = N/100, k = 3$
D^{2}MOPSO	$N = 100, p_c = 0.9, p_m = 1/n, \eta_c = 20, \eta_m = 20,$ $w \in [0.1, 0.5], c_1 \in [1.5, 2.0], c_2 \in [1.5, 2.0]$
MMOPSO	$N = 100, p_c = 0.9, p_m = 1/n, \eta_c = 20, \eta_m = 20,$ $w \in [0.1, 0.5], c_1 \in [1.5, 2.0], c_2 \in [1.5, 2.0], \delta = 0.9$
AgMOPSO	$N = 100, p_c = 0.9, p_m = 1/n, \eta_c = 20, \eta_m = 20,$ $w \in [0.1, 0.5], F_2 = 0.5, T = 20$

D. Comparisons of AgMOPSO With Various Algorithms

1) Comparisons of AgMOPSO With NSGA-II, EAG-MOEA/D, D^{2}MOPSO, and MMOPSO: Please note that all the compared algorithms use the same crowding distance method [2] to estimate their crowding status among the solutions and adopt the same archive update method (**Algorithm 4**), in order to allow a fair comparison. This population update method [52] is an improved version of the approach in NSGA-II to prune the nondominated solutions and is also used in the original implementation of D^{2}MOPSO and MMOPSO.

Table II lists the mean and standard deviation (std) results of all the algorithms on the 31 test problems in terms of IGD. The best result obtained for each test problem was marked with boldface. As observed from Table II, AgMOPSO performs the best and presents a clear advantage over the other four algorithms on the majority of the test instances. More specifically, AgMOPSO obtains the best IGD results on 17 out of 31 test instances. The proportions of the test instances on which AgMOPSO performs better than NSGA-II, EAG-MOEA/D, D^{2}MOPSO, and MMOPSO are 22/31, 22/31, 26/31, and 22/31, respectively. Conversely, the proportions on which AgMOPSO is defeated by the peer algorithms are 5/31, 5/31, 3/31, and 2/31, respectively, for NSGA-II, EAG-MOEA/D, D^{2}MOPSO, and MMOPSO. Especially, D^{2}MOPSO shows poor performance on the test MOPs with many local POFs, such as ZDT4, DTLZ1, and DTLZ3. This is mainly because D^{2}MOPSO only performs the PSO search method, which may easily fall into local optima. MMOPSO and

AgMOPSO can overcome this shortcoming by further evolving the archive using other search patterns, and AgMOPSO performs even better with the use of immune-based evolutionary strategy. For DTLZ5 and DTLZ6 which have a degenerated curve, EAG-MOEA/D, MMOPSO, and AgMOPSO show a similar performance and are able to find good approximations of true POF as their corresponding mean values of IGD are under an accuracy level of 10^{-3}.

The HV results of all the 31 test problems are provided in Table S-I of the supplementary material. Similar observations from the IGD results can be found in the HV results. AgMOPSO also performs best on most of test instances, such as ZDT1-ZDT4, ZDT6, WFG1-WFG5, WFG7, UF1-UF3, UF8, DTLZ1, DTLZ2, DTLZ4, and DTLZ7. As indicated in the last row of Table S-I, in the supplementary material, the proportions on which AgMOPSO performs better than or similarly to NSGA-II, EAG-MOEA/D, D^2MOPSO, and MMOPSO, are 27/31, 26/31, 29/31, and 29/31, respectively.

Based on these IGD and HV comparison results, it is sufficient to conclude that AgMOPSO shows a better performance than NSGA-II, EAG-MOEA/D, D^2MOPSO, and MMOPSO on solving these test instances.

2) Comparison of AgMOPSO With BCE-MOEA/D: As BCE-MOEA/D also combines Pareto dominance and decomposition approach to keep the archive and the NPC population, it is also compared to AgMOPSO for solving different test MOPs with various objectives. The main differences of AgMOPSO and BCE-MOEA/D were clarified in Section III-F. As studied in [60], the performance of the compared algorithms will be significantly different when different density estimators are used. For this consideration, we use the same density estimator with BCE-MOEA/D in order to have a fair comparison, and this variant is called AgMOPSO-niche.

Table III presents their performance comparisons in terms of HV, when tackling ZDT, WFG, UF, and DTLZ test problems with different numbers of objectives. The best mean result of each problem was highlighted in boldface. As observed from Table III, AgMOPSO-niche obtains a better performance in 47 out of 63 test instances. According to the Wilcoxon's rank sum test, AgMOPSO-niche performs similarly to BCE-MOEA/D on 10 test instances. For the simple ZDT and the more complicated WFG test problems with two objectives, AgMOPSO-niche is found to have a significantly better performance, which is mainly brought by the use of immune-inspired evolutionary strategy, as this approach is more effective on simple MOPs without variable dependence [25]. Regarding UF1–UF7 with complicated POS, AgMOPSO-niche also shows a superior performance as it performs better than BCE-MOEA/D on UF1–UF3, UF7 and similarly to BCE-MOEA/D on UF4-UF6. This is mainly due to the use of archive-guided PSO search approach, in which each particle is used to optimize one particular subproblem as guided by the elitist individuals from the external archive. About the test MOPs with more than three objectives, AgMOPSO also performs better on most cases, while BCE-MOEA/D is only better on UF9 and DTLZ1 with three objectives, on DTLZ2 and WFG5 with five objectives, and on DTLZ3, WFG2, WFG4, WFG5, and WFG8 with ten objectives. In summary, as observed from the last row of Table IV, AgMOPSO performs better or similarly to BCE-MOEA/D on 53 out of 63 test instances. Conversely, AgMOPSO-niche is only defeated by BCE-MOEA/D on ten test instances. From the above discussion, it is reasonable to conclude that AgMOPSO also shows a superior performance over BCE-MOEA/D on tackling most of test instances adopted.

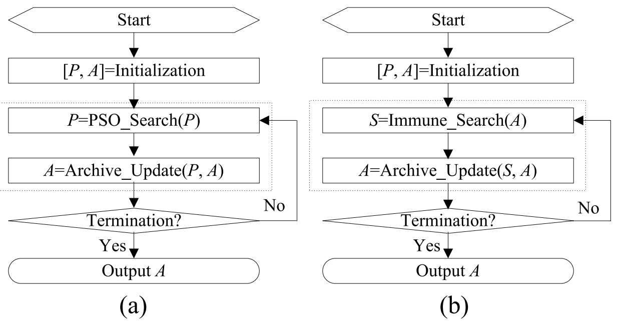


Fig. 3. Two variants of AgMOPSO. (a) AgMOPSO-I. (b) AgMOPSO-II.

