

A multi-objective artificial bee colony algorithm based on division of the searching space

Yu-Bin Zhong · Yi Xiang · Hai-Lin Liu

Published online: 19 June 2014
© Springer Science+Business Media New York 2014

Abstract This paper presents a new multi-objective artificial bee colony algorithm called dMOABC by dividing the whole searching space S into two independent parts S_1 and S_2 . In this algorithm, two "basic" colonies are assigned to search potential solutions in regions S_1 and S_2 , while the so-called "synthetic" colony explores in S . This multi-colony model could enable the good diversity of the population, and three colonies share information in a special way. A fixed-size external archive is used to store the non-dominated solutions found so far. The diversity over the archived solutions is controlled by utilizing a self-adaptive grid. For basic colonies, neighbor information is used to generate new food sources. For the synthetic colony, besides neighbor information, the global best food source g_{best} selected from the archive, is also adopted to guide the flying trajectory of both employed and onlooker bees. The scout bees are used to get

rid of food sources with poor qualities. The proposed algorithm is evaluated on a set of unconstrained multi-objective test problems taken from CEC09, and is compared with 11 other state-of-the-art multi-objective algorithms by applying Friedman test in terms of four indicators: HV, SPREAD, EPSILON and IGD. It is shown by the test results that our algorithm significantly surpasses its competitors.

Keywords Multi-objective optimization · Pareto dominance · Multi-objective artificial bee colony algorithm · Multi-colony model · Friedman test

1 Introduction

In optimization field, scientists and engineers are usually faced with problems with more than one objective function, which are called multi-objective problems (MOPs). Compared to single objective problems, MOPs are more difficult to be solved since no single solution is available for these problems. Quite often, objectives in MOPs are conflicting with each other, and the performance of each objective can't be improved without sacrificing the performance of at least one of the others. Hence, the goal for settling MOPs is to find a set of solutions that represent a trade-off among the objectives [3].

Recently, evolutionary and swarm-based search methods have been widely used to solve problems with multiple objectives, and they can be classified into different categories such as aggregative, lexicographic, sub-population, Pareto-based, and hybrid methods [43, 56]. Among the multi-objective methods, the majority of research is concentrated on Pareto-based approaches [20]. Pareto-based methods select a part of individuals based on the Pareto dominance notion as leaders (or non-dominated solutions)

This paper is supported by National Natural Science Foundation of China under Grant 61350003 and the Project of Department of Education of Guangdong Province (No.20131130543031) and by the major Research Project of Guangdong Baiyun University (No. BYKY201317).

Y.-B. Zhong
School of Mathematics and Information Sciences & Key
Laboratory of Mathematics and Interdisciplinary Sciences of
Guangdong Higher Education Institutes, Guangzhou University,
Guangzhou 510006, China
e-mail: zhong_yb@163.com

Y. Xiang (✉)
Department of Advanced Mathematics, Guangdong Baiyun
University, Guangzhou 510450, People's Republic of China
e-mail: gzhuxiang_yi@163.com

H.-L. Liu (✉)
Department of Applied Mathematics, Guangdong University
of Technology, Guangzhou 510090, People's Republic of China
e-mail: gzhumath@163.com

which constitute the non-dominated set of the current iteration. These methods use different approaches to continually update the non-dominated set till the algorithms are terminated. Usually leaders are maintained in an external archive.

Many evolutionary algorithms have been extended to deal with multi-objective problems in recent years. The main issues of multi-objective evolutionary algorithms (MOEAs) are selecting and updating individuals using evolutionary operations. In 1985, Schaffer proposed the first multi-objective evolutionary algorithm named Vector Evaluated Genetic Algorithm (VEGA) [41]. After that, some other MOEAs were available, such as MOGA [17], NPGA [22] and NSGA [44]. During the time from 1994 to 2003, MOEAs got rapid development and many new methods were proposed by introducing the external archive. Typical MOEAs in this phase were: SPEA [59], NSGA [11], PAES [27], PESA [9] and SPEA2 [58], etc. From the year 2003 until now, the study on MOEAs went into a new phase, with import of various new notions, mechanisms and strategies. During this time period, Indicator-Based Evolutionary Algorithm (IBEA) [57], MOCcell [36], cooperative multi-objective evolutionary algorithms such as DCCEA [45] and CO-MOEA [8], and dynamic MOEAs such as DMOEA [51], and parallel MOEAs as PSFGA [38] were reported in corresponding literatures. Meanwhile, Multi-Objective Evolutionary Algorithm based on Division (MOEA/D) [52], Generalized Differential Evolution 3 (GDE3) [29], Enhancing MOEA/D with Guided Mutation and Priority Update (MOEA/DGM) [6], improved MOEA/D with adaptive weight vector adjustment (MOEA/D-AWA) [40], Multiple Trajectory Search (MTS) [48], LiuLiAlgorithm [33], an improved version of Dynamical Multi-Objective Evolutionary Algorithm (DMOEA-DD) [34] and Archive-based Micro Genetic Algorithm (AMGA) [46] are some other competitive methods aimed to obtain a true Pareto front for multi-objective problems by considering the main issues of multi-objective evolutionary algorithms.

Several types of swarm-based algorithms for MOPs have been presented in the literature. Particle swarm optimization (PSO) algorithm, proposed by Kennedy and Eberhart [16, 26], was modified in recent years so as to deal with optimization problems with more than one objective. Some typical kinds of PSO-based multi-objective algorithms were available, such as Multi-Objective Particle Swarm Optimization (MOPSO) [7], OMOPSO [42], Multi-objective Algorithm based on Comprehensive Learning Particle Swarm Optimizer (MOCLPSO) [23], Time Variant Multi-Objective Particle Swarm Optimization (TV-MOPSO) [47], Interactive Particle Swarm Optimization (IPSO) [1], PSO-Based Multi-Objective Optimization With Dynamic Population Size and Adaptive Local Archives (DMOPSO)

[31], Dynamic Multiple Swarms in Multi-Objective Particle Swarm Optimization (DSMOPSO) [50], Speed-constrained Multi-objective PSO (SMPSO) [35], Competitive and Cooperative co-evolutionary Multi-objective Particle Swarm Optimization (CCPSO) [19], Two Local Bests (*lbest*) based MOPSO (2LB-MOPSO) [54], Pareto-based particle swarm optimization [55], etc.

The Artificial bee colony (ABC) algorithm is one of the most recently introduced swarm-based methods [2, 25]. In the ABC, there are three kinds of bees who do different tasks to make the algorithm useful. The employed bees will be sent to food sources and try to improve them by using neighbor information. The onlookers will choose one of those food sources based on the quality of food sources shared by employed bees, and then try to improve it. At last, scout bees will find the food sources which have not been optimized in a limited number of cycles so as to reinitialize them to get rid of poor solutions. The ABC seems particularly suitable for multi-objective optimization mainly because of solution quality and the high speed of convergence that the algorithm presents for single-objective optimization [3].

However, only a few works that extend ABC for handling multi-objective problems were reported in recent years. Hedayatzadeh et al. proposed a multi-objective artificial bee colony (MOABC) which has adapted the basic ABC algorithm and a grid-based approach for maintaining and adaptively assessing the Pareto front [21]. Omkar et al. presented a method called the Vector-Evaluated ABC (or VEABC) for the multi-objective design of composite structures [39]. In the VEABC, multiple populations were used to concurrently optimize problem at hand. Also, a multi-objective variant of the ABC was used by Atashkari et al. for the optimization of power and heating system [5]. A Pareto-based discrete artificial bee colony algorithm was used by Li et al. for solving multi-objective flexible job shop scheduling problems in which a crossover operator was developed for the employed bees and an external Pareto archive set was designed to record the non-dominated solutions found so far, besides, several local search approaches were designed to balance the exploration and exploitation capability of the algorithm [32]. A multi-objective ABC was used in [4] for scheduling in grid environment. Zou et al. proposed a novel algorithm based on Artificial Bee Colony (ABC) to deal with multi-objective optimization problems which used the concept of Pareto dominance to determine the flight direction of a bee, and it maintained non-dominated solution vectors in an external archive [60]. Finally, Akbari et al. proposed a new type of MOABC utilizing different types of bees (i.e. employed bees, onlookers, and scouts) and a fixed-sized archive which was maintained by an ϵ -dominance method. In this algorithm, the social information provided by the external archive was used by the employed

bees to adjust their flying trajectories. The diversity over the external archive was controlled using a grid. The onlookers evaluate the solutions provided by the employed bees to adjust their next position. Finally, the scout bees replaced the solutions who had reached trial limit with a new random solution in the search space [3].

In this paper, we are going to suggest a new multi-objective ABC algorithm by dividing the whole searching space S into two subspaces S_1 and S_2 . We abbreviate this algorithm as dMOABC, and it uses three colonies: two basic colonies are searching in subspaces, while the synthetic colony explores in space S . They share information through a multi-colony communication model. An external archive is adopted to store and maintain non-dominated solutions found so far. For the purpose of diversity maintenance, the dMOABC utilizes a self-adaptive grid to divide the objective space and maintain the archive, thus the achieved solutions may distribute uniformly along the true Pareto front. For basic colonies, only neighbor information is used for generating new food sources, while in the synthetic colony, both neighbor and social information (shared by the external archive) are employed to adjust the flying trajectories of bees. Finally, the scout bee for each colony will do a random search if the food source is abandoned. Just like basic ABC algorithm, only one scout bee is allowed for each colony in each cycle.

The rest of the paper is organized as follows: some preliminaries on multi-objective optimization and artificial bee colony algorithm are presented in Section 2.

Section 3 describes the details of the proposed dMOABC algorithm for handling problems with multiple objectives. Next, the experimental study is presented in Section 4. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Multi-objective problems

Multi-objective problems (MOPs) concerns optimizing problems with multiple and often conflicting objectives. Generally, a MOP with D decision variables, M objectives, p inequality constraints and q equality constraints could be formulated as below [30]:

$$\begin{aligned} \text{Min } \mathbf{y} = \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_M(\mathbf{x})], \\ \text{s.t.} & \\ g_i(\mathbf{x}) &\leq 0, i = 1, 2, \dots, p, \\ h_j(\mathbf{x}) &= 0, j = 1, 2, \dots, q, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \dots, x_D) \in S$ and $\mathbf{y} = (f_1, f_2, \dots, f_M) \in Y$ are called decision and target vectors; S and Y denote the searching and target spaces, respectively.

Unlike single objective optimization, solutions of MOPs are in such a way that the performance of each objective cannot be improved without sacrificing the performance of at least another one. Hence, the solution to a MOP exists in the form of an alternate trade-off known as a Pareto optimal

Fig. 1 The flowchart of dMOABC algorithm

dMOABC algorithm
Input: <i>depth, ColonySize, limit, MaxIterations, ArchiveSize</i>
Output: <i>archive</i>
1. Determine values of parameters <i>depth, ColonySize, limit, MaxIterations, ArchiveSize</i> 2. Initialize the external archive <i>archive</i> 3. Divide the searching space S 4. Initialize colonies 4.1. Initialize <i>basicColony1</i> 4.2. Initialize <i>basicColony2</i> 4.3. Initialize <i>syntheticColony</i> 4.4. Initialize <i>gbest</i> 5. For ($t = 0; t < (maxCycle - 3)/3; t++$) { 5.1.1 <i>basicColony1.SendEmployedBees()</i> ; 5.1.2 <i>basicColony1.SendOnlookerBees()</i> ; 5.1.3 <i>basicColony1.SendScoutBees()</i> ; 5.2.1 <i>basicColony2.SendEmployedBees()</i> ; 5.2.2 <i>basicColony2.SendOnlookerBees()</i> ; 5.2.3 <i>basicColony2.SendScoutBees()</i> ; 5.3.1 <i>syntheticColony.SendEmployedBees()</i> ; 5.3.2 <i>syntheticColony.SendOnlookerBees()</i> ; 5.3.3 <i>syntheticColony.SendScoutBees()</i> ; } 6. Return <i>archive</i>

Fig. 2 The pseudo code of *Send_Employed_Bees*

```

Function Send_Employed_Bees
  For  $i = 1$  to FoodNumber
    Determine one dimension  $d$  to be changed randomly
    Select a neighbor  $k$  from the colony randomly
    For food source  $\mathbf{x}_i$ , calculate its new position  $\mathbf{v}_i$  according to different types of colonies
       $v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd})$ (basic colonies)
       $v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd}) + \varphi_{id} \cdot (x_{id} - gbest_d)$ (synthetic colony)
    Calculate each objective of  $\mathbf{v}_i$ 
    If ( $\mathbf{v}_i$  dominates  $\mathbf{x}_i$ )
       $\mathbf{x}_i = \mathbf{v}_i$ 
       $trial_i = 0$ 
      Try to add  $\mathbf{v}_i$  into the archive
       $gbest = update\_gbest(gbest, \mathbf{v}_i)$ 
    Else If ( $\mathbf{x}_i$  and  $\mathbf{v}_i$  are non-dominated with each other)
      If ( $\mathbf{v}_i$  could be successfully added into archive)
         $\mathbf{x}_i = \mathbf{v}_i$ 
         $trial_i = 0$ 
         $gbest = update\_gbest(gbest, \mathbf{v}_i)$ 
      Else
         $trial_i = trial_i + 1$ 
      End If
    Else
       $trial_i = trial_i + 1$ 
    End If
  End For
  Use function calculateFitness() to get fitness value of each food source
End of Function Send_Employed_Bees

```

set. The Pareto optimal set is defined based on Pareto dominance [3]. A vector \mathbf{x}_1 is said to dominate another vector \mathbf{x}_2 (denoted as $\mathbf{x}_1 < \mathbf{x}_2$), if and only if

$$\begin{aligned} f_i(\mathbf{x}_1) &\leq f_i(\mathbf{x}_2), i = 1, 2, \dots, M, \\ f_j(\mathbf{x}_1) &< f_j(\mathbf{x}_2), \exists j \in \{1, 2, \dots, M\}. \end{aligned} \quad (2)$$

Considering Pareto dominance, a vector \mathbf{x}_0 is called Pareto optimal if and only if $\neg \exists \mathbf{x} \in S$ such that $\mathbf{x} < \mathbf{x}_0$. We call the set containing all Pareto optimal solutions the Pareto optimal set (PS), which is defined as:

$$PS = \{\mathbf{x}_0 | \neg \exists \mathbf{x} \in S \text{ and } \mathbf{x} < \mathbf{x}_0\}. \quad (3)$$

Fig. 3 The pseudo code of *Send_Onlooker_Bees*

```

Function Send_Onlooker_Bees
  Calculate the probability prob for each food source by Eq.(12)
   $i=0, t=0;$ 
  While ( $t < FoodNumber$ )
    If ( $rand < prob_i$ )
       $t = t + 1$ 
      Determine one dimension  $d$  to be changed randomly
      Select a neighbor  $k$  from the colony randomly
      For food source  $\mathbf{x}_i$ , calculate its new position  $\mathbf{v}_i$  according to different types of colonies
         $v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd})$ (basic colonies)
         $v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd}) + \varphi_{id} \cdot (x_{id} - gbest_d)$ (synthetic colony)
      Calculate each objective of  $\mathbf{v}_i$ 
      If ( $\mathbf{v}_i$  dominates  $\mathbf{x}_i$ )
         $\mathbf{x}_i = \mathbf{v}_i$ 
         $trial_i = 0$ 
        Try to add  $\mathbf{v}_i$  into the archive
         $gbest = update\_gbest(gbest, \mathbf{v}_i)$ 
      Else If ( $\mathbf{x}_i$  and  $\mathbf{v}_i$  are non-dominated with each other)
        If ( $\mathbf{v}_i$  could be successfully added into archive)
           $\mathbf{x}_i = \mathbf{v}_i$ 
           $trial_i = 0$ 
           $gbest = update\_gbest(gbest, \mathbf{v}_i)$ 
        Else
           $trial_i = trial_i + 1$ 
        End If
      Else
         $trial_i = trial_i + 1$ 
      End If
    End If
     $i = i + 1$ 
    If ( $i == FoodNumber$ )
       $i = 0$ 
    End If
  End While
End of Function Send_Onlooker_Bees

```

Fig. 4 The pseudo code of *Send_scout_Bees*

```

Function Send_Scout_Bees
    Determine a food source  $k$  with maximum trial value
    If  $trial_k \geq limit$ 
         $x_{kd} = lb_d + rand(0, 1) \cdot (ub_d - lb_d)$ , where  $d = 1, 2, \dots, D$ .
        Evaluate the new food source  $x_k$ 
         $trial_k = 0$ 
    EndIf
End of Function Send_Scout_Bees
    
```

Therefore, a Pareto front (PF) for a given multi-objective problem and the Pareto optimal set PS , is defined as:

$$PF = \{f(x) = [f_1(x), f_2(x), \dots, f_M(x)] | x \in PS\}. \quad (4)$$

Given a set A and an element $x \in A$, x is called the non-dominated solution concerning set A if x is not dominated by any member in set A . And all non-dominated solutions will constitute a set, which is called the non-dominated set. A Pareto based multi-objective method tries to find the optimal Pareto front by continually updating the current non-dominated set, so as to make it approximate the true Pareto front as close as possible. When the algorithm is terminated, the output will be the final non-dominated set. However, the determination of a true Pareto front is a difficult task due to a large number of suboptimal Pareto fronts, and difficulty may be increased due to the nature of the Pareto front. More precisely, for most MOP methods, a partially convex, concave, or discontinuous problem is harder to solve compared to convex ones [3]. Actually, MOPs can be divided into two types: constrained and unconstrained MOPs. The focus of this paper is studying the second type of MOPs with only boundary constraint.

