Swarm and Evolutionary Computation 2 (2012) 39–52

Contents lists available at SciVerse ScienceDirect

Swarm and Evolutionary Computation

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

Regular paper

A multi-objective artificial bee colony algorithm

Reza Akbari [a,*], Ramin Hedayatzadeh [a], Koorush Ziarati [b], Bahareh Hassanizadeh [a]

[a] Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, Iran
[b] Department of Computer Science and Engineering, Shiraz University, Shiraz, Iran

ARTICLE INFO

Article history:
Received 25 October 2010
Received in revised form
11 August 2011
Accepted 11 August 2011
Available online 21 September 2011

Keywords:
Artificial bee colony
Multi-objective optimization
Multi-objective artificial bee colony

ABSTRACT

This work presents a multi-objective optimization method based on the artificial bee colony, called the MOABC, for optimizing problems with multiple objectives. The MOABC uses a grid-based approach to adaptively assess the Pareto front maintained in an external archive. The external archive is used to control the flying behaviours of the individuals and structuring the bee colony. The employed bees adjust their trajectories based on the non-dominated solutions maintained in the external archive. On the other hand, the onlooker bees select the food sources advertised by the employed bees to update their positions. The qualities of these food sources are computed based on the Pareto dominance notion. The scout bees are used by the MOABC to get rid of food sources with poor qualities. The proposed algorithm was evaluated on a set of standard test problems in comparison with other state-of-the-art algorithms. Experimental results indicate that the proposed approach is competitive compared to other algorithms considered in this work.

1. Introduction

Optimization problems with more than one objective function are common in many engineering problems. Our goal is to find a set of solutions that represent a trade-off among the objectives; because in such areas there is no single solution available for these problems.

Due to the computational complexity, evolutionary search methods such as genetic algorithm, tabu search, simulated annealing, differential evolution, particle swarm optimization, ant colony optimization, bee algorithms, etc., have been widely used in the last decade to solve problems with multiple objectives [1–8]. In recent years, some survey papers on multi-objective optimization methods have been published [9–11]. From these survey papers and the other original papers, the multi-objective optimization methods can be classified in different categories such as aggregative, lexicographic, sub-population, Pareto-based, and hybrid methods. Among the multi-objective methods, the majority of research is concentrated on Pareto-based approaches. However there are a few strategies based on foraging behaviours of honey bees for optimizing multi-objective problems [12].

Pareto-based methods select a part of individuals based on the Pareto dominance notion as leaders. These methods use different approaches to select non-dominated individuals as leaders [1–6]. Usually leaders are maintained in an external archive.

The Artificial bee colony (ABC) algorithm is one of the most recently introduced evolutionary methods [13,14].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. In the ABC, there are different types of bees who do different tasks to make the algorithm useful. The foragers will find food sources and try to find better ones with updating those they have found. The onlookers will choose one of those food sources based on the fitness of food sources. At last, scout bees will find the food sources which has not been optimized in a limited number of cycles so as to reinitialize them to get rid of poor solutions.

Our method is based on the standard ABC algorithm introduced by Karaboga and Basturk in [13]. The proposed method, called the multi-objective artificial bee colony (or MOABC), utilizes different types of bees (i.e. employed bees, onlookers, and scouts) and a fixed-sized archive to maintain the good solutions. To maintain the archive, we have used an ε-dominance method [15]. In ε-dominance, the size of the final external archive depends on the ε value, which is normally a user-defined parameter [15]. The social information provided by the external archive is used by the employed bees to adjust their flying trajectories. The diversity over the external archive is controlled using a grid. The onlookers evaluate the solutions provided by the employed bees to adjust their next position. Finally, the scout bees replace the solutions who has reached trial limit by the new random solution in the search space.

* Corresponding author.
E-mail addresses: akbari@sutech.ac.ir (R. Akbari), rhedayatzade@gmail.com (R. Hedayatzadeh), ziarati@shirazu.ac.ir (K. Ziarati), bahareh0902@yahoo.com (B. Hassanizadeh).