E. Effectiveness of the Proposed Approaches

There are two search modules used in AgMOPSO, such as immune-based evolutionary search and PSO-based search. In order to study their effectiveness, AgMOPSO was further compared to two variants of AgMOPSO, i.e., AgMOPSO-I and AgMOPSO-II. AgMOPSO-I only runs the PSO-based search on the particle swarm as shown in Fig. 3(a), while AgMOPSO-II only uses the immune-based evolutionary search on the external archive as shown in Fig. 3(b).

The IGD comparison results of AgMOPSO-I, AgMOPSO-II, and AgMOPSO on all the 31 test problems are listed in Table IV. As observed from Table IV, AgMOPSO performs best on 21 out of 31 test problems regarding IGD when compared to AgMOPSO-I and AgMOPSO-II. The Wilcoxon's rank sum test results also reveal that AgMOPSO obtains better IGD results than AgMOPSO-I and AgMOPSO-II on 30 and 21 test problems, respectively. Moreover, AgMOPSO performs similarly to AgMOPSO-I on 1 test problem and to AgMOPSO-II on seven test problems. In other words, AgMOPSO performs better than or similarly to AgMOPSO-I and AgMOPSO-II on 31 and 28 out of 31 test problems regarding IGD. These experimental results clearly justify the usefulness of the evolution on the external archive as well as the effectiveness of the archive-guided PSO search in AgMOPSO.

As observed from Table IV, the performance of AgMOPSO-I seems very poor when compared to AgMOPSO. This is mainly because the proposed PSO-based search has to be driven by the elite individuals from external archive. When the archive cannot provide the guiding particles with good diversity and good convergence, the performance of PSO-based search in (5) will not work well. AgMOPSO-I may search for some undesirable subproblems, as the archive did not provide good leader information for the particle swarm. Especially for the problems with many local POFs, such as ZDT4, DTLZ1, DTLZ3, AgMOPSO-I is unable to jump out

TABLE II
PERFORMANCE COMPARISONS OF IGD VALUES

Problems	NSGA-II	EAG-MOEA/D	D^{2}MOPSO	MMOPSO	AgMOPSO
ZDT1	4.976E−3 (1.73E−4) –	3.757E−3 (1.02E−4) –	1.038E−2 (6.07E−3) –	3.936E−3 (4.56E−5) –	**3.701E−3 (2.83E−5)**
ZDT2	5.102E−3 (1.79E−4) –	2.113E−2 (8.19E−2) –	4.904E−1 (2.45E−1) –	2.414E−2 (1.11E−1) –	**3.828E−3 (3.15E−5)**
ZDT3	6.408E−3 (5.41E−3) –	3.142E−2 (3.73E−2) –	1.404E−2 (4.34E−3) –	4.413E−3 (4.28E−5) –	**4.367E−3 (5.23E−5)**
ZDT4	**7.654E−3 (2.45E−3)** +	2.357E−2 (3.18E−2) –	3.203E+0 (2.06E+0) –	2.342E−2 (4.42E−2) –	7.942E−3 (2.23E−2)
ZDT6	9.088E−3 (1.00E−3) –	3.132E−3 (2.13E−4) –	1.423E−2 (8.14E−3) –	3.635E−3 (2.31E−4) –	**2.997E−3 (9.51E−5)**
WFG1	7.374E−1 (2.75E−1) –	5.039E−1 (9.55E−2) –	9.037E−1 (6.57E−2) –	3.652E−1 (3.72E−2) –	**3.261E−1 (6.38E−2)**
WFG2	7.628E−2 (3.08E−2) –	9.998E−2 (7.85E−2) –	1.130E−1 (4.84E−2) –	4.645E−2 (2.71E−2) –	**1.363E−2 (5.34E−3)**
WFG3	3.930E−1 (3.62E−3) –	3.861E−1 (7.74E−4) –	3.874E−1 (1.93E−3) –	3.848E−1 (8.47E−4) –	**3.841E−1 (3.75E−4)**
WFG4	1.753E−2 (1.27E−3) –	3.279E−2 (6.38E−3) –	4.818E−2 (7.57E−3) –	1.455E−2 (1.52E−3) –	**1.369E−2 (1.13E−3)**
WFG5	6.963E−2 (5.84E−4) –	**6.649E−2 (5.96E−4)** +	6.889E−2 (3.13E−4) –	6.696E−2 (1.47E−4) –	6.650E−2 (5.98E−5)
WFG6	6.270E−2 (9.08E−3) ≈	5.528E−2 (2.80E−2) ≈	**4.527E−2 (1.18E−2)** ≈	4.702E−2 (1.12E−2) ≈	6.987E−2 (4.16E−2)
WFG7	1.795E−2 (1.70E−3) –	1.303E−2 (2.91E−4) –	1.280E−2 (1.87E−4) –	1.216E−2 (1.57E−4) ≈	**1.214E−2 (1.65E−4)**
WFG8	1.161E−1 (4.31E−3) –	**1.040E−1 (4.52E−3)** +	1.285E−1 (7.94E−3) –	1.125E−1 (4.87E−3) ≈	1.116E−1 (3.41E−3)
WFG9	9.916E−2 (4.59E−2) +	**7.401E−2 (4.86E−2)** +	1.220E−1 (1.50E−2) –	1.171E−1 (2.71E−2) –	1.137E−1 (3.18E−2)
UF1	7.872E−2 (2.12E−2) –	1.936E−1 (1.26E−1) –	8.940E−2 (1.64E−3) –	3.349E−2 (5.01E−3) –	**1.018E−2 (9.48E−4)**
UF2	3.264E−2 (7.26E−3) –	8.008E−2 (3.96E−2) –	5.547E−2 (1.28E−2) –	1.205E−2 (1.23E−3) –	**1.063E−2 (9.46E−4)**
UF3	1.588E−1 (3.41E−2) –	3.069E−1 (2.66E−2) –	3.240E−1 (1.18E−2) –	2.767E−1 (3.80E−2) –	**4.188E−2 (1.39E−2)**
UF4	4.561E−2 (7.75E−4) –	6.419E−2 (3.21E−3) –	8.230E−2 (8.42E−3) –	**4.184E−2 (1.56E−3)** +	4.410E−2 (2.96E−3)
UF5	**2.454E−1 (6.11E−2)** ≈	4.667E−1 (1.12E−1) –	4.627E−1 (1.94E−1) –	4.809E−1 (2.43E−1) –	2.839E−1 (1.07E−1)
UF6	2.377E−1 (6.86E−2) ≈	5.494E−1 (1.19E−1) –	**1.574E−1 (1.16E−1)** +	3.932E−1 (2.00E−1) ≈	3.334E−1 (1.84E−1)
UF7	1.341E−1 (1.48E−1) –	4.156E−1 (1.77E−1) –	**4.057E−2 (5.01E−4)** +	1.244E−1 (1.81E−1) –	4.937E−2 (1.51E−1)
UF8	1.905E−1 (6.03E−2) –	3.244E−1 (2.26E−1) –	1.388E−1 (3.98E−3) –	2.223E−1 (6.75E−2) –	**1.115E−1 (3.55E−2)**
UF9	2.589E−1 (8.13E−2) ≈	2.210E−1 (2.48E−2) ≈	**1.447E−1 (4.17E−2)** +	2.379E−1 (3.89E−2) ≈	2.742E−1 (8.91E−2)
UF10	5.098E−1 (1.50E−1) +	**4.612E−1 (1.22E−1)** +	7.078E−1 (2.22E−1) ≈	5.201E−1 (1.47E−1) +	6.104E−1 (1.86E−1)
DTLZ1	2.544E−2 (3.02E−3) –	2.582E−2 (3.14E−3) –	1.515E+0 (2.22E+0) –	2.754E−2 (2.56E−2) –	**2.183E−2 (5.60E−4)**
DTLZ2	6.725E−2 (2.74E−3) –	5.930E−2 (1.73E−3) –	6.078E−2 (1.53E−3) –	6.354E−2 (1.82E−3) –	**5.133E−2 (2.50E−4)**
DTLZ3	1.525E−1 (2.52E−1) +	**1.505E−1 (1.78E−1)** +	4.505E+1 (2.49E+1) –	1.929E+0 (1.61E+0) –	3.619E−1 (5.79E−1)
DTLZ4	6.181E−2 (6.24E−3) –	1.872E−1 (1.47E−1) –	6.261E−2 (3.36E−3) –	6.325E−2 (4.55E−3) –	**3.304E−2 (4.63E−4)**
DTLZ5	5.217E−3 (2.69E−4) –	3.876E−3 (8.97E−5) ≈	6.072E−3 (1.12E−3) –	**3.825E−3 (9.45E−5)** ≈	3.868E−3 (8.12E−5)
DTLZ6	1.733E−2 (1.38E−2) –	3.730E−3 (1.05E−4) ≈	1.392E−2 (2.06E−3) –	3.756E−3 (1.57E−4) –	**3.670E−3 (1.27E−4)**
DTLZ7	**7.405E−2 (3.00E−3)** +	4.100E−1 (2.49E−1) –	8.312E−2 (5.29E−3) –	7.712E−2 (4.40E−3) ≈	7.846E−2 (5.48E−3)
+/–/≈	5/22/4	5/22/4	3/26/2	2/22/7	–/–/–