2.2 Artificial bee colony algorithm

Artificial bee colony algorithm (ABC), proposed by Karaboga in 2005 for real parameter optimization, is a recently introduced optimization algorithm which was

inspired by the method adopted from a swarm of honey bees trying to locate food sources [24]. In the ABC algorithm, the honey bee colony contains three different kinds of bees [2, 25], namely, employed bees, onlooker bees and scout bees. Half of the colony consists of employed bees, and the other half includes onlooker bees. Employed bees search the food around the food source in their memory, meanwhile they share information about the quality of the food source with onlooker bees. Onlooker bees tend to select good food sources from those founded by the employed bees, then further search the foods around the selected food source. The employed bee whose food source is exhausted will become a scout [18, 49].

The ABC algorithm starts with randomly producing food source sites that correspond to the solutions in the allowable domain. Initial food sources are generated randomly within the range of the boundaries of the parameters, formulated as below [49].

$$X_{ij} = lb_j + rand(0, 1)(ub_j - lb_j), \quad (5)$$

where $i = 1, 2, \dots, S_n$; $j = 1, 2, \dots, D$; S_n is the number of food sources and D is the dimensions of the problem to be optimized. ub_j and lb_j are upper and lower boundaries of parameter j respectively. $random(0, 1)$ is a random number uniformly distributed over interval $[0,1]$.

After initialization, each employed bee is sent to a food source site to find a neighboring food source using local information (visual information), and then its quality is evaluated. In ABC, the following equation is used to find a neighboring food source:

$$V_{ij} = X_{ij} + \phi_{ij}(X_{ij} - X_{kj}). \quad (6)$$

A neighboring food source V_i is determined by changing one parameter of each food source X_i . In (6), j is a random integer in the range $[1, D]$ and D is the number of dimensions. ϕ_{ij} is a random number uniformly distributed over the range $[-1,1]$ and k is the index of a randomly chosen solution. A parameter value produced by this operation may exceed its predetermined boundaries. In standard ABC, the value of the parameter exceeding its boundary is set to its boundaries, i.e. if $X_{ij} > ub_j$, then $X_{ij} = ub_j$; if $X_{ij} < lb_j$, then $X_{ij} = lb_j$. Both V_i and X_i are then compared

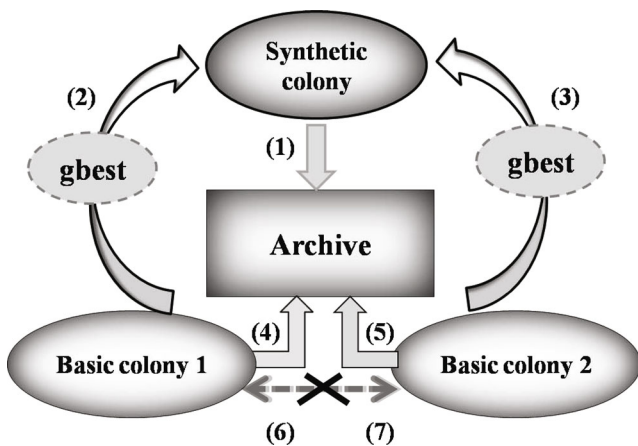


Fig. 5 Multi-colony model and communication mechanism

Table 1 Mathematical representation of the unconstrained CEC09 test problems

Problem	Mathematical representation
UF1	$f_1 = x_1 + \frac{2}{ J_1 } \sum_{j \in J_1} [x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{ J_2 } \sum_{j \in J_2} [x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2,$ <p>where $J_1 = \{j j \text{ is odd and } 2 \leq j \leq n\}$ and $J_2 = \{j j \text{ is even and } 2 \leq j \leq n\}$</p>
UF2	$f_1 = x_1 + \frac{2}{ J_1 } \sum_{j \in J_1} y_j^2, f_2 = 1 - \sqrt{x_1} + \frac{2}{ J_2 } \sum_{j \in J_2} y_j^2,$ <p>where $J_1 = \{j j \text{ is odd and } 2 \leq j \leq n\}$ and $J_2 = \{j j \text{ is even and } 2 \leq j \leq n\}$,</p> $y_j = \begin{cases} x_j - [0.3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1] \cos(6\pi x_1 + \frac{j\pi}{n}), & j \in J_1 \\ x_j - [0.3x_1^2 \cos(24\pi x_1 + \frac{4j\pi}{n}) + 0.6x_1] \sin(6\pi x_1 + \frac{j\pi}{n}), & j \in J_2 \end{cases}$
UF3	$f_1 = x_1 + \frac{2}{ J_1 } (4 \sum_{j \in J_1} y_j^2 - 2 \prod_{j \in J_1} \cos(\frac{20y_j\pi}{\sqrt{j}} + 2))$ $f_2 = 1 - \sqrt{x_1} + \frac{2}{ J_2 } (4 \sum_{j \in J_2} y_j^2 - 2 \prod_{j \in J_2} \cos(\frac{20y_j\pi}{\sqrt{j}} + 2)),$ <p>where J_1 and J_2 are the same as those of UF1, and $y_j = x_j - x_1^{0.5(1.0 + \frac{3(j-2)}{n-2})}, j = 2, \dots, n$.</p>
UF4	$f_1 = x_1 + \frac{2}{ J_1 } \sum_{j \in J_1} h(y_j), f_2 = 1 - x_1^2 + \frac{2}{ J_2 } \sum_{j \in J_2} h(y_j),$ <p>where $J_1 = \{j j \text{ is odd and } 2 \leq j \leq n\}$ and $J_2 = \{j j \text{ is even and } 2 \leq j \leq n\}$,</p> $y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \dots, n, \text{ and } h(t) = \frac{ t }{1+e^{2t\pi}}.$
UF5	$f_1 = x_1 + (\frac{1}{2N} + \varepsilon) \sin(2N\pi x_1) + \frac{2}{ J_1 } \sum_{j \in J_1} h(y_j),$ $f_2 = 1 - x_1 + (\frac{1}{2N} + \varepsilon) \sin(2N\pi x_1) + \frac{2}{ J_2 } \sum_{j \in J_2} h(y_j),$ <p>where $J_1 = \{j j \text{ is odd and } 2 \leq j \leq n\}$ and $J_2 = \{j j \text{ is even and } 2 \leq j \leq n\}$. N is an integer, $\varepsilon > 0$, $y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \dots, n$, and $h(t) = 2t^2 - \cos(4\pi t) + 1$.</p>
UF6	$f_1 = x_1 + \max\{0, 2(\frac{1}{2N} + \varepsilon) \sin(2N\pi x_1)\} + \frac{2}{ J_1 } (4 \sum_{j \in J_1} y_j^2 - 2 \prod_{j \in J_1} \cos(\frac{20y_j\pi}{\sqrt{j}} + 2))$ $f_2 = 1 - x_1 + \max\{0, 2(\frac{1}{2N} + \varepsilon) \sin(2N\pi x_1)\} + \frac{2}{ J_2 } (4 \sum_{j \in J_2} y_j^2 - 2 \prod_{j \in J_2} \cos(\frac{20y_j\pi}{\sqrt{j}} + 2))$ <p>where $J_1 = \{j j \text{ is odd and } 2 \leq j \leq n\}$ and $J_2 = \{j j \text{ is even and } 2 \leq j \leq n\}$, and $y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \dots, n$.</p>
UF7	$f_1 = \sqrt[3]{x_1} + \frac{2}{ J_1 } \sum_{j \in J_1} y_j^2, f_2 = 1 - \sqrt[3]{x_1} + \frac{2}{ J_2 } \sum_{j \in J_2} y_j^2,$ <p>where $J_1 = \{j j \text{ is odd and } 2 \leq j \leq n\}$, $J_2 = \{j j \text{ is even and } 2 \leq j \leq n\}$, and $y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \dots, n$.</p>

against each other and the employed bee exploits the better food source in terms of fitness value which is calculated by (7), and this is known as *greedy selection mechanism*.

$$fit_i = \begin{cases} 1/(1 + f_i) & \text{if } f_i \geq 0 \\ 1 + abs(f_i) & \text{if } f_i < 0 \end{cases}, \tag{7}$$

where f_i is the objective value of solution V_i or X_i . Equation 7 is used to calculate fitness values for a minimization problem, while the objective function can be directly used as a fitness function for maximization problems [49].

In the next step, each onlooker bee randomly chooses a food source according to the probability given in (8) by

using roulette wheel selection scheme. Then, each onlooker bee tries to improve the selected food source using (6):

$$p_i = \frac{fit_i}{\sum_{j=1}^n fit_j}, \tag{8}$$

where fit_j is the fitness value of i th food source.

Finally, if a certain food source i cannot be improved for a predetermined number of cycles, referred to as *limit*, this food source is then abandoned. The employed bee that was exploiting this food source becomes a scout that looks for a new food source by randomly searching the problem domain using (5). In basic ABC, it is assumed that only one source can be exhausted in each cycle, and only one

Table 1 Continued.

Problem Mathematical representation

UF8

$$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|} \sum_{j \in J_1} (x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|} \sum_{j \in J_2} (x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_3 = \sin(0.5x_1\pi) + \frac{2}{|J_3|} \sum_{j \in J_3} (x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

where $J_1 = \{j|3 \leq j \leq n \text{ and } j - 1 \text{ is a multiplication of } 3\}$, $J_2 = \{j|3 \leq j \leq n \text{ and } j - 2 \text{ is a multiplication of } 3\}$, $J_3 = \{j|3 \leq j \leq n \text{ and } j \text{ is a multiplication of } 3\}$.

UF9

$$f_1 = 0.5[\max\{0, (1 + \varepsilon)(1 - 4(2x_1 - 1)^2)\} + 2x_1]x_2 + \frac{2}{|J_1|} \sum_{j \in J_1} (x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2 = 0.5[\max\{0, (1 + \varepsilon)(1 - 4(2x_1 - 1)^2)\} - 2x_1 + 2]x_2 + \frac{2}{|J_2|} \sum_{j \in J_2} (x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_3 = 1 - x_2 + \frac{2}{|J_3|} \sum_{j \in J_3} (x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2, \text{ where } J_1 = \{j|3 \leq$$

$j \leq n \text{ and } j - 1 \text{ is a multiplication of } 3\}$, $J_2 = \{j|3 \leq j \leq n \text{ and } j - 2 \text{ is a multiplication of } 3\}$, $J_3 = \{j|3 \leq j \leq n \text{ and } j \text{ is a multiplication of } 3\}$, and $\varepsilon = 0.1$, and it can take any other positive values.

UF10

$$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|} \sum_{j \in J_1} [4y_j^2 - \cos(8\pi y_j) + 1]$$

$$f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|} \sum_{j \in J_2} [4y_j^2 - \cos(8\pi y_j) + 1]$$

$$f_3 = \sin(0.5x_1\pi) + \frac{2}{|J_3|} \sum_{j \in J_3} [4y_j^2 - \cos(8\pi y_j) + 1]$$

where $J_1 = \{j|3 \leq j \leq n \text{ and } j - 1 \text{ is a multiplication of } 3\}$, $J_2 = \{j|3 \leq j \leq n \text{ and } j - 2 \text{ is a multiplication of } 3\}$, $J_3 = \{j|3 \leq j \leq n \text{ and } j \text{ is a multiplication of } 3\}$, and $y_j = x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n})$, $j = 3, \dots, n$.

employed bee can be a scout. The algorithm will be terminated after repeating a predefined max number of cycles, denoted as *Max_Cycles*.

3 Multi-objective artificial bee colony algorithm based on division of the searching space

Our dMOABC algorithm is designed by dividing the searching space S into two independent parts S_1 and S_2 , and it uses a special multi-colony model which will be described in Subsection 3.7 in detail. In this model, three colonies are used: two of them are basic colonies and they search in S_1 and S_2 , while the third one is called synthetic colony whose search space is S . In dMOABC, all control parameters that must be determined beforehand are:

- *depth*, the number of recursive subdivisions of the objective space carried out in order to divide the objective space into a grid for the purposes of diversity maintenance. Values between 3 and 6 are useful, depending on number of objectives [28].

- *ColonySize*, the number of colony size, namely, the sum of the numbers of employed bees and onlooker bees. In our algorithm, three colonies are with the same colony size.
- *FoodNumber*, the number of food sources. Since the number of employed bees equals to that of onlooker bees in dMOABC algorithm, the value of *FoodNumber* will be the half of the *ColonySize*.
- *limit*, the abandonment criteria. A food source which could not be improved through "limit" trials will be abandoned by its employed bee.
- *MaxIterations*, the termination criteria. The value of this parameter could be tuned according to the maximal number of function evaluations(*max_FEs*) and *Colony-Size*.
- *ArchiveSize*, the maximum number of non-dominated solutions stored in the archive.