2210-6502/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.swevo.2011.08.001

The rest of the paper is organized as follows: An introduction to multi-objective optimization is presented in Section 2. Also, a survey on related works is presented in this section. Section 3 describes the details of the proposed MOABC algorithm for handling problems with multiple objectives. Next, the experimental results are discussed in Section 4. Finally, Section 5 concludes the paper.

2. Multi-objective optimization survey

Let $x = (x_1, x_2, \ldots, x_D)$ be a D-dimensional vector of decision variables and S be a search space. Facing with only a single objective, an optimal solution is just one for the objective function $f(x)$. We call $f(x^*)$ as the global minimum of a given function $f(x) : x \in R^D \to R$, for $x \in S$ if and only if:

$$\forall x \in S : \quad f(x^*) \le f(x). \tag{1}$$

However, multi-objective optimization (MOO) concerns optimizing problems with multiple and often conflicting objectives. A MOO problem can be stated as finding a solution vector $\vec{x} \in S$ which minimizes the function z with conflicting objectives:

$$z = f(x) = (f_1(x), f_2(x), \ldots, f_n(x)) \tag{2}$$

subject to k inequality constraint

$$g(x) = (g_1(x), g_2(x), \ldots, g_k(x)) \ge 0 \tag{3}$$

and m equality constraint

$$h(x) = (h_1(x), h_2(x), \ldots, h_m(x)) = 0. \tag{4}$$

Unlike single objective optimization, MOO solutions 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 MOO problem exists in the form of an alternate trade-off known as a Pareto optimal set. The Pareto optimal set is defined based on Pareto dominance. A decision vector x_a is said to dominate another vector x_b (denote as $x_a \prec x_b$) if:

$$f(x_{a,i}) \le f(x_{b,i}) \quad \forall i = \{1, 2, \ldots, D\} \quad \text{and}$$
$$\exists j \in \{1, 2, \ldots, D\} \quad \text{where } f(x_{a,i}) < f(x_{b,i}). \tag{5}$$

Therefore, a set of M decision vectors $\{w_i\}$ is a non-dominate set if:

$$w_i \nprec w_j \quad \forall i, j = 1, 2, \ldots, M. \tag{6}$$

Considering Pareto dominance, decision vector x_a is Pareto optimal if for every decision vector x_b and $I = \{1, 2, \ldots, D\}$ either $\forall_{i \in I} (f_i(x_b) = f_i(x_a))$ or if there is at least one $i \in I$ such that $f_i(x_b) > f_i(x_a)$. The Pareto optimal set (P^*) for a multi-objective problem $f(\vec{x})$ containing all Pareto optimal solutions is defined as:

$$P^* = \left\{ x \in S \mid \neg \exists x' \in S \text{ and } f(x') \prec f(x) \right\}. \tag{7}$$

Therefore, a Pareto front (PF^*) for a given multi-objective problem $f(\vec{x})$ and the Pareto optimal set P^*, is defined as:

$$PF^* = \left\{ f(x) \mid x \in P^* \right\}. \tag{8}$$

A Pareto based MOO method tries to find the optimal Pareto front. A Pareto front can be convex, concave or partially convex and/or concave and/or discontinuous. A Pareto front PF^* is called convex, if and only if:

$$\forall x_a, x_b \in PF^*, \forall \lambda \in (0,1), \exists \omega \in PF^* :$$
$$\lambda \|x_a\| + (1-\lambda)\|x_b\| \ge \|\omega\|. \tag{9}$$

Respectively, it is called concave, if and only if:

$$\forall x_a, x_b \in PF^*, \forall \lambda \in (0,1), \exists \omega \in PF^* :$$
$$\lambda \|x_a\| + (1-\lambda)\|x_b\| \le \|\omega\|. \tag{10}$$

The determination of a true Pareto front is a difficult task due to a large number of suboptimal Pareto fronts, the existing memory constraints, and computational complexity. However, difficulty may be increased due to the nature of the Pareto front. More precisely, for most MOO methods, a partially convex, concave, or discontinuous problem is harder to solve compared to convex ones.