"+" indicates that the peer algorithm significantly improves AgMOPSO at a 0.05 level by the Wilcoxon's rank sum test, where "–" indicates the opposite, i.e., AgMOPSO shows significant improvements over the peer algorithm. If no significant difference is detected, it will be marked by the symbol "≈". They have the same meanings in other tables.

from the local optimal only using PSO-based search, and this leads to a pretty poor performance on these test problems. Due to the use of immune-based evolutionary search, AgMOPSO-II focuses on the sparse area, which helps to keep the archive with good properties of convergence and diversity. From the above discussion, it is concluded that the two search modules compensate each other and get better performance when they are evolved cooperatively. Actually, the immune-based evolutionary search is a general evolutionary module that can also enhance the performance of other multiobjective metaheuristic algorithms, as supported by the experiments in Section II of the supplementary material.

F. Comparison of Running Times

In order to evaluate the computational efficiency of the compared algorithms, the actual running time (in *s*) on the WFG1–WFG9 test problems was recorded in Fig. 4. Please note that all the compared algorithms were implemented in Sun JAVA using a personal computer with an i7-6700 CPU running at 3.40GHz (processor) and 20.0 GB in RAM. Clearly, D^{2}MOPSO consumes significantly more time than the other competitors, as its leaders archive is updated based on the crowding distances on both decision and objective spaces. Therefore, the running time of D^{2}MOPSO is greatly lengthened as the number of variables in the WFG test problems is 24, which is significantly larger than the number of objectives. For NSGA-II, EAG-MOEA/D, MMOPSO, and AgMOPSO, they show similar running times, as they all adopt the same archive updated method, which is the main factor affecting the running time.

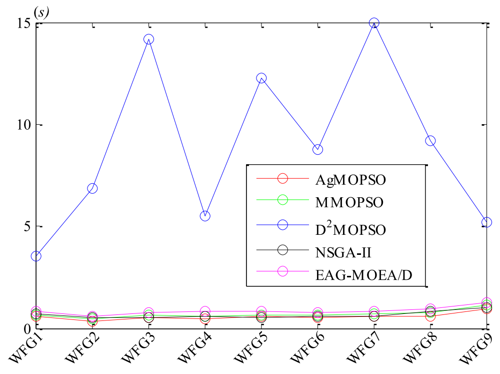


Fig. 4. Running times of all the compared algorithms on WFG test problems.