The dMOABC is a Pareto based algorithm with an external archive to store non-dominated solutions. Since the length of the archive are usually limited, we need an updating method to maintain it so that the latest solutions

Table 2 HV. Mean and standard deviation

	dMOABC	MOEAD	SMPSO	GDE3
UF1	6.58e-01 _{6.2e-04}	6.64e-01 _{1.5e-04}	5.56e-01 _{1.5e-02}	5.71e-01 _{1.5e-02}
UF2	6.59e-01 _{5.2e-04}	6.57e-01 _{2.3e-03}	6.35e-01 _{3.2e-03}	6.35e-01 _{5.3e-03}
UF3	5.67e-01 _{1.3e-02}	6.54e-01 _{1.2e-02}	4.97e-01 _{4.6e-02}	3.10e-01 _{1.9e-02}
UF4	2.84e-01 _{8.5e-04}	2.47e-01 _{6.1e-03}	2.56e-01 _{5.3e-03}	2.76e-01 _{4.3e-04}
UF5	3.50e-01 _{2.0e-02}	6.30e-02 _{6.5e-02}	1.08e-03 _{5.8e-03}	2.19e-02 _{7.5e-02}
UF6	3.51e-01 _{1.6e-02}	1.97e-01 _{1.0e-01}	7.27e-02 _{8.3e-02}	1.12e-01 _{1.2e-01}
UF7	4.81e-01 _{5.7e-03}	4.82e-01 _{7.1e-02}	4.60e-01 _{4.7e-03}	4.51e-01 _{1.7e-03}
UF8	2.44e-01 _{2.9e-02}	4.65e-02 _{2.2e-02}	1.26e-01 _{4.4e-02}	5.21e-02 _{2.7e-02}
UF9	4.76e-01 _{7.8e-02}	1.40e-01 _{3.5e-02}	1.88e-01 _{6.7e-02}	3.22e-01 _{1.7e-01}
UF10	2.25e-01 _{2.2e-02}	4.76e-02 _{2.0e-02}	3.59e-02 _{3.3e-02}	1.07e-02 _{2.7e-02}
	AbYSS	CellDE	IBEA	MOCcell
UF1	5.58e-01 _{4.3e-02}	5.98e-01 _{3.0e-02}	5.39e-01 _{2.1e-02}	5.12e-01 _{4.8e-02}
UF2	6.34e-01 _{8.3e-03}	6.39e-01 _{1.6e-03}	6.26e-01 _{1.1e-02}	6.24e-01 _{1.1e-02}
UF3	4.03e-01 _{5.0e-02}	5.04e-01 _{1.6e-02}	3.79e-01 _{4.7e-02}	3.30e-01 _{4.5e-02}
UF4	2.59e-01 _{5.1e-03}	2.66e-01 _{3.2e-03}	2.44e-01 _{6.5e-03}	2.69e-01 _{1.4e-03}
UF5	1.83e-01 _{5.5e-02}	5.57e-03 _{1.2e-02}	1.61e-01 _{6.9e-02}	1.36e-01 _{8.1e-02}
UF6	2.27e-01 _{7.3e-02}	1.99e-01 _{1.2e-01}	2.11e-01 _{7.4e-02}	1.70e-01 _{9.2e-02}
UF7	2.85e-01 _{7.8e-02}	4.44e-01 _{4.8e-02}	3.33e-01 _{1.2e-01}	3.16e-01 _{1.4e-01}
UF8	2.12e-01 _{6.8e-02}	5.56e-03 _{1.0e-02}	1.94e-01 _{4.8e-02}	7.57e-02 _{5.4e-02}
UF9	3.22e-01 _{9.5e-02}	1.57e-02 _{3.1e-02}	5.35e-01 _{4.2e-02}	1.78e-01 _{7.3e-02}
UF10	7.02e-02 _{4.0e-02}	0.00e+00 _{0.0e+00}	5.95e-02 _{6.3e-02}	4.85e-03 _{9.4e-03}
	OMOPSO	NSGAI	PAES	SPEA2
UF1	5.42e-01 _{1.3e-02}	5.60e-01 _{3.3e-02}	4.51e-01 _{8.2e-02}	5.43e-01 _{2.1e-02}
UF2	6.37e-01 _{2.1e-03}	6.32e-01 _{8.0e-03}	5.99e-01 _{6.4e-02}	6.30e-01 _{8.4e-03}
UF3	5.76e-01 _{3.9e-02}	4.62e-01 _{4.3e-02}	2.63e-01 _{5.8e-02}	4.30e-01 _{4.6e-02}
UF4	2.61e-01 _{3.3e-03}	2.64e-01 _{8.3e-04}	1.93e-01 _{1.5e-02}	2.70e-01 _{4.5e-04}
UF5	4.44e-02 _{6.0e-02}	1.76e-01 _{7.0e-02}	1.20e-01 _{8.8e-02}	1.79e-01 _{7.4e-02}
UF6	1.25e-01 _{7.1e-02}	2.41e-01 _{7.3e-02}	1.03e-01 _{9.4e-02}	2.34e-01 _{8.8e-02}
UF7	4.49e-01 _{4.9e-03}	3.52e-01 _{9.4e-02}	1.76e-01 _{1.3e-01}	3.52e-01 _{9.8e-02}
UF8	1.12e-02 _{1.7e-02}	1.49e-01 _{6.1e-02}	1.58e-01 _{6.6e-02}	1.43e-01 _{2.1e-02}
UF9	2.72e-02 _{3.0e-02}	3.42e-01 _{9.6e-02}	4.10e-01 _{1.1e-01}	5.22e-01 _{5.0e-02}
UF10	0.00e+00 _{0.0e+00}	2.21e-02 _{2.8e-02}	4.58e-02 _{2.6e-02}	3.36e-02 _{2.9e-02}

included are not dominated by other archive members or vice versa. The dMOABC uses a self-adaptive grid method that was used in PAES [27] to achieve and maintain diversity. This method will be described in Subsection 3.3. In our dMOABC method, all colonies share the same archive, and it should be initialized as null before the algorithm really goes into run.

There are 6 parameters in dMOABC, but *ColonySize*, *MaxIterations* and *ArchiveSize* are general parameters, i.e., they are involved in many multi-objective intelligent algorithms. *FoodNumber* is determined by *ColonySize*, i.e., $FoodNumber = ColonySize/2$. Hence, there are only two parameters *depth* and *limit* that should be tuned by the users. The value of *depth* is suggested to be set between 3 and 6 [28]. The larger of this value, the better of the approximated fronts will be, and the longer the computation time will be. The *limit* is an important parameter in ABC algorithm family. According to our empirical experiment [49], this parameter is recommended to be set at a value varying from 100 to 200 for general usage.

The flowchart of our dMOABC algorithm is shown in Fig. 1. The dMOABC method is constituted of the following main parts: *Division*, *Initialization*, *Send Employed Bees*,

Send Onlooker Bees, *Maintain archive*, *Send Scout Bees* and *Information Exchange*. These parts are explained in the following subsections respectively.

3.1 Division

Each problem considered in this work has a number of decision parameters, and these parameters are continuous. Each parameter is limited to a span, between lower and upper bound values, which could be formulated as $lb_d \leq x_d \leq ub_d$, where x_d is the d th parameter or dimension, and lb_d and ub_d are lower and upper bounds of this dimension, respectively. Thus, for a D dimensional problem, the searching space $S = [lb_1, ub_1] \times [lb_2, ub_2] \times \dots \times [lb_D, ub_D]$ will be a subset of R^D . For each colony, each food source will be associated with a position vector $\mathbf{x} = (x_1, x_2, \dots, x_D) \in S$.

In our algorithm, the whole search space S is divided into two independent subsets, denoted as S_1 and S_2 . This division is executed by equally dividing the first dimension of the problem, that is to say, the ranges of the first dimension of S_1 and S_2 are $[lb_1, \frac{(lb_1 + ub_1)}{2}]$ and $[\frac{(lb_1 + ub_1)}{2}, ub_1]$, respectively. And the 2 to D dimensions of S_1 and S_2 keep

Table 3 HV, Median and IQR

	dMOABC	MOEAD	SMPSO	GDE3
UF1	6.58e-018.1e-04	6.64e-012.2e-04	5.57e-012.6e-02	5.67e-019.0e-03
UF2	6.59e-016.8e-04	6.57e-013.8e-03	6.35e-014.5e-03	6.36e-012.7e-03
UF3	5.67e-011.8e-02	6.59e-017.9e-03	5.04e-016.5e-02	3.11e-012.9e-02
UF4	2.84e-019.5e-04	2.48e-018.4e-03	2.56e-018.3e-03	2.76e-014.0e-04
UF5	3.52e-012.5e-02	6.03e-021.1e-01	0.00e+000.0e+00	0.00e+000.0e+00
UF6	3.53e-012.1e-02	2.34e-017.6e-02	3.03e-028.4e-02	1.50e-042.2e-01
UF7	4.83e-017.6e-03	4.96e-017.6e-04	4.60e-015.2e-03	4.51e-012.5e-03
UF8	2.41e-014.0e-02	5.24e-023.4e-02	1.17e-014.2e-02	4.63e-024.4e-02
UF9	4.74e-011.4e-01	1.30e-014.5e-02	1.70e-016.3e-02	4.10e-013.4e-01
UF10	2.30e-012.8e-02	5.31e-022.7e-02	2.32e-024.5e-02	0.00e+001.0e-02
	AbYSS	CellDE	IBEA	MOCeII
UF1	5.84e-015.1e-02	6.04e-015.1e-02	5.45e-013.3e-02	5.19e-016.6e-02
UF2	6.35e-017.8e-03	6.39e-012.9e-03	6.28e-018.2e-03	6.26e-019.4e-03
UF3	3.90e-017.9e-02	5.07e-012.1e-02	3.72e-016.5e-02	3.32e-017.4e-02
UF4	2.60e-017.3e-03	2.67e-013.1e-03	2.44e-019.8e-03	2.69e-011.7e-03
UF5	1.87e-018.3e-02	0.00e+001.6e-03	1.63e-018.6e-02	1.53e-011.3e-01
UF6	2.44e-011.2e-01	2.26e-011.8e-01	2.31e-016.0e-02	1.86e-011.5e-01
UF7	2.62e-016.1e-02	4.59e-011.3e-02	4.06e-012.3e-01	3.15e-012.4e-01
UF8	2.42e-017.8e-02	1.24e-045.4e-03	2.07e-016.9e-04	6.15e-027.0e-02
UF9	3.34e-011.3e-01	0.00e+001.9e-02	5.29e-013.2e-02	1.72e-019.6e-02
UF10	7.33e-024.8e-02	0.00e+000.0e+00	4.13e-027.0e-02	6.05e-046.8e-03
	OMOPSO	NSGAI	PAES	SPEA2
UF1	5.44e-011.8e-02	5.72e-011.8e-02	4.72e-011.2e-01	5.49e-012.2e-02
UF2	6.37e-012.8e-03	6.33e-011.1e-02	6.15e-011.3e-02	6.30e-018.4e-03
UF3	5.84e-015.2e-02	4.76e-016.4e-02	2.70e-019.3e-02	4.35e-015.9e-02
UF4	2.61e-013.8e-03	2.64e-011.1e-03	1.96e-011.6e-02	2.70e-017.0e-04
UF5	2.75e-039.1e-02	1.85e-019.5e-02	1.28e-011.9e-01	1.98e-018.5e-02
UF6	1.26e-011.2e-01	2.44e-018.4e-02	8.11e-022.0e-01	2.52e-018.0e-02
UF7	4.49e-016.0e-03	3.44e-011.9e-01	1.59e-011.6e-01	3.55e-011.8e-01
UF8	7.18e-031.4e-02	1.76e-011.1e-01	1.85e-015.0e-02	1.41e-011.5e-02
UF9	1.70e-024.2e-02	3.06e-011.8e-01	4.17e-011.8e-01	5.39e-018.4e-02
UF10	0.00e+000.0e+00	8.23e-034.3e-02	5.31e-024.5e-02	3.04e-025.8e-02

the same as S . Mathematically, $S_1 = [lb_1, \frac{(lb_1 + ub_1)}{2}] \times [lb_2, ub_2] \times \dots \times [lb_D, ub_D]$ and $S_2 = [\frac{(lb_1 + ub_1)}{2}, ub_1] \times [lb_2, ub_2] \times \dots \times [lb_D, ub_D]$, where "×" represents Cartesian product. It should be noted here that, except the first dimension, this division can be carried out according to any other dimensions. Two basic colonies will search in S_1 and S_2 independently, while the synthetic colony will seek solutions in the whole searching space S .

The division will enable more refined search for good solutions with less chances of missing some parts of the searching space S . Since the goal of multi-objective algorithms is to approximate a set instead of a single point, the performance of the algorithm is evaluated based on the closeness between the approximated and the true fronts, as well as the distribution of the computed non-dominated set. Thus, to achieve better approximated front, the algorithm can't miss any part of the whole PF. When using only one colony, once it is dragged into a local minima, then there will be a higher chance that the algorithm may miss some parts of the whole true PF. But, if we use multi colonies, the probability of the occurrence of the above situation will be much lower. In our algorithm, as described before, basic

colonies are independently searching in the subspaces, thus, the whole PF is divided into two parts and each part corresponds to a basic colony. Even if a basic colony is trapped, the synthetic colony could provide necessary remedy. This mechanism enhances the exploration abilities of the algorithm. Of course, more subspaces are feasible, but this may sacrifice the exploitation abilities of each colony, since the maximal number of function evaluations is fixed. To balance exploration and exploitation, we here use three colonies in the dMOABC algorithm.

3.2 Initialization

In the initialization phase, *FoodNumber* food sources will be randomly generated for each colony according to its corresponding searching space. Note again that the searching spaces for two basic colonies are S_1 and S_2 , and that for synthetic colony is S . To be convenient for explanation, we will only take the synthetic colony as an example to show how to initialize it:

First, each food source will be initialized by a function named *init*(i, S), where i is the index of the food source and S is the searching space defined before. In this way,

Table 4 SPREAD. Mean and standard deviation

	dMOABC	MOEAD	SMPSO	GDE3
UF1	5.51e - 01 _{5.4e-02}	3.47e - 01 _{9.8e-02}	9.28e - 01 _{1.1e-01}	1.25e + 00 _{7.8e-02}
UF2	4.66e - 01 _{4.2e-02}	4.76e - 01 _{8.3e-02}	3.24e - 01 _{5.1e-02}	5.02e - 01 _{6.6e-02}
UF3	1.15e + 00 _{1.3e-01}	3.44e - 01 _{1.3e-01}	4.05e - 01 _{1.2e-01}	1.68e + 00 _{9.5e-02}
UF4	5.53e - 01 _{4.3e-02}	4.27e - 01 _{6.3e-02}	3.06e - 01 _{5.6e-02}	2.76e - 01 _{1.5e-02}
UF5	6.09e - 01 _{1.0e-01}	1.26e + 00 _{7.9e-02}	9.79e - 01 _{1.0e-01}	1.92e + 00 _{1.6e-01}
UF6	8.06e - 01 _{9.3e-02}	1.13e + 00 _{1.1e-01}	1.03e + 00 _{8.9e-02}	1.89e + 00 _{1.8e-01}
UF7	7.32e - 01 _{1.1e-01}	2.74e - 01 _{2.9e-01}	5.96e - 01 _{5.8e-02}	8.00e - 01 _{2.7e-02}
UF8	9.48e - 01 _{9.6e-02}	9.77e - 01 _{2.6e-02}	7.50e - 01 _{6.6e-02}	6.31e - 01 _{4.6e-02}
UF9	9.13e - 01 _{8.2e-02}	9.88e - 01 _{1.7e-02}	8.52e - 01 _{7.0e-02}	7.33e - 01 _{5.3e-02}
UF10	1.09e + 00 _{1.5e-01}	9.36e - 01 _{3.1e-02}	7.38e - 01 _{6.7e-02}	6.81e - 01 _{7.2e-02}
	AbYSS	CellDE	IBEA	MOCcell
UF1	1.11e + 00 _{1.2e-01}	7.67e - 01 _{1.8e-01}	1.44e + 00 _{7.9e-02}	1.31e + 00 _{9.2e-02}
UF2	3.83e - 01 _{2.4e-02}	4.69e - 01 _{4.6e-02}	9.85e - 01 _{6.4e-02}	4.39e - 01 _{6.3e-02}
UF3	1.10e + 00 _{1.0e-01}	1.07e + 00 _{1.6e-01}	1.07e + 00 _{7.9e-02}	1.10e + 00 _{9.8e-02}
UF4	3.40e - 01 _{7.3e-02}	4.11e - 01 _{3.5e-02}	1.37e + 00 _{1.4e-01}	2.54e - 01 _{4.7e-02}
UF5	1.01e + 00 _{4.1e-01}	1.32e + 00 _{2.5e-01}	1.25e + 00 _{2.4e-01}	1.14e + 00 _{1.9e-01}
UF6	1.21e + 00 _{2.8e-01}	1.44e + 00 _{3.9e-01}	1.29e + 00 _{2.6e-01}	1.18e + 00 _{2.2e-01}
UF7	1.06e + 00 _{2.0e-01}	6.96e - 01 _{3.2e-01}	1.18e + 00 _{6.9e-02}	9.37e - 01 _{2.3e-01}
UF8	8.63e - 01 _{1.1e-01}	6.53e - 01 _{6.3e-02}	7.83e - 01 _{2.5e-01}	7.93e - 01 _{7.6e-02}
UF9	9.04e - 01 _{6.4e-02}	6.63e - 01 _{5.2e-02}	1.33e + 00 _{9.0e-02}	8.93e - 01 _{7.8e-02}
UF10	9.16e - 01 _{1.1e-01}	5.98e - 01 _{4.8e-02}	1.36e + 00 _{3.8e-01}	8.47e - 01 _{6.5e-02}
	OMOPSO	NSGAI	PAES	SPEA2
UF1	1.04e + 00 _{1.3e-01}	1.27e + 00 _{1.1e-01}	1.36e + 00 _{1.7e-01}	1.44e + 00 _{8.1e-02}
UF2	3.35e - 01 _{7.0e-02}	5.79e - 01 _{4.4e-02}	9.71e - 01 _{1.2e-01}	4.73e - 01 _{5.8e-02}
UF3	4.09e - 01 _{1.2e-01}	1.30e + 00 _{1.1e-01}	1.06e + 00 _{6.6e-02}	1.22e + 00 _{1.2e-01}
UF4	2.60e - 01 _{3.0e-02}	4.64e - 01 _{3.0e-02}	1.06e + 00 _{1.1e-01}	2.32e - 01 _{2.8e-02}
UF5	9.47e - 01 _{1.1e-01}	1.46e + 00 _{1.5e-01}	1.03e + 00 _{9.3e-02}	1.37e + 00 _{1.7e-01}
UF6	1.07e + 00 _{1.5e-01}	1.48e + 00 _{2.0e-01}	9.95e - 01 _{1.2e-01}	1.44e + 00 _{2.4e-01}
UF7	6.47e - 01 _{7.9e-02}	9.95e - 01 _{1.9e-01}	1.09e + 00 _{7.6e-02}	1.03e + 00 _{2.1e-01}
UF8	8.25e - 01 _{7.8e-02}	7.94e - 01 _{1.0e-01}	7.43e - 01 _{1.6e-01}	5.98e - 01 _{3.6e-02}
UF9	8.95e - 01 _{6.7e-02}	9.09e - 01 _{8.3e-02}	7.26e - 01 _{7.2e-02}	6.16e - 01 _{4.1e-02}
UF10	7.35e - 01 _{6.0e-02}	8.27e - 01 _{8.0e-02}	1.16e + 00 _{7.3e-02}	6.96e - 01 _{7.3e-02}

a D dimensional vector $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ will be assigned randomly to food source i through the following equation:

$$x_{id} = lb_d + rand(0, 1) \cdot (ub_d - lb_d), \tag{9}$$

where $i = 1, 2, \dots, FoodNumber$; $d = 1, 2, \dots, D$, and $rand(0, 1)$ is a random number distributed uniformly over the interval $[0,1]$; lb_d and ub_d are lower and upper bounds of the d th dimension respectively.