Different types of MOO algorithms have been proposed in recent years. Evolutionary algorithms are among most popular meta-heuristic optimization methods. Several extensions to these algorithms for handling multiple objectives have been proposed. Individual selection and updating individuals using evolutionary operations are the main issues of multi-objective evolutionary algorithms. Archive-based Micro Genetic Algorithm (AMGA) [16], Clustering Multi-Objective Evolutionary Algorithm (ClusteringMOEA) [17], Differential Evolution with Self-adaptation and Local Search Algorithm (DECMOSA-SQP) [18], an improved version of Dynamical Multi-Objective Evolutionary Algorithm (DMOEADD) [19], Generalized Differential Evolution 3 (GDE3) [20], LiuLiAlgorithm [21], Multi-Objective Evolutionary Algorithm based on Decomposition (MOEAD) [22], Enhancing MOEA/D with Guided Mutation and Priority Update (MOEADGM) [23], Multi-Objective Evolutionary Programming (MOEP) [24], Multiple Trajectory Search (MTS) [25], Local Search Based Evolutionary Multi-Objective Optimization Algorithm (NSGAIILS) [26], an improved algorithm based on An Efficient Multi-objective Evolutionary Algorithm (OMOEAII) [27] and Multi-Objective Self-adaptive Differential Evolution Algorithm with Objective-wise Learning Strategies (OWMOSaDE) [28] are some of the competitive methods that aimed to obtain a true Pareto front for multi-objective problems by considering the main issues of MOO evolutionary algorithms.

Several types of swarm-based MOO algorithms have been presented in the literature. Updating leaders and leader selection are two main issues which are investigated by the particle swarm based methods. Multi-Objective Particle Swarm Optimization (MOPSO) [3], Interactive Particle Swarm Optimization (IPSO) [1], PSO-Based Multi-Objective Optimization With Dynamic Population Size and Adaptive Local Archives (DMOPSO) [29], Dynamic Multiple Swarms in Multi-Objective Particle Swarm Optimization (DSMOPSO) [30] are some of the competitive swarm-based algorithms which have investigated these issues.

The aforementioned algorithms are efficient and have competitive performance over the MOO problems. However, these algorithms have deficiencies in optimizing some of the multi-objective problems. Thus, designing new MOO algorithms for obtaining better solutions are required. Bee algorithms are among the recently introduced meta-heuristics. Recently, different types of bee algorithms such as the so called artificial bee colony (ABC) [13], bee swarm optimization (BSO) [31,32], bees algorithm (BA) [33], and bee colony optimization (BCO) [34] have been presented in the literature. These algorithms have the ability to extend them for optimizing multi-objective problems. A few MOO methods based on the aforementioned bee algorithms have been introduced [35–37].

In this work, a multi-objective variant of the ABC is presented.The ABC uses less control parameters. Hence it can be efficiently used for solving problems with multiple objectives. The ABC is an efficient optimization algorithm and compared to other algorithms such as GA and PSO has some attractive characteristics and in most cases has better performance. Also, gradient-based information is not used in ABC. Finally, it uses a flexible and well-balanced mechanism to balance exploration and exploitation in a fast manner [12]. From these observations, it seems that the ABC has the ability to handle problems with multiple objectives in an efficient way.

The flexibility and efficiency of the ABC provide the ability to design multi-objective optimization algorithms. Zou et al. presented a multi-objective optimization method based on the artificial bee colony [35]. Their algorithm called the MOABC uses the concept of Pareto dominance to determine the flight direction of a bee and it maintains the non-dominated solution vectors in an external archive. The main differences between their method and ours are: (1) They sort bees based on non-domination in the initialization phase and store them in the external archive; (2) In their method, all the bees are regarded as onlookers and there does not exist employeds and scouts; (3) The flying pattern of the bees is different from the flying patterns which are used by our method. Multi-objective variants of the ABC have been successfully applied on engineering problems. For example, Omkar et al. proposed a method called the Vector-Evaluated ABC (or VEABC) for the multi-objective design of composite structures [12]. The VEABC was designed based on the ideas borrowed from the VEGA [38] and VEPSO [39] algorithms. In the VEABC, multiple populations are 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 [40]. A Pareto-based discrete artificial bee colony algorithm was used by Li et al. for multi-objective flexible job shop scheduling problems in which a crossover operator is used by employed bees and several local searches are used to balance exploration and exploitation [41]. Finally, a multi-objective ABC was used in [42] for scheduling in grid environment.