G. More Discussions About AgMOPSO

Due to page limitations, further discussions were provided in the supplementary material of this paper, in order

TABLE III
PERFORMANCE COMPARISONS OF AGMOPSO AND BCE-MOEA/D

m	Problems	BCE-MOEA/D	AgMOPSO-niche
2	ZDT1	0.91458 (1.56E–4) –	**0.91545 (2.34E–5)**
	ZDT2	0.83077 (4.75E–4) –	**0.83212 (2.66E–5)**
	ZDT3	1.19957 (1.68E–2) –	**1.20378 (2.57E–5)**
	ZDT4	**0.91047 (2.71E–3)** +	0.90751 (2.31E–2)
	ZDT6	0.75305 (1.10E–3) –	**0.76048 (2.15E–6)**
	WFG1	0.45826 (2.96E–2) –	**0.70446 (1.54E–2)**
	WFG2	0.70700 (9.96E–3) –	**0.76172 (3.22E–3)**
	WFG3	0.72582 (9.25E–4) –	**0.72955 (2.03E–4)**
	WFG4	0.57487 (8.60E–4) –	**0.57719 (4.09E–4)**
	WFG5	0.54157 (3.95E–4) –	**0.54729 (2.72E–3)**
	WFG6	0.54877 (6.22E–3) –	**0.56085 (1.49E–2)**
	WFG7	0.57644 (6.29E–4) –	**0.57866 (6.77E–5)**
	WFG8	0.51485 (1.77E–3) –	**0.53656 (1.51E–3)**
	WFG9	**0.53712 (1.82E–2)** ≈	0.51911 (1.11E–2)
	UF1	0.85045 (2.78E–2) –	**0.91238 (1.02E–3)**
	UF2	0.88492 (1.74E–2) –	**0.91128 (2.08E–3)**
	UF3	0.67227 (3.42E–2) –	**0.90348 (6.38E–3)**
	UF4	**0.80477 (1.72E–3)** ≈	0.80415 (1.37E–3)
	UF5	0.62667 (8.32E–2) ≈	**0.64143 (9.17E–2)**
	UF6	**0.64205 (6.96E–2)** ≈	0.61528 (9.89E–2)
	UF7	0.71026 (1.13E–1) –	**0.86556 (5.83E–3)**
3	UF8	0.86635 (4.95E–2) –	**0.88000 (4.63E–2)**
	UF9	**0.89719 (3.45E–2)** +	0.87667 (3.05E–2)
	UF10	**0.60713 (1.39E–1)** ≈	0.56462 (1.46E–1)
	DTLZ1	**0.97381 (2.85E–4)** +	0.97380 (6.49E–5)
	DTLZ2	0.92635 (7.35E–4) –	**0.92678 (1.56E–4)**
	DTLZ3	**0.88568 (1.72E–1)** ≈	0.64348 (3.42E–1)
	DTLZ4	0.92609 (8.45E–4) –	**0.92689 (1.31E–4)**
	DTLZ5	0.76271 (2.18E–4) –	**0.76288 (2.75E–5)**
	DTLZ6	0.73989 (8.19E–3) –	**0.76297 (1.01E–5)**
	DTLZ7	0.47884 (3.31E–3) –	**0.48081 (6.04E–4)**
5	DTLZ1	0.00740 (4.05E–2) –	**0.97302 (1.20E–3)**
	DTLZ2	**0.80205 (3.40E–3)** +	0.76888 (3.71E–3)
	DTLZ3	0.00000 (0.00E+0) ≈	**0.03290 (1.25E–1)**
	DTLZ4	**0.77906 (1.06E–2)** ≈	0.77896 (3.57E–3)
	DTLZ5	0.19217 (2.94E–2) –	**0.21185 (8.69E–3)**
	DTLZ6	0.05154 (3.62E–2) –	**0.22608 (9.22E–3)**
	DTLZ7	0.70153 (3.31E–3) –	**0.70378 (1.72E–3)**
	WFG1	0.36437 (1.30E–2) –	**0.47102 (3.48E–2)**
	WFG2	0.93324 (8.53E–2) –	**0.98602 (2.97E–3)**
	WFG3	0.58887 (2.78E–2) –	**0.60791 (1.88E–2)**
	WFG4	0.72162 (1.18E–2) –	**0.75320 (9.51E–3)**
	WFG5	**0.71482 (5.39E–3)** +	0.70759 (8.07E–3)
	WFG6	0.68672 (2.07E–2) –	**0.70225 (4.53E–3)**
	WFG7	0.74102 (1.36E–2) –	**0.77542 (4.69E–3)**
	WFG8	0.58781 (1.52E–2) –	**0.63144 (8.64E–3)**
	WFG9	0.60360 (1.15E–2) ≈	**0.60888 (1.55E–2)**
10	DTLZ1	0.21092 (3.97E–1) –	**0.82372 (2.00E–1)**
	DTLZ2	1.00000 (3.12E–8) –	**1.0000 (4.72E–15)**
	DTLZ3	**0.21042 (3.93E–1)** +	0.0000 (0.00E+0)
	DTLZ4	0.99996 (1.58E–4) –	**1.0000 (2.55E–15)**
	DTLZ5	0.95551 (3.53E–2) –	**0.99722 (5.46E–4)**
	DTLZ6	0.90823 (8.48E-2) –	**0.99687 (8.12E-4)**
	DTLZ7	0.59510 (9.04E-3) –	**0.61304 (2.23E-2)**
	WFG1	0.30804 (1.14E-2) –	**0.46823 (3.76E-2)**
	WFG2	**0.98096 (3.72E-2)** +	0.97673 (1.36E-2)
	WFG3	0.45915 (9.84E-2) –	**0.56517 (3.05E-2)**
	WFG4	**0.77387 (2.44E-2)** +	0.74334 (4.48E-2)
	WFG5	**0.79392 (1.42E-2)** +	0.60060 (2.86E-2)
	WFG6	0.68492 (3.33E-2) –	**0.82041 (3.98E-3)**
	WFG7	0.85337 (2.12E-2) –	**0.88458 (1.43E-2)**
	WFG8	**0.66419 (3.58E-2)** +	0.61088 (3.38E-2)
	WFG9	0.52131 (5.40E-2) ≈	**0.53882 (6.14E-2)**
+/–/≈		10/43/10	–/–/–