Second, a variable $trial_i$ will be assigned to each food source i in order to find food sources to be abandoned in the next iterations. If a food source could not get optimized in a number of trials (i.e. $limit$), its related employed bee will turn into a scout bee and after doing a random search, it will turn back to be an employed bee again. Variable $trial_i$ is a counter of unsuccessful trials for food source i , and each $trial_i, i = 1, 2, \dots, FoodNumber$ is set to 0 in the initialization phase.

Finally, we calculate the objectives for each food source, and try to add it into the external archive. The global best food source $gbest$ is initialized as a solution selected from the archive by using roulette wheel method. If the location of a solution is less crowded, then it will be chosen

with higher probability. The following subsection will give details on archive maintenance.

3.3 Maintain archive

In dMOABC, an external archive is used to store and update all of the non-dominated solutions. The archive has a maximum size, $ArchiveSize$, which is set by the user to reflect the required number of final solutions desired. In order to make sure that solutions in the archive approximate the true Pareto front as close as possible, a self-adaptive grid is adopted for the purpose of diversity maintenance. Actually, maintaining archive involves two main tasks:

- *Update grid* [27]. For a problem with m criterions, the whole objective space is divided by a m -dimensional grid which is used to keep track of the degree of crowding in different regions of the solution space. When a solution is generated its grid location is determined by using recursive subdivision and noted using a tree encoding. A map of the grid is also maintained, indicating for each grid location how many and which solutions in the archive currently reside there. When trying to add a candidate solution into the archive,

Table 5 SPREAD. Median and IQR

	dMOABC	MOEAD	SMPSO	GDE3
UF1	5.45e - 01 _{9.3e-02}	3.11e - 01 _{4.3e-02}	9.45e - 01 _{1.6e-01}	1.25e + 00 _{1.0e-01}
UF2	4.62e - 01 _{5.5e-02}	4.53e - 01 _{7.0e-02}	3.22e - 01 _{4.7e-02}	5.10e - 01 _{1.0e-01}
UF3	1.15e + 00 _{1.9e-01}	2.92e - 01 _{3.8e-02}	4.11e - 01 _{1.5e-01}	1.65e + 00 _{1.7e-01}
UF4	5.53e - 01 _{5.8e-02}	4.13e - 01 _{1.0e-01}	2.85e - 01 _{1.1e-01}	2.77e - 01 _{1.6e-02}
UF5	6.08e - 01 _{9.1e-02}	1.26e + 00 _{1.5e-01}	9.66e - 01 _{1.2e-01}	1.98e + 00 _{0.0e+00}
UF6	7.84e - 01 _{9.8e-02}	1.12e + 00 _{1.3e-01}	1.02e + 00 _{5.5e-02}	1.98e + 00 _{2.4e-02}
UF7	7.19e - 01 _{1.6e-01}	1.42e - 01 _{1.0e-01}	5.85e - 01 _{8.3e-02}	8.03e - 01 _{2.8e-02}
UF8	9.51e - 01 _{1.5e-01}	9.80e - 01 _{2.8e-02}	7.30e - 01 _{9.1e-02}	6.33e - 01 _{4.8e-02}
UF9	9.03e - 01 _{8.5e-02}	9.92e - 01 _{2.3e-02}	8.55e - 01 _{7.2e-02}	7.27e - 01 _{6.0e-02}
UF10	1.09e + 00 _{2.2e-01}	9.38e - 01 _{4.4e-02}	7.46e - 01 _{6.3e-02}	6.76e - 01 _{1.2e-01}
	AbYSS	CellDE	IBEA	MOCcell
UF1	1.06e + 00 _{1.0e-01}	7.59e - 01 _{2.3e-01}	1.43e + 00 _{4.1e-02}	1.33e + 00 _{1.3e-01}
UF2	3.86e - 01 _{3.6e-02}	4.73e - 01 _{4.4e-02}	9.98e - 01 _{1.0e-01}	4.33e - 01 _{9.5e-02}
UF3	1.06e + 00 _{2.0e-01}	1.06e + 00 _{2.2e-01}	1.04e + 00 _{1.0e-01}	1.09e + 00 _{1.1e-01}
UF4	3.22e - 01 _{5.4e-02}	4.02e - 01 _{3.2e-02}	1.40e + 00 _{1.8e-01}	2.47e - 01 _{6.8e-02}
UF5	1.16e + 00 _{6.5e-01}	1.36e + 00 _{2.1e-01}	1.27e + 00 _{5.0e-01}	1.05e + 00 _{2.0e-01}
UF6	1.33e + 00 _{3.3e-01}	1.44e + 00 _{3.5e-01}	1.32e + 00 _{6.0e-01}	1.02e + 00 _{3.3e-01}
UF7	1.14e + 00 _{1.3e-01}	6.02e - 01 _{1.2e-01}	1.20e + 00 _{9.5e-02}	1.00e + 00 _{4.4e-01}
UF8	9.04e - 01 _{1.4e-01}	6.39e - 01 _{1.1e-01}	7.25e - 01 _{6.6e-02}	7.77e - 01 _{9.7e-02}
UF9	8.98e - 01 _{6.0e-02}	6.67e - 01 _{5.7e-02}	1.32e + 00 _{1.3e-01}	9.04e - 01 _{9.0e-02}
UF10	9.17e - 01 _{1.2e-01}	5.95e - 01 _{7.8e-02}	1.25e + 00 _{7.3e-01}	8.56e - 01 _{8.5e-02}
	OMOPSO	NSGAI	PAES	SPEA2
UF1	1.05e + 00 _{2.1e-01}	1.26e + 00 _{1.8e-01}	1.32e + 00 _{2.6e-01}	1.43e + 00 _{8.4e-02}
UF2	3.27e - 01 _{9.1e-02}	5.80e - 01 _{4.8e-02}	9.24e - 01 _{1.5e-01}	4.76e - 01 _{9.2e-02}
UF3	4.04e - 01 _{1.8e-01}	1.32e + 00 _{1.7e-01}	1.05e + 00 _{7.9e-02}	1.22e + 00 _{1.7e-01}
UF4	2.59e - 01 _{3.2e-02}	4.67e - 01 _{5.3e-02}	1.06e + 00 _{1.1e-01}	2.27e - 01 _{3.4e-02}
UF5	9.39e - 01 _{1.6e-01}	1.47e + 00 _{1.7e-01}	1.00e + 00 _{4.2e-02}	1.36e + 00 _{1.5e-01}
UF6	1.05e + 00 _{1.9e-01}	1.53e + 00 _{2.0e-01}	1.00e + 00 _{8.7e-02}	1.43e + 00 _{2.8e-01}
UF7	6.40e - 01 _{1.2e-01}	1.09e + 00 _{3.9e-01}	1.08e + 00 _{1.4e-01}	1.13e + 00 _{3.9e-01}
UF8	8.25e - 01 _{1.1e-01}	7.88e - 01 _{1.4e-01}	6.93e - 01 _{2.3e-01}	5.94e - 01 _{5.2e-02}
UF9	9.02e - 01 _{6.5e-02}	9.12e - 01 _{1.3e-01}	7.16e - 01 _{8.2e-02}	6.18e - 01 _{5.9e-02}
UF10	7.24e - 01 _{1.1e-01}	8.27e - 01 _{1.2e-01}	1.15e + 00 _{1.3e-01}	6.83e - 01 _{8.3e-02}

the algorithm needs to first check whether the candidate exceeds the grid or not. If the candidate is located beyond the grid, the grid will be self-adaptively updated so as to make the new solution included. The objective space is then redivided, and grid locations for the new candidate and all solutions in the archive are recalculated. Besides, the degree of crowding(i.e., grid-location count) for each grid is also redetermined.

- Add candidate solution into the archive [28]. After the grid is updated (if necessary), the new solution s will be added into the archive provided one of the following conditions is satisfied:

- the archive is empty ;
- the archive is not full and s is not dominated by or equal to any member in the current archive;
- s dominates any member in the archive ;
- the archive is full but s is non-dominated and is in a no more crowded square than at least one solution in the archive.

In addition, the archive will be maintained such that all solutions are non-dominated. To be specific, all solutions that are dominated by s will be removed from the archive in case c, and s will replace one of the archived solutions with the highest grid-location count in case d.

3.4 Send employed bees

The pseudo code of *Send_Employed_Bees()* is given in Fig. 2. For each food source x_i , its employed bee will explore a temporary position denoted as v_i . The position v_i is a copy of the food source with one randomly selected dimension d to be changed [3]. However, the updating equations for new positions are different for different types of colonies. Specifically, for basic colonies, one randomly selected neighbor k is used to generate new positions which could be formulated as follows:

$$v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd}), \tag{10}$$

where ϕ_{id} is a real number randomly selected from interval $[-1,1]$. Although k is determined randomly, it has to be different from i . From (10) we can find that as the difference between x_{id} and x_{kd} decreases, the perturbation on the position x_{id} gets decreased, too. Thus, as the search approaches the optimum solution in the search space, the step length is adaptively reduced.

For the synthetic colony, the updating equation becomes

$$v_{id} = x_{id} + \phi_{id} \cdot (x_{id} - x_{kd}) + \varphi_{id} \cdot (x_{id} - gbest_d), \tag{11}$$

where ϕ_{id} and φ_{id} are two different random numbers distributed uniformly over $[-1, 1]$, and k is the same as in (10).

Table 6 EPSILON. Mean and standard deviation

	dMOABC	MOEAD	SMPSO	GDE3
UF1	1.90e - 02 _{3.4e-03}	5.72e - 03 _{1.1e-03}	1.20e - 01 _{1.9e-02}	1.56e - 01 _{2.4e-02}
UF2	1.84e - 02 _{6.1e-03}	5.35e - 02 _{1.3e-02}	6.72e - 02 _{8.3e-03}	8.40e - 02 _{2.2e-02}
UF3	1.66e - 01 _{2.9e-02}	3.40e - 02 _{4.7e-02}	1.43e - 01 _{2.9e-02}	3.27e - 01 _{5.3e-02}
UF4	4.19e - 02 _{2.5e-03}	7.11e - 02 _{7.5e-03}	6.37e - 02 _{9.9e-03}	3.86e - 02 _{2.1e-03}
UF5	1.48e - 01 _{2.1e-02}	4.88e - 01 _{1.1e-01}	1.20e + 00 _{3.9e-01}	4.93e - 01 _{5.7e-02}
UF6	1.25e - 01 _{4.1e-02}	4.26e - 01 _{2.6e-01}	5.06e - 01 _{9.4e-02}	3.99e - 01 _{1.2e-01}
UF7	5.63e - 02 _{1.3e-02}	5.28e - 02 _{1.6e-01}	8.44e - 02 _{1.9e-02}	1.36e - 01 _{3.2e-03}
UF8	5.60e - 01 _{1.4e-01}	8.74e - 01 _{9.3e-02}	6.73e - 01 _{1.3e-01}	7.56e - 01 _{3.5e-02}
UF9	4.69e - 01 _{9.8e-02}	8.26e - 01 _{5.7e-02}	6.55e - 01 _{8.6e-02}	3.82e - 01 _{2.2e-01}
UF10	4.83e - 01 _{9.1e-02}	8.68e - 01 _{7.8e-02}	7.96e - 01 _{8.6e-02}	9.64e - 01 _{1.4e-01}
	AbYSS	CellDE	IBEA	MOCeII
UF1	1.78e - 01 _{6.5e-02}	1.18e - 01 _{5.6e-02}	1.93e - 01 _{4.1e-02}	2.51e - 01 _{9.9e-02}
UF2	1.27e - 01 _{4.8e-02}	5.22e - 02 _{1.0e-02}	1.23e - 01 _{4.4e-02}	1.53e - 01 _{5.0e-02}
UF3	4.12e - 01 _{7.0e-02}	1.94e - 01 _{3.8e-02}	4.66e - 01 _{7.7e-02}	5.37e - 01 _{8.7e-02}
UF4	6.32e - 02 _{1.0e-02}	5.34e - 02 _{6.7e-03}	1.04e - 01 _{1.2e-02}	4.75e - 02 _{5.8e-03}
UF5	4.54e - 01 _{1.1e-01}	8.47e - 01 _{1.7e-01}	5.69e - 01 _{2.0e-01}	7.06e - 01 _{2.0e-01}
UF6	5.07e - 01 _{1.5e-01}	4.31e - 01 _{1.5e-01}	5.23e - 01 _{1.8e-01}	6.76e - 01 _{2.1e-01}
UF7	5.82e - 01 _{2.0e-01}	1.25e - 01 _{5.5e-02}	4.05e - 01 _{3.0e-01}	4.65e - 01 _{3.1e-01}
UF8	4.23e - 01 _{1.9e-01}	6.89e - 01 _{1.9e-01}	6.90e - 01 _{7.4e-02}	6.69e - 01 _{1.8e-01}
UF9	5.21e - 01 _{8.2e-02}	9.00e - 01 _{2.1e-01}	4.19e - 01 _{1.1e-01}	6.07e - 01 _{1.2e-01}
UF10	8.01e - 01 _{1.1e-01}	1.53e + 00 _{3.1e-01}	8.91e - 01 _{1.1e-01}	9.52e - 01 _{1.1e-01}
	OMOPSO	NSGAI	PAES	SPEA2
UF1	1.59e - 01 _{2.6e-02}	1.69e - 01 _{4.7e-02}	3.57e - 01 _{1.6e-01}	1.88e - 01 _{4.0e-02}
UF2	5.37e - 02 _{5.8e-03}	9.64e - 02 _{2.5e-02}	1.59e - 01 _{1.1e-01}	1.25e - 01 _{4.7e-02}
UF3	8.80e - 02 _{4.1e-02}	3.31e - 01 _{8.2e-02}	6.29e - 01 _{1.1e-01}	3.93e - 01 _{7.8e-02}
UF4	5.86e - 02 _{6.6e-03}	4.92e - 02 _{1.9e-03}	1.63e - 01 _{9.0e-02}	4.31e - 02 _{1.5e-03}
UF5	6.86e - 01 _{3.9e-01}	4.27e - 01 _{1.1e-01}	7.68e - 01 _{2.1e-01}	4.81e - 01 _{1.4e-01}
UF6	4.35e - 01 _{1.6e-01}	4.35e - 01 _{1.7e-01}	8.76e - 01 _{2.7e-01}	4.21e - 01 _{1.3e-01}
UF7	1.13e - 01 _{1.3e-02}	3.95e - 01 _{2.6e-01}	7.43e - 01 _{2.5e-01}	3.91e - 01 _{2.6e-01}
UF8	7.78e - 01 _{1.3e-01}	5.16e - 01 _{2.3e-01}	7.49e - 01 _{1.0e-01}	7.17e - 01 _{1.2e-01}
UF9	9.30e - 01 _{1.5e-01}	4.26e - 01 _{1.4e-01}	5.35e - 01 _{1.0e-01}	3.42e - 01 _{1.4e-01}
UF10	1.86e + 00 _{4.8e-01}	9.06e - 01 _{1.1e-01}	8.74e - 01 _{1.0e-01}	9.15e - 01 _{9.0e-02}

As mentioned before, *gbest* is called the global best food source. Equation 11 uses three parts to modify one dimension of the food sources which is analogous to the idea introduced in literature [49]. By introducing *gbest* position, it will attract the whole colony into a "better" search region and this operation will also enable a precise local search around the neighborhood that *gbest* is located at. Note that if the value produced by these operations overflows, then it will be shifted onto the bound value.