TABLE IV
PERFORMANCE COMPARISONS OF AGMOPSO AND TWO VARIANTS

Problems	AgMOPSO-I	AgMOPSO-II	AgMOPSO
ZDT1	6.47E–1 (2.5E–1) –	4.01E–3 (2.9E–4) –	**3.70E–3 (2.8E–5)**
ZDT2	2.22E+0 (7.4E–1) –	4.05E–3 (1.6E–4) –	**3.82E–3 (3.1E–5)**
ZDT3	5.52E–1 (1.9E–1) –	4.41E–3 (4.6E–5) –	**4.36E–3 (5.2E–5)**
ZDT4	3.60E+1 (1.2E+1) –	1.04E–2 (2.3E–2) –	**7.94E–3 (2.2E–2)**
ZDT6	6.69E–1 (1.7E+0) –	3.00E–3 (1.0E–4) ≈	**2.99E–3 (9.5E–5)**
WFG1	1.23E+0 (1.2E–2) –	4.96E–1 (1.4E–1) –	**3.26E–1 (6.3E–2)**
WFG2	2.92E–1 (6.3E–2) –	9.81E–2 (4.4E–2) –	**1.36E–2 (5.3E–3)**
WFG3	5.51E–1 (9.0E–2) –	3.97E–1 (1.0E–2) –	**3.84E–1 (3.7E–4)**
WFG4	1.22E–1 (1.3E–2) –	**1.12E–2 (2.8E–4)** +	1.36E–2 (1.1E–3)
WFG5	6.76E–2 (2.0E–3) –	**6.64E–2 (3.8E–4)** ≈	6.65E–2 (5.9E–5)
WFG6	1.32E–1 (1.5E–1) –	**6.04E–2 (1.3E–2)** ≈	6.98E–2 (4.1E–2)
WFG7	1.30E–1 (2.5E–2) –	1.22E–2 (2.0E–4) ≈	**1.21E–2 (1.6E–4)**
WFG8	3.16E–1 (2.8E–2) –	1.14E–1 (5.2E–3) –	**1.11E–1 (3.4E–3)**
WFG9	1.22E–1 (8.0E–3) ≈	**1.04E–1 (4.0E–2)** +	1.13E–1 (3.1E–2)
UF1	3.03E–1 (9.1E–2) –	6.37E–2 (4.2E–2) –	**1.01E–2 (9.4E–4)**
UF2	1.29E–1 (1.9E–2) –	2.82E–2 (1.5E–2) –	**1.06E–2 (9.4E–4)**
UF3	5.61E–1 (4.0E–2) –	1.85E–1 (5.1E–2) –	**4.18E–2 (1.3E–2)**
UF4	8.20E–2 (1.0E–2) –	4.97E–2 (3.7E–3) –	**4.41E–2 (2.9E–3)**
UF5	3.13E+0 (4.5E–1) –	**2.80E–1 (1.1E–1)** ≈	2.83E–1 (1.0E–1)
UF6	1.25E+0 (2.4E–1) –	4.43E–1 (1.8E–1) –	**3.33E–1 (1.8E–1)**
UF7	3.82E–1 (1.9E–1) –	1.58E–1 (1.9E–1) –	**4.93E–2 (1.5E–1)**
UF8	4.42E–1 (1.4E–1) –	1.97E–1 (4.5E–2) –	**1.11E–1 (3.5E–2)**
UF9	5.50E–1 (9.5E–2) –	3.22E–1 (7.1E–2) –	**2.74E–1 (8.9E–2)**
UF10	4.47E+0 (1.0E+0) –	7.67E–1 (1.8E–1) –	**6.10E–1 (1.8E–1)**
DTLZ1	1.23E+1 (6.9E+0) –	2.62E–2 (2.6E–3) –	**2.18E–2 (5.6E–4)**
DTLZ2	6.45E–2 (2.0E–3) –	6.78E–2 (3.7E–3) –	**5.13E–2 (2.5E–4)**
DTLZ3	8.24E+1 (4.0E+1) –	**1.12E–1 (1.7E–1)** +	3.61E–1 (5.7E–1)
DTLZ4	7.50E–2 (2.3E–2) –	7.44E–2 (6.5E–2) –	**3.30E–2 (4.6E–4)**
DTLZ5	5.11E–3 (1.4E–3) –	3.89E–3 (9.6E–5) ≈	**3.86E–3 (8.1E–5)**
DTLZ6	4.42E–1 (6.7E–1) –	7.48E–2 (3.3E–2) –	**3.67E–3 (1.2E–4)**
DTLZ7	1.10E+0 (8.8E–1) –	7.99E–2 (4.8E–3) ≈	**7.84E–2 (5.4E–3)**
+/–/≈	0/30/1	3/21/7	–/–/–

to further study the performance of AgMOPSO on many-objective optimization problems, the effectiveness of the immune-based evolutionary search and the cloning operator, and the parameter sensitivity analysis of AgMOPSO on T, w, and F.

V. HANDLING CONSTRAINTS

After demonstrating the superiority of AgMOPSO for solving unconstrained MOPs (i.e., only with box constraints on decision variables), this section extends AgMOPSO (denoted as C-AgMOPSO) to solve constrained MOPs.

In case of the presence of infeasible solutions, some modifications are suggested to the archive update procedure of AgMOPSO, which is aimed to give more emphasis on feasible solutions. The other components of AgMOPSO keep untouched, as introduced in **Algorithm 5**. An individual with a lower constraint violation value is considered first and the population diversity should be maintained at the same time.

A. Modifications on the Archive Update Procedure

As suggested in [61], the constraint violation value of a solution x, denoted as $\mathrm{CV}(x)$, is calculated by the following form:

$$\mathrm{CV}(x) = \sum_{j=1}^{J}\langle g_j(x)\rangle + \sum_{k=1}^{K}\langle h_k(x)\rangle \tag{7}$$

where the bracket operator $\langle\alpha\rangle$ returns the absolute value of α if $\alpha < 0$, and returns 0 otherwise. It is obvious that a smaller value of $\mathrm{CV}(x)$ indicates the better quality of x, and a feasible solution x always has a $\mathrm{CV}(x)$ value as 0.

TABLE V
COMPARISON RESULTS ON CONSTRAINED TEST PROBLEMS

Problems	C-MOEA/DD	C-AgMOPSO
CF1	0.84031 (4.25E–3) –	**0.86803 (1.46E–4)**
CF2	0.60237 (1.34E–1) –	**0.90273 (7.92E–5)**
CF3	0.49492 (5.27E–2) –	**0.59908 (1.06E–1)**
CF4	0.71891 (5.26E–2) –	**0.78686 (6.98E–2)**
CF5	0.54987 (4.39E–2) –	**0.61858 (4.46E–2)**
CF6	0.83861 (3.19E–2) –	**0.88888 (1.76E–2)**
CF7	0.63973 (1.04E–1) –	**0.70051 (5.19E–2)**
CF8	0.82826 (4.62E–1) –	**0.85325 (1.63E–1)**
CF9	0.90171 (4.89E–3) –	**0.91671 (2.99E–3)**
CF10	**0.43362 (8.12E–2)** +	0.17342 (3.23E–1)
C1-DTLZ1	0.97030 (2.84E–3) –	**0.97231 (3.07E–3)**
C2-DTLZ2	0.91883 (1.17E–3) –	**0.91979 (2.86E–4)**
C3-DTLZ1	**0.97621 (2.20E–3)** ≈	0.97608 (1.83E–3)
C3-DTLZ4	**0.91971 (1.34E–4)** +	0.91839 (1.24E–4)
+/–/≈	2/11/1	–/–/–

The pseudo-code of this modified archive update procedure is given in **Algorithm 6**. At first, the archive A and the offspring population S are combined (in line 1). Then, the feasible and infeasible solutions in the union population are identified (in lines 3–9). If the number of feasible solutions is larger than N, the update procedure is the same as **Algorithm 4** (in line 11). Otherwise, the feasible solutions are added to the archive first (in lines 13–15) and then the infeasible solutions are sorted in descending order according to the CV values using (7). The corresponding subproblem k nearest to the individual with the lowest CV value is found. If this subproblem k is not marked, this solution is added to the archive and the subproblem k is also marked. This procedure will go on until the archive size reaches N (in lines 17–25). Please note that when evaluating the quality of infeasible solutions, its constraint violation value and the population diversity in the archive are simultaneously considered in **Algorithm 6**.

B. Experiments

In order to validate the performance of C-AgMOPSO in solving constrained MOPs, C-AgMOPSO was further compared to C-MOEA/DD [11] on tackling 14 benchmark constrained MOPs, including ten CF problems proposed in the CEC 2009 test suite [55] and four constrained test instances in [61] (C1-DTLZ1, C2-DTLZ2, C3-DTLZ1, and C3-DTLZ4). The population size was set to 100 for the bi-objective test problems and to 105 for the three-objective ones. The maximum numbers of generation were set to 3000 for the CF problems and to 500 for the constrained DTLZ problems. The reference points for HV are set to 1.1 times the nadir points of the true POFs, i.e., $1.1 \times (0.5, \ldots, 0.5)$ for C1_DTLZ1, $1.1 \times (1.0, \ldots, 1.0)$ for CF1-CF10, C2_DTLZ2, C3_DTLZ1, and $1.1 \times (2.0, \ldots, 2.0)$ for C3-DTLZ4. Other parameters settings were set the same as introduced in Section IV-C.

1) Comparison of C-AgMOPSO and C-MOEA/DD: Table V shows the HV results obtained from 30 independent runs on 14 test instances. The better results were marked with boldface. These statistical results were obtained based on feasible nondominated solutions that dominate the reference point for each problem. As observed from Table V, C-AgMOPSO significantly outperforms C-MOEA/DD on two-objective constrained test instances (CF1–CF7). Moreover, for the three-objective constrained test instances, C-MOEA/DD performs better only on CF10 and C3-DTLZ4, while C-AgMOPSO performs better on CF8, CF9, C1-DTLZ1, and C2-DTLZ2. For C3-DTLZ1, C-AgMOPSO, and C-MOEA/DD show statistically similar performance. From the last row in Table V, C-AgMOPSO performs better than C-MOEA/DD on 10 out of 14 test instances. Therefore, it is concluded that C-AgMOPSO presents some advantages over C-MOEA/DD in solving these constrained test problems.

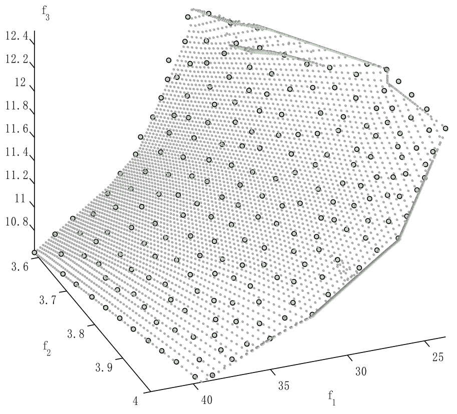


Fig. 5. Final results of C-AgMOPSO on the car-side impact problem.

To visually show the performance, the best final solution sets obtained by C-MOEA/DD and C-AgMOPSO on CF1–CF6, and C1-DTLZ1, C2-DTLZ2, C3-DTLZ1, and C3-DTLZ4 were plotted in Fig. S-1 of the supplementary material.

2) Further Study on Tackling Real World Problem: Here, a real world engineering problem (car side-impact problem [61]) was also included to validate the performance of AgMOPSO. This problem has three objectives and ten constraints, which aims at minimizing the weight of a car and simultaneously minimizing the public force experienced by a passenger and the average velocity of the V-Pillar responsibility for withstanding the impact load. More details about this problem can be found in [61].

For this problem, the population size N was set to 210 and the maximum number of generation was set to 2000. All other experimental configurations were set the same as introduced in Section III-C. Fig. 5 illustrates the final solutions obtained by C-AgMOPSO, where the generated approximated POF using the classical generative procedure (i.e., the fmincon function in MATLAB) is marked with small circles while the approximated POF obtained by C-AgMOPSO is identified with bigger circles. Apparently, as observed from Fig. 5, all the points found by C-AgMOPSO are uniformly distributed over the entire surface formed by the classical generative procedure. To investigate the closeness of our found solutions with that obtained by the classical generative procedure, the convergence metric (i.e., the average distance from our points to that found by the classical generative procedure) is computed and its value is 3.59×10^{-3}. This value is very small and it clearly indicates that the solutions obtained by C-AgMOPSO can closely approach the approximated POF of this problem. Moreover, the spread of solutions is also demonstrated visually in Fig. 5.

VI. Conclusion

In this paper, a novel MOPSO algorithm with an archive-guided velocity update method was presented, which is based on a decomposition approach to transform MOPs into a set of aggregated subproblems. The *p*best, *l*best, and *g*best particles are all properly selected from the external archive. Additionally, an immune-based evolutionary strategy is further applied on some individuals that are selected from the external archive for being located in sparse areas of the search space. The evolution on the external archive was verified to promote the convergence speed and keep the diversity, which can help to guide the swarm to do the PSO-based search. In this way, the performance of AgMOPSO was enhanced to enable it tackle various types of MOPs. The effectiveness of the proposed immune-based search and archive-guided PSO search approaches was also justified by experimental results. When compared to three state-of-the-art MOEAs and two competitive MOPSOs, our experimental results confirmed that AgMOPSO showed a competitive performance in solving most of the test problems adopted. Moreover, the extensions of AgMOPSO to solve constrained optimization problems, many-objective optimization problems, and a real world engineering problem were also conducted in this paper to show its potential in tackling different types of optimization problems.