After the new position for a food source is generated, its objective values are then evaluated. If the new position \mathbf{v}_i can dominate the old one, then this food source will be replaced by the new position vector. If not, the update is deemed as unsuccessful and trial value of this food source will be incremented by one. If \mathbf{v}_i and \mathbf{x}_i are non-dominated with each other, then this trial is also successful if \mathbf{v}_i could be added into the archive.

If a trial for a food source is successful, then its new food source \mathbf{v}_i will be used to update the global best food source *gbest*. The logic for function $gbest = update_gbest(gbest, \mathbf{v}_i)$ (see Fig. 2) is as follows:

If (\mathbf{v}_i dominates *gbest*)
 $gbest = \mathbf{v}_i$

Else If (\mathbf{v}_i and *gbest* are non-dominated with each other)
If (The grid-location count of \mathbf{v}_i is smaller than that of *gbest*)

$$gbest = \mathbf{v}_i$$

End If

End If

After all foods sources have been updated, their fitness values are calculated through function *calculateFitness()* (see Fig. 2). In this paper, the method for calculating fitness value is the same as SPEA2 [58]. Given an individual *i*, the fitness value *F(i)* contains two parts: raw fitness value *R(i)* and density information *D(i)*, namely, $F(i) = R(i) + D(i)$. The raw fitness is determined by the strengths of its dominators in both archive and colony, and it could avoid the situation that individuals dominated by the same archive members have identical fitness values. Density information is incorporated to discriminate between individuals having identical raw fitness values. Details of calculating this kind of fitness value are available in literature [58]. It is important to note that fitness in SPEA2 is to be minimized.

Table 7 EPSILON. Median and IQR

	dMOABC	MOEAD	SMPSO	GDE3
UF1	1.81e - 02 _{4.2e-03}	5.87e - 03 _{1.3e-03}	1.20e - 01 _{2.3e-02}	1.63e - 01 _{1.5e-02}
UF2	1.69e - 02 _{5.5e-03}	5.09e - 02 _{1.9e-02}	6.77e - 02 _{1.3e-02}	7.55e - 02 _{3.4e-02}
UF3	1.68e - 01 _{4.3e-02}	1.74e - 02 _{2.5e-02}	1.52e - 01 _{3.0e-02}	3.16e - 01 _{9.0e-02}
UF4	4.17e - 02 _{3.6e-03}	6.98e - 02 _{9.1e-03}	6.06e - 02 _{1.6e-02}	3.80e - 02 _{1.8e-03}
UF5	1.48e - 01 _{2.6e-02}	4.75e - 01 _{8.0e-02}	1.16e + 00 _{6.7e-01}	5.00e - 01 _{0.0e+00}
UF6	1.12e - 01 _{3.4e-02}	3.16e - 01 _{2.5e-01}	4.92e - 01 _{1.8e-01}	5.00e - 01 _{2.5e-01}
UF7	5.81e - 02 _{1.8e-02}	1.81e - 02 _{7.8e-03}	8.81e - 02 _{3.8e-02}	1.37e - 01 _{4.6e-03}
UF8	5.69e - 01 _{2.5e-01}	9.00e - 01 _{1.4e-01}	7.18e - 01 _{2.1e-02}	7.46e - 01 _{3.5e-02}
UF9	4.78e - 01 _{1.2e-01}	8.46e - 01 _{6.7e-02}	6.71e - 01 _{8.9e-02}	2.56e - 01 _{3.6e-01}
UF10	4.75e - 01 _{1.2e-01}	8.58e - 01 _{1.3e-01}	7.59e - 01 _{1.7e-01}	9.99e - 01 _{2.6e-01}
	AbYSS	CellDE	IBEA	MOCcell
UF1	1.35e - 01 _{1.2e-01}	1.12e - 01 _{8.5e-02}	1.74e - 01 _{6.0e-02}	2.48e - 01 _{1.3e-01}
UF2	1.12e - 01 _{7.2e-02}	5.31e - 02 _{1.6e-02}	1.14e - 01 _{4.3e-02}	1.42e - 01 _{8.0e-02}
UF3	4.11e - 01 _{1.2e-01}	1.80e - 01 _{4.8e-02}	4.54e - 01 _{1.0e-01}	5.43e - 01 _{1.6e-01}
UF4	6.32e - 02 _{1.3e-02}	5.20e - 02 _{8.1e-03}	1.04e - 01 _{1.7e-02}	4.63e - 02 _{7.1e-03}
UF5	4.21e - 01 _{1.7e-01}	8.69e - 01 _{2.5e-01}	5.37e - 01 _{3.1e-01}	7.25e - 01 _{2.8e-01}
UF6	4.91e - 01 _{8.9e-02}	4.12e - 01 _{2.1e-01}	4.90e - 01 _{2.0e-01}	6.76e - 01 _{4.3e-01}
UF7	6.56e - 01 _{8.2e-02}	1.06e - 01 _{3.3e-02}	2.06e - 01 _{5.9e-01}	5.47e - 01 _{6.0e-01}
UF8	2.89e - 01 _{3.3e-01}	6.18e - 01 _{2.7e-01}	7.09e - 01 _{2.1e-03}	7.40e - 01 _{7.1e-02}
UF9	5.08e - 01 _{6.0e-02}	8.81e - 01 _{2.1e-01}	4.57e - 01 _{4.2e-02}	6.06e - 01 _{1.4e-01}
UF10	8.28e - 01 _{1.3e-01}	1.46e + 00 _{3.7e-01}	9.56e - 01 _{2.1e-01}	9.40e - 01 _{1.3e-01}
	OMOPSO	NSGAI	PAES	SPEA2
UF1	1.67e - 01 _{5.3e-02}	1.52e - 01 _{3.1e-02}	3.20e - 01 _{2.7e-01}	1.76e - 01 _{4.9e-02}
UF2	5.26e - 02 _{8.7e-03}	9.29e - 02 _{3.0e-02}	1.26e - 01 _{4.1e-02}	1.16e - 01 _{4.8e-02}
UF3	8.38e - 02 _{6.7e-02}	3.08e - 01 _{1.4e-01}	6.25e - 01 _{1.6e-01}	3.92e - 01 _{1.1e-01}
UF4	5.78e - 02 _{9.7e-03}	4.88e - 02 _{3.2e-03}	1.36e - 01 _{3.4e-02}	4.26e - 02 _{2.2e-03}
UF5	5.40e - 01 _{4.0e-01}	3.97e - 01 _{1.3e-01}	7.76e - 01 _{3.5e-01}	4.26e - 01 _{1.9e-01}
UF6	4.21e - 01 _{3.4e-01}	4.15e - 01 _{2.2e-01}	8.70e - 01 _{2.8e-01}	4.17e - 01 _{1.5e-01}
UF7	1.15e - 01 _{1.1e-02}	3.99e - 01 _{5.3e-01}	8.11e - 01 _{2.2e-01}	3.60e - 01 _{4.9e-01}
UF8	8.18e - 01 _{1.7e-01}	5.15e - 01 _{4.4e-01}	7.14e - 01 _{8.7e-03}	7.40e - 01 _{3.7e-02}
UF9	8.94e - 01 _{1.5e-01}	5.01e - 01 _{2.7e-01}	5.01e - 01 _{1.1e-01}	3.42e - 01 _{2.7e-01}
UF10	1.73e + 00 _{7.9e-01}	8.76e - 01 _{1.6e-01}	8.90e - 01 _{1.6e-01}	9.47e - 01 _{1.2e-01}

3.5 Send onlooker bees

The pseudo code of *Send_Onlooker_Bees* is shown in Fig. 3. After all employed bees optimized their food sources, they will come to the hive to share information with the onlooker bees about the quality of corresponding food sources, and onlooker bees will then choose one food source to be exploited. For this purpose, the probability for each food source k advertised by the corresponding employed bee will be calculated as follows:

$$prob_k = 1 - \frac{fit(\mathbf{x}_k)}{\sum_{m=1}^{FoodNumber} fit(\mathbf{x}_m)}, \tag{12}$$

where $fit(\mathbf{x}_m)$ returns the fitness value of \mathbf{x}_m . From (12) we can find that a food source with lower fitness value will be assigned with a higher selection probability, and this is because the fitness value used in dMAOBC is to be minimized.

Next, onlooker bees will choose a food source i based on the probability provided by its employed bee by using roulette wheel method. After that, they will randomly select one dimension of that food source and do same as employed bees to improve it. The updating equations for both basic

and synthetic colonies are in accordance with those in employed bees phase given by (10) and (11), respectively. Note again that if the parameter value produced by those operations exceeds its predetermined limit, it will be set to its boundaries.

3.6 Send Scout Bees

The pseudo code of *Send_Scout_Bees* is given in Fig. 4. In this step, the algorithm will find abandoned food sources to replace them with new ones. A food source that can't be improved by its employed or onlooker bee for *limit* cycles will be discarded, and be replaced with a vector which is generated similarly as in the initialization phase (see (9)). The scout bees phase could help the algorithm get rid of food sources that have been trapped in a local optimum. Just as basic ABC, only one scout is allowed in each cycle for each colony in our method.

3.7 Multi-colony model and information exchange

As previously stated, the dMOABC method employs three colonies: two are basic ones and the third one is synthetic. The schematic diagram of our multi-colony model is given

Table 8 IGD. Mean and standard deviation

	dMOABC	MOEAD	SMPSO	GDE3
UF1	2.43e-04 _{2.7e-05}	6.90e-05 _{4.2e-06}	2.18e-03 _{3.1e-04}	2.69e-03 _{4.3e-04}
UF2	2.53e-04 _{5.6e-05}	4.55e-04 _{1.8e-04}	9.95e-04 _{1.1e-04}	1.19e-03 _{3.7e-04}
UF3	2.44e-03 _{4.7e-04}	3.81e-04 _{5.8e-04}	3.83e-03 _{1.1e-03}	8.21e-03 _{9.3e-04}
UF4	9.97e-04 _{1.9e-05}	1.93e-03 _{1.6e-04}	1.65e-03 _{9.1e-05}	1.25e-03 _{1.5e-05}
UF5	2.17e-02 _{3.0e-03}	6.88e-02 _{1.8e-02}	2.42e-01 _{1.0e-01}	8.48e-02 _{1.0e-02}
UF6	2.64e-03 _{1.2e-03}	8.39e-03 _{7.9e-03}	1.42e-02 _{2.4e-03}	8.68e-03 _{2.8e-03}
UF7	6.19e-04 _{1.9e-04}	8.49e-04 _{3.8e-03}	1.01e-03 _{1.7e-04}	1.71e-03 _{8.1e-05}
UF8	2.88e-03 _{7.8e-04}	6.61e-03 _{1.2e-03}	3.02e-03 _{4.8e-04}	3.61e-03 _{2.9e-04}
UF9	2.44e-03 _{8.4e-04}	6.07e-03 _{3.9e-04}	4.37e-03 _{5.9e-04}	2.98e-03 _{1.6e-03}
UF10	2.12e-03 _{3.2e-04}	6.46e-03 _{8.5e-04}	4.22e-03 _{4.0e-04}	5.69e-03 _{1.8e-03}
	AbYSS	CellDE	IBEA	MOCcell
UF1	3.83e-03 _{1.8e-03}	1.76e-03 _{1.1e-03}	4.16e-03 _{7.5e-04}	5.40e-03 _{2.1e-03}
UF2	1.98e-03 _{1.1e-03}	6.98e-04 _{8.9e-05}	2.49e-03 _{1.2e-03}	2.84e-03 _{1.4e-03}
UF3	1.08e-02 _{2.0e-03}	3.69e-03 _{8.0e-04}	1.07e-02 _{1.5e-03}	1.12e-02 _{1.2e-03}
UF4	1.59e-03 _{1.3e-04}	1.43e-03 _{6.8e-05}	2.17e-03 _{2.2e-04}	1.41e-03 _{4.1e-05}
UF5	6.56e-02 _{1.5e-02}	1.46e-01 _{2.9e-02}	8.84e-02 _{3.5e-02}	1.06e-01 _{3.5e-02}
UF6	1.35e-02 _{4.4e-03}	1.02e-02 _{3.8e-03}	1.42e-02 _{5.5e-03}	1.85e-02 _{6.7e-03}
UF7	1.19e-02 _{4.7e-03}	1.66e-03 _{1.3e-03}	8.50e-03 _{7.3e-03}	9.74e-03 _{7.9e-03}
UF8	2.29e-03 _{7.7e-04}	5.21e-03 _{1.1e-03}	5.60e-03 _{1.4e-04}	3.39e-03 _{6.5e-04}
UF9	3.31e-03 _{7.3e-04}	8.63e-03 _{2.4e-03}	3.35e-03 _{1.0e-03}	4.26e-03 _{7.5e-04}
UF10	4.60e-03 _{7.7e-04}	1.60e-02 _{4.6e-03}	6.47e-03 _{9.8e-04}	7.35e-03 _{1.5e-03}
	OMOPSO	NSGAI	PAES	SPEA2
UF1	2.63e-03 _{4.2e-04}	3.30e-03 _{9.1e-04}	7.84e-03 _{3.5e-03}	3.88e-03 _{8.9e-04}
UF2	8.04e-04 _{6.3e-05}	1.42e-03 _{3.9e-04}	3.10e-03 _{1.6e-03}	2.13e-03 _{1.1e-03}
UF3	1.86e-03 _{8.5e-04}	6.80e-03 _{1.7e-03}	1.23e-02 _{1.8e-03}	8.13e-03 _{1.9e-03}
UF4	1.56e-03 _{6.7e-05}	1.46e-03 _{2.5e-05}	3.48e-03 _{8.2e-04}	1.36e-03 _{1.0e-05}
UF5	1.40e-01 _{1.0e-01}	5.89e-02 _{1.4e-02}	1.16e-01 _{3.6e-02}	6.70e-02 _{2.0e-02}
UF6	9.87e-03 _{4.9e-03}	1.04e-02 _{4.5e-03}	2.11e-02 _{7.0e-03}	1.05e-02 _{3.8e-03}
UF7	1.26e-03 _{1.3e-04}	7.75e-03 _{5.9e-03}	1.75e-02 _{7.1e-03}	7.66e-03 _{6.1e-03}
UF8	5.05e-03 _{1.1e-03}	2.53e-03 _{7.8e-04}	5.49e-03 _{1.5e-03}	2.85e-03 _{3.6e-04}
UF9	6.96e-03 _{1.6e-03}	2.81e-03 _{8.6e-04}	4.47e-03 _{1.4e-03}	1.82e-03 _{6.8e-04}
UF10	2.16e-02 _{6.8e-03}	4.78e-03 _{1.1e-03}	5.78e-03 _{7.9e-04}	4.14e-03 _{9.6e-04}

in Fig. 5 which shows all possible relationships between colonies and the archive. Line (1) means that non-dominated solutions found by the synthetic colony are added into the archive. Basic colonies share information with synthetic colony by updating *gbest*, shown as line (2) and line (3). Non-dominated solutions found by two basic colonies are also added to the archive, so line (4) and line (5) exist. We can find line (6) and line (7) are dotted, this is because there exists no direct communication between two basic colonies.

4 Experimental study

This section contains the computational results obtained by the dMOABC algorithm compared to 11 other state-of-the-art multi-objective methods over a set of unconstrained test problems introduced in CEC09 [53].

4.1 Performance metrics

In this work, we use four indicators to provide quantitative assessments over all algorithms, and they are: Hypervolume (HV) [59], SPREAD [10], EPSILON [14], and IGD [14].

These metrics are computed by jMetal¹, an object-oriented Java-based framework for multi-objective optimization with meta-heuristics, which was developed by J.J. Durillo and A.J. Nebro [13, 14]. It should be noted here that all indicators but Hypervolume are to be minimized.

4.2 Benchmarks and experimental setup

In this section, several test functions from the CEC09 [53] are used in order to compare the performance of dMOABC against other multi-objective algorithms. The CEC09 benchmark presents a set of complex test problems, and ten unconstrained ones (UF1-UF10) are selected for the purpose of performance comparison. The mathematical representation of these test problems are given in Table 1. The UF1-UF7 are two objective test problems while UF8-UF10 are three objective ones. The Pareto fronts of these test problems have different characteristics (e.g. a part of them are convex while the other ones are concave or some of them are continuous while the other ones are discontinuous) [3].

¹url=<http://jmetal.sourceforge.net/>

Table 9 IGD. Median and IQR

	dMOABC	MOEAD	SMPSO	GDE3
UF1	2.42e-04	6.92e-05	2.12e-03	2.75e-03
UF2	2.38e-04	4.11e-04	9.85e-04	1.04e-03
UF3	2.41e-03	1.60e-04	3.67e-03	8.35e-03
UF4	1.00e-03	1.91e-03	1.66e-03	1.24e-03
UF5	2.15e-02	6.26e-02	1.99e-01	8.72e-02
UF6	2.25e-03	6.45e-03	1.45e-02	1.05e-02
UF7	5.83e-04	1.09e-04	1.00e-02	1.72e-03
UF8	2.64e-03	6.02e-03	3.16e-02	3.56e-03
UF9	2.29e-03	6.19e-03	4.52e-03	2.08e-03
UF10	2.06e-03	6.21e-03	4.25e-03	5.66e-03
	AbYSS	CellIDE	IBEA	MOCcell
UF1	2.65e-03	1.56e-03	4.05e-03	4.79e-03
UF2	1.50e-03	6.93e-04	2.01e-03	2.53e-03
UF3	1.17e-02	3.43e-03	1.11e-02	1.13e-02
UF4	1.56e-03	1.42e-03	2.20e-03	1.40e-03
UF5	6.31e-02	1.46e-01	7.90e-02	1.11e-01
UF6	1.29e-02	1.03e-02	1.24e-02	1.67e-02
UF7	1.37e-02	1.23e-03	3.44e-03	1.01e-02
UF8	2.02e-03	5.07e-03	5.64e-03	3.58e-03
UF9	3.21e-03	8.44e-03	3.66e-03	4.26e-03
UF10	4.68e-03	1.60e-02	6.52e-03	7.38e-03
	OMOPSO	NSGAI	PAES	SPEA2
UF1	2.54e-03	3.01e-03	7.11e-03	3.89e-03
UF2	8.16e-04	1.34e-03	2.64e-03	1.78e-03
UF3	1.68e-03	6.30e-03	1.22e-02	8.07e-03
UF4	1.56e-03	1.46e-03	3.30e-03	1.36e-03
UF5	9.98e-02	5.66e-02	1.15e-01	6.52e-02
UF6	1.01e-02	9.34e-03	1.96e-02	1.05e-02
UF7	1.26e-03	7.73e-03	1.89e-02	5.91e-03
UF8	4.81e-03	2.35e-03	4.94e-03	2.93e-03
UF9	6.44e-03	3.23e-03	4.03e-03	1.49e-03
UF10	1.93e-02	4.41e-03	5.66e-03	3.64e-03

To observe the performance of the proposed dMOABC method in solving multi-objective problems, its performance is compared with MOEAD [52], SMPSO [35], GDE3 [29], AbYSS [37], CellIDE [15], IBEA [57], MOCcell [36], OMOPSO [42], NSGAI [11], PAES [27] and SPEA2 [58]. In all experiments, the maximum number of function evaluations is set at 300,000 for each problem and each algorithm. The maximum number of final solutions for computing indicators is 100 for both two and three objectives problems. Each algorithm is evaluated independently 30 times for each test problem. The dMOABC algorithm is tested with a colony of size 10, so *MaxIterations* is set at $300,000/10 = 30,000$. Since there are three colonies, each colony iterates $30,000/3 = 10,000$ cycles. According to our experience, we set the values of control parameter *limit* and *depth* at 100 and 6, respectively. The performance of the dMOABC is evaluated under this configuration over all test problems. dMOABC algorithm is implemented by our Java code, and other algorithms are executed by running jMetal software package.

We turn now to analyze the computed fronts in terms of the diversity through the use of SPREAD indicator (Tables 4 and 5). In this case, the first three best algorithms are distributed among MOEAD, SPEA2, SMPSO and dMOABC. Both MOEAD and SPEA2 have obtained the best values for

this indicator on three evaluated problems: UF1, UF3 and UF7 for MOEAD; UF4, UF8 and UF9 for SPEA2. Another algorithm obtaining good results in this indicator has been dMOABC: it has computed the fronts with the best values of the indicator in two out of the ten evaluated problems, plus one second best value on test function UF1. For this indicator, no algorithm has obtained obviously better results than its competitors.

4.3 Performance analysis

The computational results on UF1-UF10 obtained by all algorithms are shown in Tables 2-9. They include the mean and standard deviation, as well as the median and interquartile range (IQR) of all the independent runs for the indicators HV, SPREAD, EPSILON and IGD, respectively. To ease the analysis of these tables, some cells have a gray colored background in each row; particularly, there are two different gray levels: a darker one, pointing out the algorithm obtaining the best value of the indicator, and a lighter one, highlighting the algorithm obtaining the second best value of the indicator [14].

We start by describing the values obtained in the HV indicator. For this indicator, the higher the value, the better the computed fronts. Thus, attending to Tables 2 and 3, we

Fig. 6 Boxplots of the HV obtained by the different algorithms in the evaluated problems

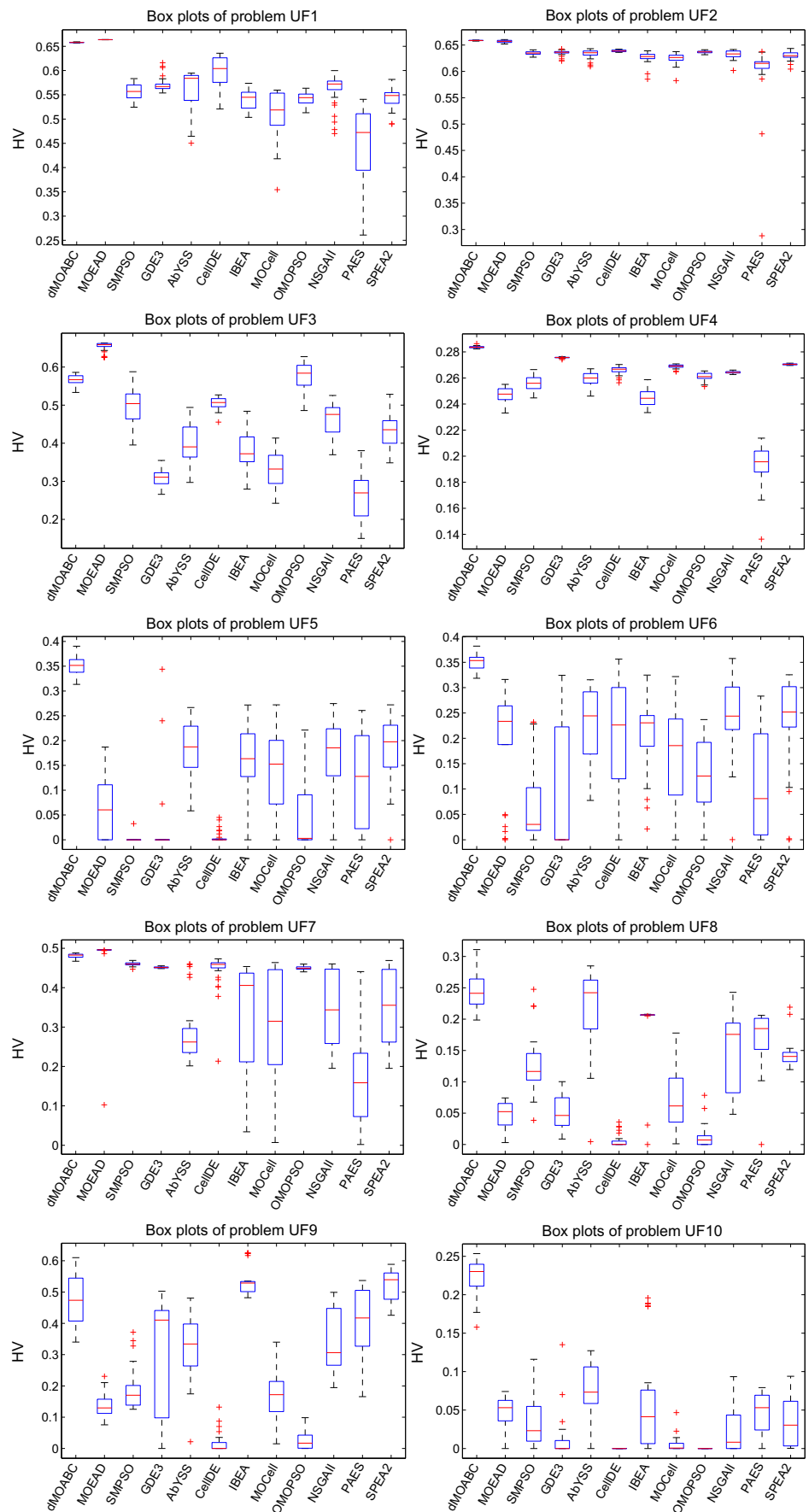


Fig. 7 Boxplots of the SPREAD obtained by the different algorithms in the evaluated problems

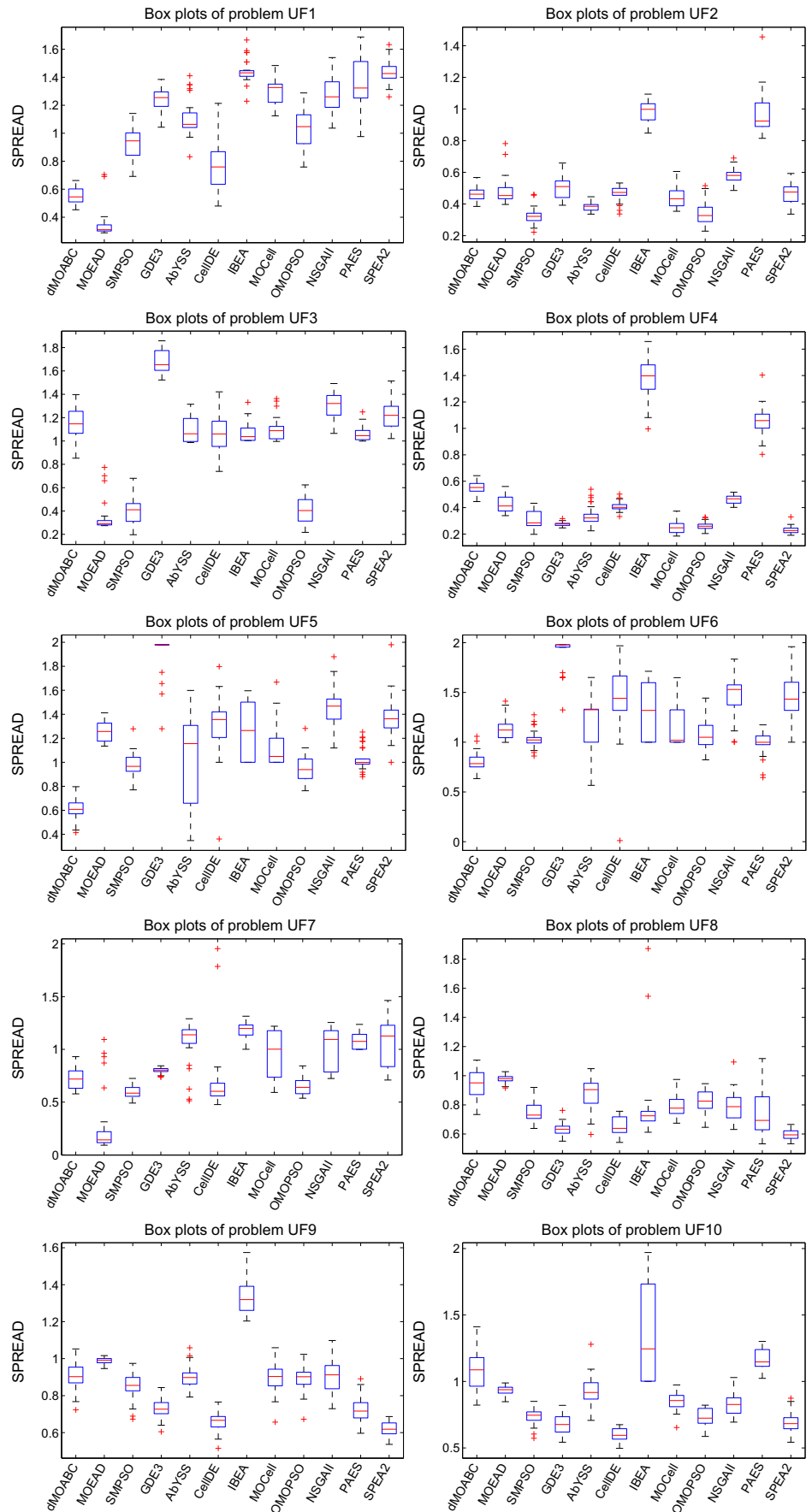


Fig. 8 Boxplots of the EPSILON obtained by the different algorithms in the evaluated problems