As part of our future work, the performance of AgMOPSO in tackling many-objective optimization problems will be further studied and the applications of AgMOPSO for more practical problems will also be studied.

References

[1] X. Zhang, Y. Zhou, Q. Zhang, V. C. S. Lee, and M. Li, "Problem specific MOEA/D for barrier coverage with wireless sensors," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2585745.

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

[3] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. Evol. Methods Design Optim. Control Appl. Ind. Problems (EUROGEN)*, 2002, pp. 95–100.

[4] 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.

[5] E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Int. Conf. Parallel Problem Solving Nat. (PPSN)*, 2004, pp. 832–842.

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

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

[8] K. Li, S. Kwong, Q. Zhang, and K. Deb, "Interrelationship-based selection for decomposition multiobjective optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2076–2088, Oct. 2015.

[9] Q. Lin, J. Li, Z. Du, J. Chen, and Z. Ming, "A novel multi-objective particle swarm optimization with multiple search strategies," *Eur. J. Oper. Res.*, vol. 247, no. 3, pp. 732–744, Dec. 2015.

[10] K. Li, S. Kwong, and K. Deb, "A dual-population paradigm for evolutionary multiobjective optimization," *Inf. Sci.*, vol. 309, pp. 50–72, Jul. 2015.

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

[12] M. Li, S. Yang, and X. Liu, "Pareto or non-Pareto: Bi-criterion evolution in multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 645–665, Oct. 2016, doi: 10.1109/TEVC.2015.2504730.

[13] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 21, no. 3, pp. 440–462, Jun. 2017, doi: 10.1109/TEVC.2016.2608507.

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

[15] J. Li, J. Q. Zhang, C. J. Jiang, and M. C. Zhou, "Composite particle swarm optimizer with historical memory for function optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2350–2363, Oct. 2015.

[16] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle swarm optimization with interswarm interactive learning strategy," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2238–2251, Oct. 2016, doi: 10.1109/TCYB.2015.2474153.

[17] W. Hu and G. G. Yen, "Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system," *IEEE Trans. Evol. Comput.*, vol. 19, no. 1, pp. 1–18, Feb. 2015.

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

[19] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 5, no. 6, pp. 565–588, Dec. 2001.

[20] N. A. Moubayed, A. Petrovski, and J. McCall, "D^{2}MOPSO: MOPSO based on decomposition and dominance with archiving using crowding distance in objective and solution spaces," *Evol. Comput.*, vol. 22, no. 1, pp. 44–77, May 2014.

[21] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2202–2215, Dec. 2013.

[22] Y.-L. Li *et al.*, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.

[23] P. C. Roy, M. M. Islam, K. Murase, and X. Yao, "Evolutionary path control strategy for solving many-objective optimization problem," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 702–715, Apr. 2015.

[24] X. Qiu, J.-X. Xu, K. C. Tan, and H. A. Abbass, "Adaptive cross-generation differential evolution operators for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 232–244, Apr. 2016, doi: 10.1109/TEVC.2015.2433672.

[25] Q. Z. Lin *et al.*, "A hybrid evolutionary immune algorithm for multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 711–729, Oct. 2016, doi: 10.1109/TEVC.2015.2512930.

[26] R. H. Shang, L. C. Jiao, F. Liu, and W. P. Ma, "A novel immune clonal algorithm for MO problems," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 35–50, Feb. 2012.

[27] C. A. C. Coello and C. N. Cruz, "An approach to solve multiobjective optimization problems based on an artificial immune system," in *Proc. 1st Conf. Artif. Immune Syst.*, 2002, pp. 212–221.

[28] L. C. Jiao, M. G. Gong, R. H. Shang, H. F. Du, and B. Lu, "Clonal selection with immune dominance and energy based multiobjective optimization," in *Proc. 3rd Conf. Evol. Multi Criterion Optim.*, vol. 3410. 2005, pp. 474–489.

[29] R. H. Shang, L. C. Jiao, M. G. Gong, and B. Lu, "Clonal selection algorithm for dynamic multiobjective optimization," in *Proc. Conf. Comput. Intell. Security*, 2005, pp. 846–851.

[30] M. G. Gong, L. C. Jiao, H. F. Du, and L. F. Bo, "Multiobjective immune algorithm with nondominated neighbor-based selection," *Evol. Comput.*, vol. 16, no. 2, pp. 225–255, 2008.

[31] K. C. Tan, C. K. Goh, A. A. Mamun, and E. Z. Ei, "An evolutionary artificial immune system for multi-objective optimization," *Eur. J. Oper. Res.*, vol. 187, no. 2, pp. 371–392, 2008.

[32] J. Y. Chen, Q. Z. Lin, and Z. Ji, "A hybrid immune multiobjective optimization algorithm," *Eur. J. Oper. Res.*, vol. 204, no. 2, pp. 294–302, 2010.

[33] Q. Z. Lin and J. Y. Chen, "A novel micro-population immune multi-objective optimization algorithm," *Comput. Oper. Res.*, vol. 40, no. 6, pp. 1590–1601, 2013.

[34] Q. Lin *et al.*, "A novel hybrid multi-objective immune algorithm with adaptive differential evolution," *Comput. Oper. Res.*, vol. 65, pp. 95–111, Oct. 2015.

[35] Y. T. Qi, Z. T. Hou, M. L. Yin, H. L. Sun, and J. B. Huang, "An immune multi-objective optimization algorithm with differential evolution inspired recombination," *Appl. Soft Comput.*, vol. 29, pp. 395–410, Apr. 2015.

[36] M. R. Sierra, and C. A. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and epsilon-dominance," in *Proc. Evol. Multi Criterion Optim.*, vol. 3410. 2005, pp. 505–519.

[37] A. J. Nebro *et al.*, "SMPSO: A new PSO-based metaheuristic for multi-objective optimization," in *Proc. IEEE Symp. Comput. Intell. Multi Criteria Decis. Making*, Nashville, TN, USA, 2009, pp. 66–73.

[38] 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, 2006.

[39] S.-Z. Zhao and P. N. Suganthan, "Two-lbests based multi-objective particle swarm optimizer," *Eng. Optim.*, vol. 43, no. 1, pp. 1–17, 2011.

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

[41] W. Peng and Q. Zhang, "A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems," in *Proc. Conf. Granular Comput.*, 2008, pp. 534–537.

[42] N. A. Moubayed, A. Petrovski, and J. A. W. McCall, "A novel smart multi-objective particle swarm optimisation using decomposition," in *Proc. PPSN*, 2010, pp. 1–10.

[43] S. Z. Martinez and C. A. C. Coello, "A multi-objective particle swarm optimizer based on decomposition," in *Proc. Genet. Evol. Comput.*, Dublin, Ireland, 2011, pp. 69–76.

[44] N. A. Moubayed, A. Petrovski, and J. McCall, "D^{2}MOPSO: Multi-objective particle swarm optimizer based on decomposition and dominance," in *Proc. Eur. Conf. (EvoCOP)*, vol. 7245. 2012, pp. 75–86.

[45] L. T. Bui *et al.*, "DMEA: A direction-based multiobjective evolutionary algorithm," *Memetic Comput.*, vol. 3, no. 4, pp. 271–285, 2011.

[46] L. Nguyen, L. T. Bui, and H. A. Abbass, "DMEA-II: The direction-based multi-objective evolutionary algorithm-II," *Soft Comput.*, vol. 18, no. 11, pp. 2119–2134, 2014.

[47] L. Nguyen, L. T. Bui, and H. Abbass, "A new niching method for the direction-based multi-objective evolutionary algorithm," in *Proc. IEEE Symp. Comput. Intell. Multi Criteria Decis. Making (MCDM)*, Apr. 2013, pp. 1–8.

[48] H. Li and D. Landa-Silva, "Evolutionary multi-objective simulated annealing with adaptive and competitive search direction," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2008, pp. 3311–3318.

[49] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, 1998.

[50] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 115–148, 1995.

[51] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[52] S. Kukkonen and K. Deb, "Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems," in *Proc. IEEE Int. Conf. Evol. Comput.*, Vancouver, BC, Canada, Jul. 2006, pp. 1179–1186.

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

[54] 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.

[55] Q. F. Zhang *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," School Comput. Sci. Electron. Eng., Univ. Essex, Colchester, U.K., Tech. Rep. CES-487, 2009.

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

[57] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 174–188, Apr. 2003.

[58] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 86–95, Feb. 2012.

[59] X. Cai, Y. Li, Z. Fan, and Q. Zhang, "An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 508–523, Aug. 2015.

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

[61] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

Qingling Zhu received the B.Sc. degree from the Nanchang Institution of Technology, Nanchang, China, in 2013, the M.Sc. degree from Shenzhen University, Shenzhen, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science, City University of Hong Kong, Hong Kong.

He is currently a Research Assistant with Shenzhen University. His current research interests include evolutionary multiobjective optimization and machine learning.

Qiuzhen Lin received the B.S. degree from Zhaoqing University, Zhaoqing, China, in 2007, the M.S. degree from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree from the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, in 2014.

He is currently a Lecturer with the College of Computer Science and Software Engineering, Shenzhen University. He has published over ten research papers since 2008. His current research interests include artificial immune system, multiobjective optimization, and dynamic system.

Weineng Chen (S'07–M'12) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has published over 70 papers in international journals and conferences, including over 20 papers in IEEE TRANSACTIONS journals. His current research interest includes swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society Outstanding Dissertation Award in 2016 for his doctoral thesis, and the National Science Fund for Excellent Young Scholars in 2016.

Ka-Chun Wong received the B.Eng. degree in computer engineering and the M.Phil. degree from the Chinese University of Hong Kong, Hong Kong, in 2008 and 2010, respectively, and the Ph.D. degree from the Department of Computer Science, University of Toronto, Toronto, ON, Canada, in 2015.

He was an Assistant Professor with the City University of Hong Kong. His current research interests include bioinformatics, computational biology, evolutionary computation, data mining, machine learning, and interdisciplinary research.

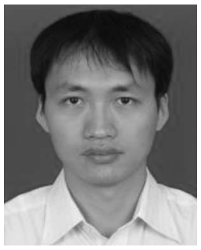
Carlos A. Coello Coello (M'98–SM'04–F'11) received the Ph.D. degree in computer science from Tulane University, New Orleans, LA, USA, in 1996.

He is currently a Professor (CINVESTAV-3F Researcher) with the Computer Science Department, CINVESTAV-IPN, Mexico City, Mexico. He has authored and co-authored over 450 technical papers and book chapters. He has also co-authored the book entitled *Evolutionary Algorithms for Solving Multi-Objective Problems* (Second Edition, Springer, 2007). His has over 34 000 Google Scholar citations with an H-index of 76. His current research interests include evolutionary multiobjective optimization and constraint-handling techniques for evolutionary algorithms.

Dr. Coello Coello was a recipient of the 2007 National Research Award from the Mexican Academy of Sciences in the area of *Exact Sciences*, the 2013 IEEE Kiyo Tomiyasu Award and the 2012 National Medal of Science and Arts in the area of Physical, Mathematical and Natural Sciences. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and serves in the editorial board of 12 other international journals. He is a member of the ACM and the Mexican Academy of Science.

Jianqiang Li received the B.S and Ph.D. degrees from the South China University of Technology, Guangzhou, China, in 2003 and 2008, respectively.

He is an Associate Professor with the College of Computer and Software Engineering, Shenzhen University, Shenzhen, China. He led a project of the National Natural Science Foundation, and a project of the Natural Science Foundation of Guangdong Province, China. His current research interests include robotic, hybrid systems, Internet of Things, and embedded systems.



Jianyong Chen (M'11) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2003.

He is a Professor with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. He was with ZTE Corporation, Shenzhen, as a Senior Engineer of network technology from 2003 to 2006. Then he joined Shenzhen University. He has published over 30 papers and got over 30 patents in the field of artificial intelligence and information security. His current research interests include in artificial intelligence and information security.

Dr. Chen was the Vice-Chairman of International Telecommunication Union-Telecommunication SG17 from 2004 to 2012, and an Editor of three recommendations developed in ITU-T SG17.

Jun Zhang (M'02–SM'08–F'17) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

From 2004 to 2016, he was a Professor with Sun Yat-sen University, Guangzhou, China. Since 2016, he has been with the South China University of Technology, Guangzhou, where he is currently a Cheung Kong Chair Professor. His current research interests include computational intelligence, cloud computing, big data, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 100 technical papers in the above areas.

Prof. Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and the Current Chair of the IEEE Guangzhou Subsection, the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters, and ACM Guangzhou Chapter.