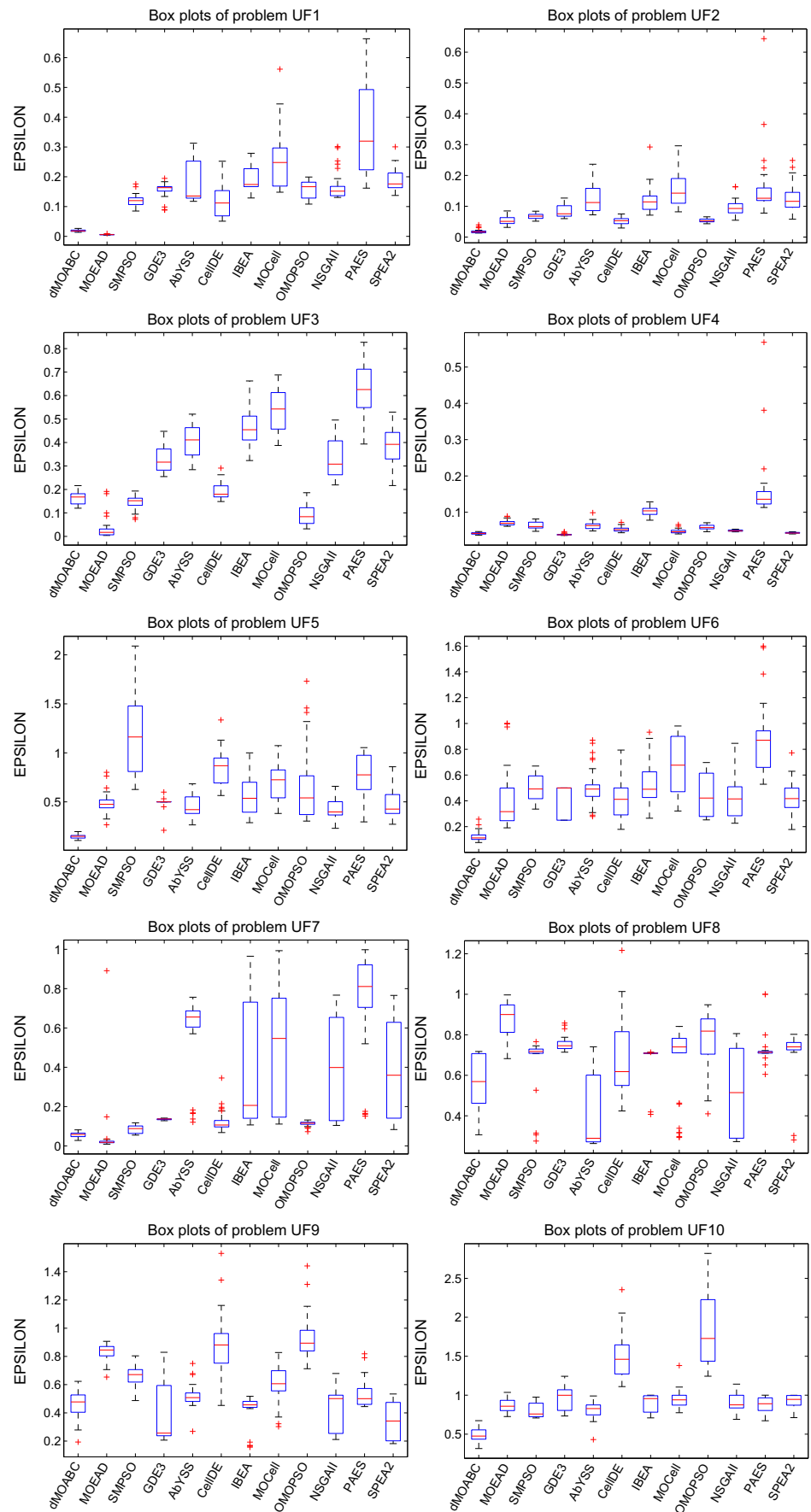


Fig. 9 Boxplots of the IGD obtained by the different algorithms in the evaluated problems

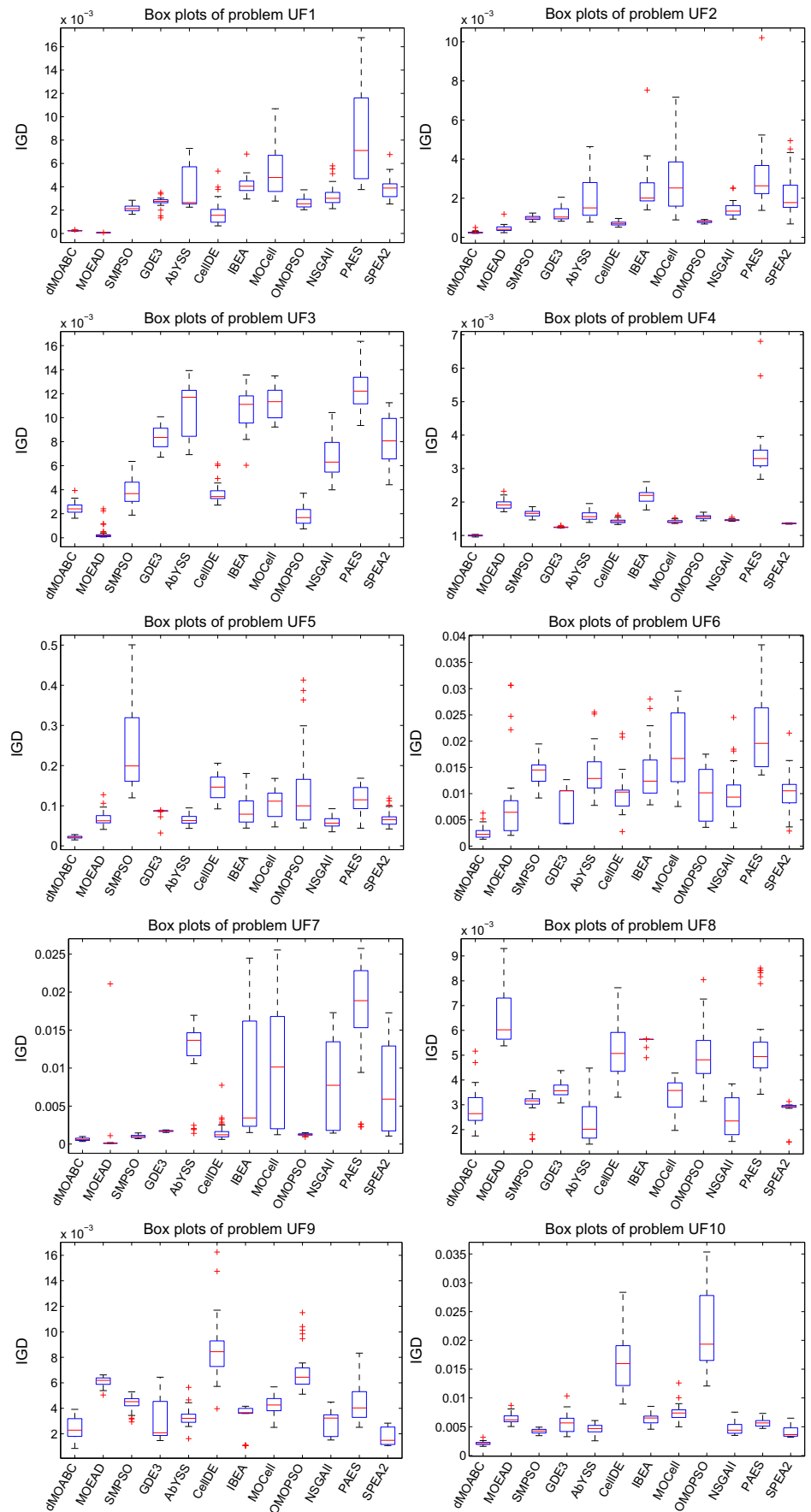


Fig. 10 The best Pareto fronts obtained by the dMOABC algorithm on the test problems UF1-UF10 (The blue and red lines represent the true and approximated Pareto fronts, respectively)

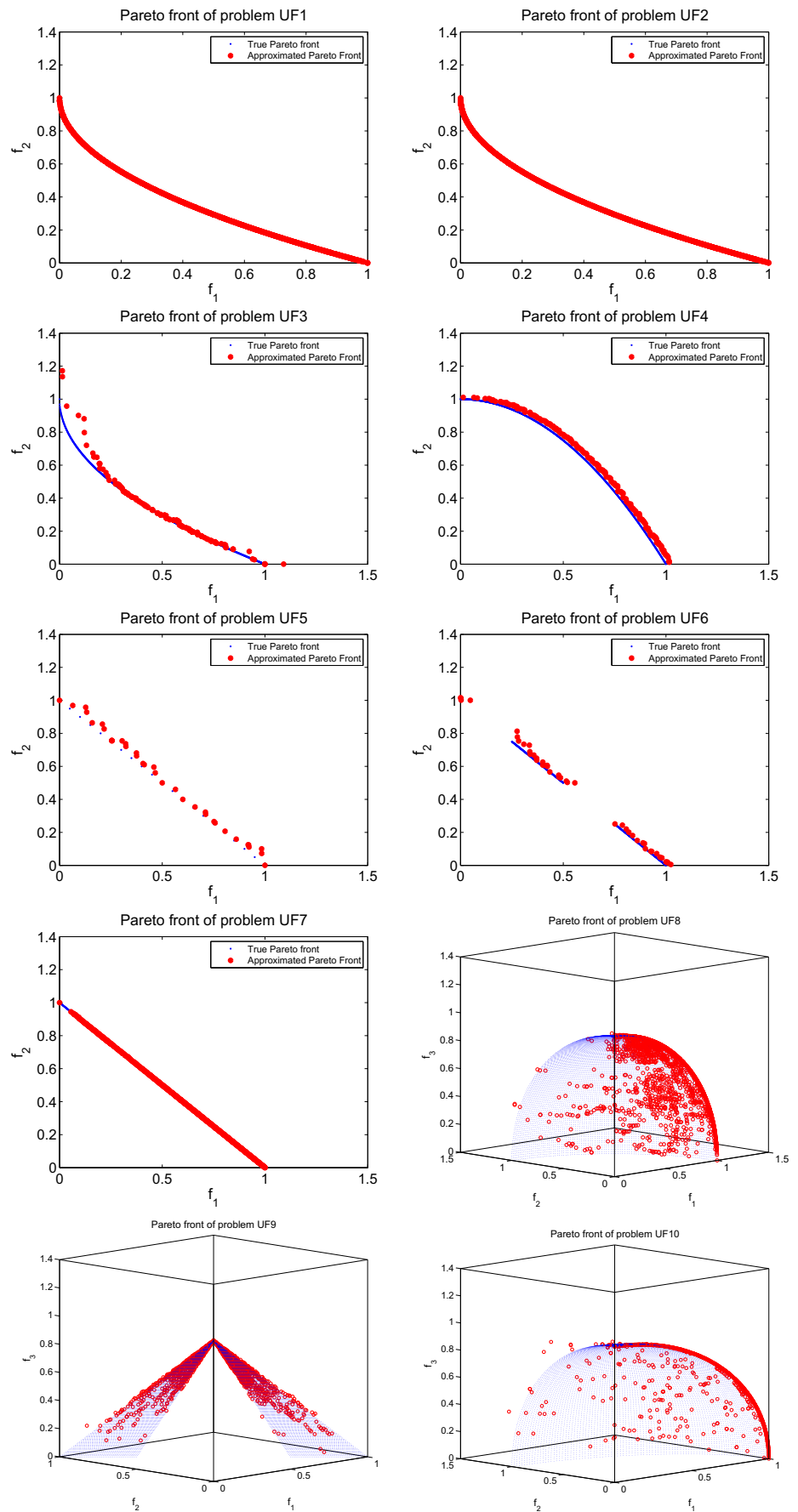


Table 10 Average Rankings of the algorithms by applying Friedman test

Algorithm	Ranking(HV)	Ranking(SPREAD)	Ranking(EPSILON)	Ranking(IGD)
dMOABC	11.4	6.4	2.2	1.7
MOEAD	7.6	6.4	5.1	5.3
SMPSO	5.5	3.5	6.0	6.3
GDE3	6.0	7.0	5.4	5.4
AbYSS	7.3	6.9	6.8	6.6
CellIDE	5.7	4.9	6.5	6.8
IBEA	6.4	9.9	8.1	9.2
MOCcell	4.3	6.2	8.8	8.9
OMOPSO	5.2	4.2	7.3	6.7
NSGAI	7.3	9.1	5.5	5.2
PAES	3.9	7.2	10.3	10.7
SPEA2	7.5	6.3	6.0	5.2

see that the best indicator values are archived by dMOABC algorithm, and the second best is MOEAD. dMOABC has obtained the best or second best values in this indicator for all the problems but UF3 and UF9; MOEAD has computed the best fronts regarding to this indicator in four out of the ten evaluated problems. AbYSS is also a competitive algorithm which has gotten the second best values on three functions (UF5, UF8 and UF10) in terms of the mean value.

Next, we pay attention to the EPSILON values (Tables 6 and 7). Regarding this indicator, dMOABC has been also the best algorithm; after it MOEAD has obtained the best values in three out of the ten problems in terms of the mean value. Another promising algorithm has been GDE3, which has archived the best value in one out of the ten problems and the second best values of the indicator in two cases (see Table 6). For the median values regarding this indicator, similar conclusion could be reached.

Finally, we compare algorithms in terms of IGD indicator (Tables 8 and 9). Definitely, in this case, dMOABC has performed the best. Then it comes to MOEAD, archiving the best or second best values regarding to this indicator in five out of the ten evaluated problems in terms of both mean and median values.

The above conclusions are drawn just based on the number of the best or the second best results obtained by each algorithm. Average rankings of the algorithms will be given by using statistical test shown in Subsection 4.4. More detailed information can be obtained if we display the results by using boxplots, which constitutes a useful way of depicting groups of numerical data. In this case, the boxplot representing the distribution of values for the all

indicators in the comparison carried out are showed in Fig. 6–9. Some interesting facts can be worked out by observing the figure. For example, in problems UF4, UF6 and UF10, we observe that dMOABC has clearly outperformed to the other algorithms in terms of HV indicator. Regarding indicator IGD, dMOABC has been the most salient on functions UF1, UF4, UF5, UF6, UF10. From these figures, we find dMOABC produces more stable results in most cases, that is to say, our algorithm has better robustness than other algorithms.

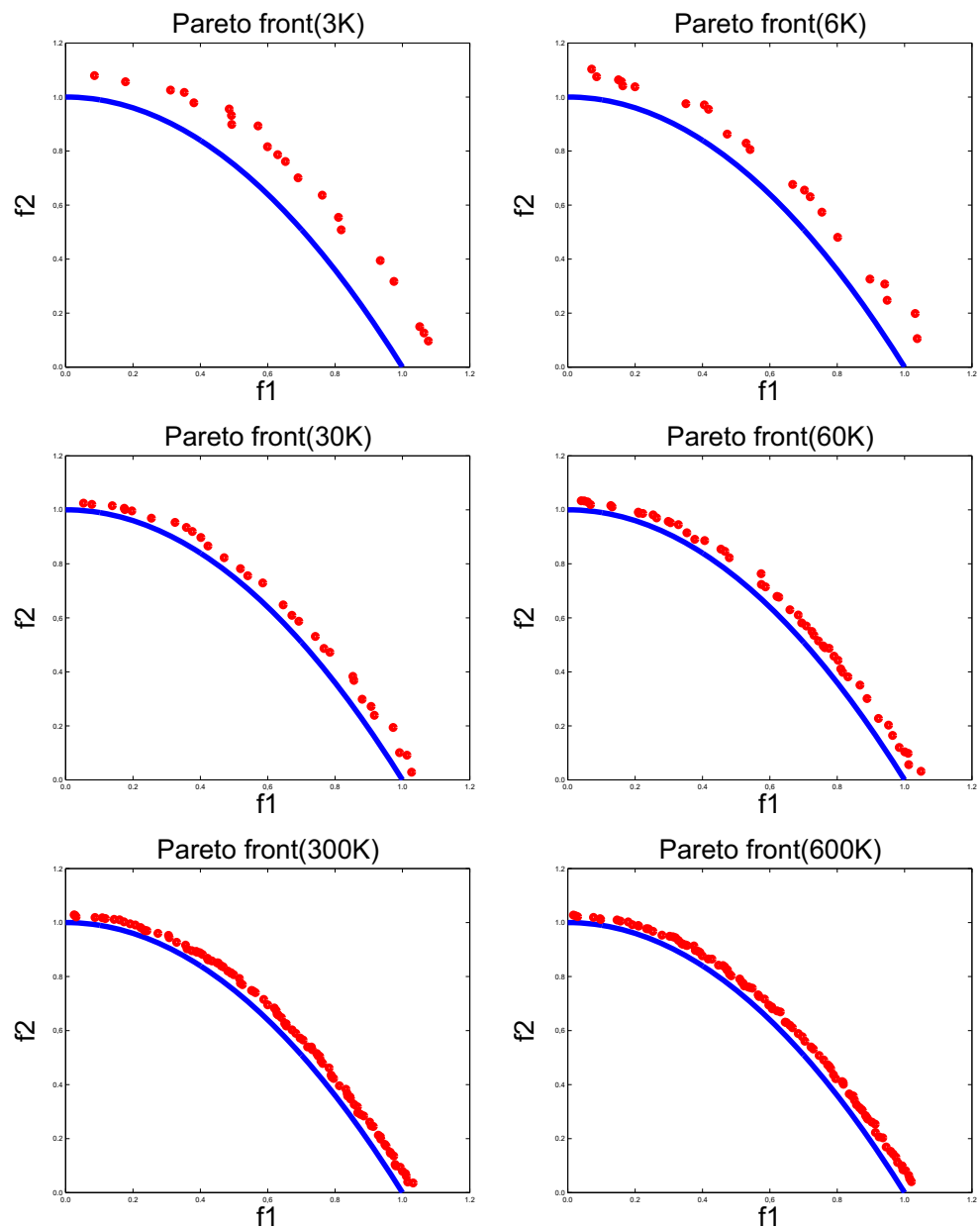
4.4 Nonparametric statistical test

One of the most frequent situations where the use of statistical procedures is requested is in the joint analysis of the results achieved by various algorithms [12]. To determine whether the differences between the 12 involved algorithms introduced in Subsection 4.2 are significant or not, this section will perform multiple comparisons using Friedman test [12] which is one of the commonly used nonparametric statistical test procedures.

Average rankings of each algorithm for each indicator obtained by applying Friedman test are shown in Table 10. These rankings are also computed by jMetal software. The Friedman test assumes that the lower the values of the indicators the better. This is true in all the indicators but the hypervolume. It could be seen from Table 10 that the dMOABC is significantly better than other algorithms in terms of indicators HV, EPSILON and IGD. As for indicator SPREAD, SMPSO performs the best, and dMOABC is comparable to MOEAD algorithm.

Summarizing, according to the problems, parameter settings and the quality indicators used, dMOABC has been the most salient algorithm in our study.

Fig. 11 The convergence behavior of dMOABC throughout maximal number of function evaluations (The blue and red lines represent the true and approximated Pareto fronts, respectively)



4.5 Distribution and convergence analysis

For each test function, the approximated fronts computed by dMOABC, together with the true fronts, are plotted in Fig. 10. In this figure, the computed front is a combination of all archive members returned by dMOABC over 30 independent runs. It is shown by this figure that the Pareto fronts produced by dMOABC algorithm over the UF1 and UF2 test problems not only have good convergence but also has appropriate distribution along the true Pareto front in objective space.

For the UF3 test problem, the approximated fronts distribute uniformly in the middle part of the true front, but

are not evenly distributed in the top-left and bottom-right corners of the Pareto front. Figure 10 shows that dMOABC algorithm has the ability to produce the uniformly distributed Pareto front on UF4 test function.

It seems that the UF5 is a hard problem to be solved since it has a discontinuous Pareto front. Most of the algorithms have difficulties in approximating to its true Pareto front [3]. It is shown by Fig. 10 that the produced Pareto front by dMOABC is uniformly distributed along the true one.

The Pareto front of UF6 is also discontinuous. Hence, an optimization algorithm needs to pay more attention to the Pareto front and moves the archive members to the parts of

solution space which contain the members of Pareto fronts. The results show that most of the algorithms have difficulty in optimizing this type of test problems [3]. Figure 10 also shows that our algorithm could produce nearly uniformly distributed fronts for the problem.

For problem UF7, it could be seen from Fig. 10 that the top-left corner of the Pareto front is not successfully covered by the dMOABC algorithm. Hence, the value of indicator SPREAD increases over the UF7 test problem.

The UF8 is the first three objective test problem investigated in this paper. Usually, the complexity of multi-objective problems increases by the number of objectives to be optimized. The quality of the approximated Pareto front is shown in Fig. 10. It is apparent from the results that the dMOABC produces a set of solution points which have an appropriate distribution in the 3 dimensional objective space.

For the test problem UF9, Fig. 10 shows that the dMOABC produces a set of non-dominated points which covers a large part of the objective space. However, small parts of the objective space are not covered by the approximated Pareto front.

It seems that UF10 is also a hard problem to be solved. Figure 10 shows that the approximated Pareto front found by the dMOABC algorithm distributes nearly uniformly, but with a small number of non-dominated solutions.

Next, the behavior of the dMOABC in converging to the optimal Pareto front is studied. By setting the maximal number of function evaluations at 3K, 6K, 30K, 60K, 300K and 600K, we plot the approximated Pareto front of the UF4 test problem under different situations in Fig. 11. The results show that the dMOABC algorithm continuously improves its performance with the increase of the maximum number of function evaluations. From Fig. 11, we can find that the number of non-dominated solutions continuously increases and the distance between the true and the approximated Pareto front continuously decreases, with function evaluations varying from 3K to 600K. Although the dMOABC rapidly converges to the optimal Pareto front at the first function evaluations, it can produce a few solution points. As the number of function evaluations proceeds, the number of produced solutions increases and better distribution of them is achieved. An algorithm needs to provide appropriate balance between exploration and exploitation and mitigate the hard constriction on the flying trajectories of its individuals in order to avoid premature convergence [3]. The dMOABC algorithm uses some mechanisms to avoid itself being trapped into a local optimum: the use of both neighbor and social knowledge helps flying in the right trajectories toward food sources; the communication between basic and synthetic colonies also enables the algorithm to explore other optimal positions produced by basic colonies so as to avoid premature convergence.

4.6 Computational complexity

The computational complexity is an important factor when evaluating an algorithm. Although our dMOABC achieves good performance on CEC09 test functions, its space complexity is great since three colonies are used. The larger the number of colonies is, the greater the space complexity will be. However, we can find that there exists potential parallelism among the three colonies. That is to say, the dMOABC algorithm can be easily modified to a parallel algorithm with three processors, and each one for each colony. Hence, the running time of the dMOABC will decrease significantly.

5 Conclusion

In this paper, we have presented a new multi-objective artificial bee colony algorithm by dividing the searching space. In our proposed method, three colonies are used: two of them are basic colonies and the third one is called the synthetic colony. These colonies search in different regions of the searching space and share information with each other, and the aim for that is to avoid being attracted into a local optimum. An external archive is adopted to store non-dominated solutions found by the algorithm, and the diversity of archived solutions is controlled by a self-adaptive grid. Besides neighbor information, social information shared by the external archive, called *gbest*, is also used to guide the flying of both employed and onlooker bees in the synthetic colony. Compared to other state-of-the-art algorithms considered in this work, the experimental study shows that our method achieves the first rank in terms of indicators HV, EPSILON and IGD. Some improvements and extensions are currently being investigated by the authors, including using ϵ dominance method to maintain the archive and extending it to be a parallel algorithm, or modifying it for handling multi-objective optimization problems. Practical applications of this algorithm would also be worth studying.

Acknowledgments The authors thank the anonymous reviewers for providing valuable comments to improve this paper, and add special thanks to J.J. Durillo and A.J. Nebro for their open source jMetal software package.

References

1. Agrawal S, Dashora Y, Tiwari MK, Son YJ (2008) Interactive particle swarm: A pareto-adaptive metaheuristic to multiobjective optimization. *IEEE Trans Syst Man Cybern A* 38(2):258–277
2. Akay B, Karaboga D (2012) A modified artificial bee colony algorithm for real-parameter optimization. *Inf Sci* 192(1):120–142

3. Akbari R, Hedayatzadeh R, Ziarati K, Hassanizadeh B (2012) A multi-objective artificial bee colony algorithm. *Swarm Evol Comput* 2:39–52
4. Arsuaga-Rios M, Vega-Rodriguez M, Prieto-Castrillo F (2011) Multi-objective artificial bee colony for scheduling in grid environments. In: *IEEE Symposium on Swarm Intelligence (SIS)*, pp. 1–7. IEEE
5. Atashkari K, NarimanZadeh N, Ghavimi AR, Mahmoodabadi MJ, Aghaienezhad F (2011) Multi-objective optimization of power and heating system based on artificial bee colony. In: *International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*, pp. 64–68
6. Chen CM, Chen YP, Zhang QF (2009) Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization. In: *Proceeding of Congress on Evolutionary Computation, CEC'09*, pp. 209–216. IEEE
7. Coello Coello CA, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. *IEEE Trans Evol Comput* 8(3):256–279
8. Coello Coello CA, Sierra MR (2003) A co-evolutionary multi-objective evolutionary algorithm. *Proceedings of the congress on evolutionary computation*, vol 1. CEC'03, New York, pp 482–489
9. Corne DW, Knowles JD, Oates MJ (2000) The pareto envelope-based selection algorithm for multiobjective optimization. *Proceedings of the parallel problem solving from nature VI*. Springer, pp 839–848
10. Deb K (2001) Multi-objective optimization. *Multi-objective optimization using evolutionary algorithms*, pp. 13–46
11. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Trans Evol Comput* 6(2):182–197
12. Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol Comput* 1(7):3–18
13. Durillo J, Nebro A, Alba E (2010) The jmetal framework for multi-objective optimization: Design and architecture. In: *CEC 2010*, pp. 4138–4325
14. Durillo JJ, Nebro AJ (2011) jmetal: A java framework for multi-objective optimization. *Adv Eng Softw* 42:760–771
15. Durillo JJ, Nebro AJ, Luna F, Alba E (2008) Solving three-objective optimization problems using a new hybrid cellular genetic algorithm. In: Rudolph G, Jensen T, Lucas S, Poloni C, Beume N (eds) *Parallel Problem Solving from Nature—PPSN X*, *Lecture Notes in Computer Science*, vol 5199. Springer, pp 661–670. doi:10.1007/978-3-540-87700-4_66
16. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. Nagoya, Japan, pp 39–43
17. Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: *Proceedings of the fifth international conference on genetic algorithms*, vol. 93, pp. 416–423. San Mateo and California
18. Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. *Comput Oper Res* 39(3):687–697
19. Goh C, Tan K, Liu D, Chiam S (2010) A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *Eur J Oper Res* 202(1):42–54
20. Guliaschi V, Toshev H, Korsemov C (2009) Survey of evolutionary algorithms used in multiobjective optimization. *Problems of Engineering Cybernetics and Robotics*, Bulgarian Academy of Sciences, vol 60
21. Hedayatzadeh R, Hassanizadeh B, Akbari R, Ziarati K (2010) A multi-objective artificial bee colony for optimizing multi-objective problems. In: *The 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 5, pp. 277–281
22. Horn J, Nafpliotis N, Goldberg DE (1994) A niched pareto genetic algorithm for multiobjective optimization. In: *Proceedings of the first IEEE conference on evolutionary computation*, pp. 82–87
23. Huang VL, Suganthan PN, Liang JJ (2006) Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. *Int J Intell Syst* 21(2):209–226
24. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Tech. rep., Engineering Faculty, Computer Engineering Department. Erciyes University
25. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J Glob Optim* 39(3):459–471
26. Kennedy J, Eberhart RC (1995) Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks*. USA, Washington, pp 1942–1948
27. Knowles J, Corne D (1999) The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimization. In: *Proceeding of Congress on Evolutionary Computation, CEC'99*, vol. 1. IEEE
28. Knowles J, Corne D (2000) (1+1)-PAES skeleton program code. <http://www.cs.man.ac.uk/jknowles/multi/paes.c>
29. Kukkonen S, Lampinen J (2009) Performance assessment of generalized differential evolution 3 with a given set of constrained multi-objective test problems. In: *Proceeding of Congress on Evolutionary Computation, CEC'09*, pp. 1943–1950. IEEE
30. Lei DM (2009) Multi-objective intelligent optimization algorithm and applications. Science Press, Beijing
31. Leong WF, Yen GG (2008) Pso-based multiobjective optimization with dynamic population size and adaptive local archives. *IEEE Trans Syst Man Cybern B (Cybernetics)* 38(5):1270–1293
32. Li JQ, Pan QK, Gao KZ (2011) Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems. *Int J Adv Manuf Technol* 55(9-12):1159–1169
33. Liu HL, Li X (2009) The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In: *Proceeding of Congress on Evolutionary Computation, CEC'09*, pp. 1928–1934. IEEE
34. Liu M, Zou X, Chen Y, Wu Z (2009) Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances. In: *Proceeding of Congress on Evolutionary Computation, CEC'09*, pp. 2913–2918. IEEE
35. Nebro AJ, Durillo J, Garcia-Nieto J, Coello Coello C, Luna F, Alba E (2009) SMPSO: A new pso-based metaheuristic for multi-objective optimization. In: *IEEE symposium on computational intelligence in multi-criteria decision-making*, pp. 66–73. IEEE
36. Nebro AJ, Durillo JJ, Luna F, Dorronsoro B, Alba E (2009) Mocell: A cellular genetic algorithm for multiobjective optimization. *Int J Intell Syst* 24(7):726–746
37. Nebro AJ, Luna F, Alba E, Dorronsoro B, Durillo JJ, Beham A (2008) AbYSS: Adapting Scatter Search to Multiobjective Optimization, vol 12
38. Negro FdT, Ortega J, Ros E, Mota S, Paechter B, Martin J (2004) PSFGA: Parallel processing and evolutionary computation for multiobjective optimisation. *Parallel Comput* 30(5):721–739
39. Omkar S, Senthilnath J, Khandelwal R, Naik GN, Gopalakrishnan S (2011) Artificial bee colony (ABC) for multi-objective design optimization of composite structures. *Appl Soft Comput* 11(1):489–499
40. Qi Y, Ma X, Liu F, Jiao L, Sun J, Wu J (2013) MOEA/D with adaptive weight adjustment. *Evol Comput*. doi:10.1162/EVCO.a.00109

41. Schaffer JD (1985) Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the first international conference on genetic algorithm, pp. 93–100. Lawrence Erlbaum
42. Sierra MR, Coello CAC (2005) Improving pso-based multi-objective optimization using crowding, mutation and ϵ -dominance. In: Evolutionary Multi-Criterion Optimization, pp. 505–519. Springer
43. Sierra MR, Coello Coello CA (2006) Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *Int J Comput Intell Res* 2(3):287–308
44. Srinivas N, Deb K (1994) Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evol comput* 2(3):221–248
45. Tan KC, Yang Y, Goh CK (2006) A distributed cooperative coevolutionary algorithm for multiobjective optimization. *Evol Comput IEEE Trans on* 10(5):527–549
46. Tiwari S, Fadel G, Koch P, Deb K (2009) Performance assessment of the hybrid archive-based micro genetic algorithm (AMGA) on the CEC09 test problems. In: Proceeding of Congress on Evolutionary Computation, CEC'09, pp. 1935–1942. IEEE
47. Tripathi PK, Bandyopadhyay S, Pal SK (2007) Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Inf Sci* 177(22):5033–5049
48. Tseng LY, Chen C (2009) Multiple trajectory search for unconstrained/constrained multi-objective optimization. In: Proceeding of Congress on Evolutionary Computation, CEC'09, pp. 1951–1958. IEEE
49. Xiang Y, Peng Y, Zhong Y, Chen Z, Lu X, Zhong X (2014) A particle swarm inspired multi-elitist artificial bee colony algorithm for real-parameter optimization. *Comput Optim Appl* 57(2):493–516
50. Yen GG, Leong WF (2009) Dynamic multiple swarms in multiobjective particle swarm optimization. *IEEE Trans Syst Man Cybern A: (Systems and Humans)* 39(4):890–911
51. Yen GG, Lu H (2003) Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. *Evol Comput IEEE Trans on* 7(3):253–274
52. Zhang Q, Liu W, Li H (2009) The performance of a new version of MOEA/D on CEC09 unconstrained MOP instances. In: IEEE congress on evolutionary computing (CEC), Trondheim, pp. 18–21
53. Zhang Q, Zhou A, Zhao S, Suganthan PN, Liu W, Tiwari S (2009) Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, Special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report
54. Zhao SZ, Suganthan PN (2011) Two-lbest based multi-objective particle swarm optimizer. *Eng Optim* 43(1):1–17
55. Zheng YJ, Chen SY (2013) Cooperative particle swarm optimization for multiobjective transportation planning. *Appl Intell* 39(1):202–216
56. Zhou A, Qu BY, Li H, Zhao SZ, Suganthan PN, Zhang Q (2011) Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm Evol Comput* 1(1):32–49
57. Zitzler E, Künzli S (2004) Indicator-based selection in multiobjective search. In: Proc. 8th International Conference on Parallel Problem Solving from Nature, PPSN VIII, pp. 832–842. Springer
58. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: Improving the strength pareto evolutionary algorithm. Tech. rep., Computer Engineering and Networks Laboratory, Department of Electrical Engineering, Swiss Federal Institute of Technology. ETH Zurich, Switzerland
59. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Trans Evol Comput* 3(4):257–271
60. Zou W, Zhu Y, Chen H, Shen H (2011) A novel multi-objective optimization algorithm based on artificial bee colony. In: Proceedings of the 13th annual conference companion on Genetic and evolutionary computation, pp. 103–104