Similar to the ABC, other bee algorithms have been used by authors to design multi-objective optimization algorithms. For example, a bee algorithm called the Autonomous Bee Colony Optimization (A-BCO) was proposed by Zeng et al. [37] for handling numerical problems with multiple objectives. Their work is different from the multi-objective methods which have been designed based on the ABC. The A-BCO used a diversity based performance metric to dynamically assess the archive set. In their work, the crowding distance method is used to manage the external archive by ranking non-dominated solutions. Also, they used three types of bees called elite, follower and scouts. The flying patterns of elite and followers are varied using crossover operator whereas scouts use mutation operators which are different from the proposed flying patterns in our work. As an application of multi-objective bee colony optimization, Low et al. used this algorithm to automated red teaming [43].

Another representative multi-objective optimization method was proposed by Pham and Ghanbarzadeh based on the standard BA in [44]. Their work uses the main steps of the BA presented in [33] to optimize a multi-objective welded beam design problem. In a multi-objective version of the BA, the non-dominated sites are designated as “selected-sites” which are used in the standard BA for neighbourhood search. It should be noted that the search process of the multi-objective BA is different from the search process of the A-BCO and multi-objective variants of the ABC algorithm. Another multi-objective variant of the BA called the BAMOP was proposed by Sayadi et al. for multi-objective optimization of time-varying channel for MIMO MC-CDMA systems [45]. A multi-objective bee algorithm called MBA was presented in [46]. The MBA method is a modified version of the BA and uses compromised fuzzy and clustering techniques to solve environment/economic dispatch problems.

3. Multi-objective artificial bee colony

The MOABC is designed based on the foraging behaviour of honey bees. In this colony, there are three types of bees: Employed, Onlooker and Scout bees. This algorithm assumes a number of food sources and works through optimizing them. Each of these food sources represents a solution of the problem. These food sources have been found by the employed bees and will be presented on the dance area. The onlooker bees wait on the dance area to choose a food source by its quality. Note that in the MOABC, the number of Employed bees and Onlooker bees are equal. When a food source

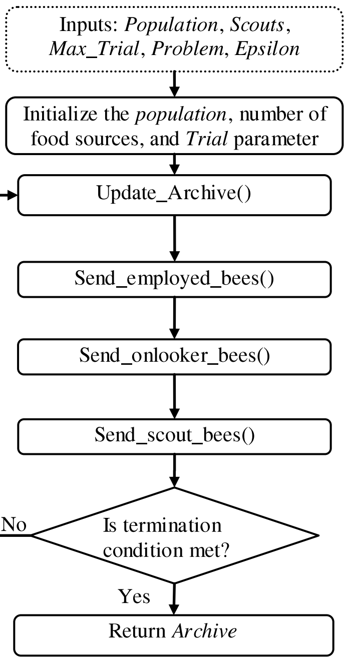

Inputs: Population, Scouts, Max_Trial, Problem, Epsilon
Initialize the population, number of food sources, and Trial parameter
Update_Archive()
Send_employed_bees()
Send_onlooker_bees()
Send_scout_bees()
Is termination condition met?
No
Yes
Return Archive

Fig. 1. Schematic diagram of the MOABC algorithm.

could not get optimized in a few cycles, a Scout bee will do a random search on the problem space to find a new food source.

Every problem which we are working on has a number of parameters and these parameters are continuous. A parameter is limited to a span, between lower and upper bound values. This can be formulated as $LB_d < x_d < UB_d$, in which d is the index of that parameter, x_d is the parameter and LB_d and UB_d are lower and upper bounds respectively. Each of these parameters can be considered as a dimension which will lead us to a search space. Hence, for a problem with D parameters or dimensions, a search space S can be considered that is a subset of R^D because none of the parameters are infinite. To work through the S space, the algorithm will associate a position vector $x = (x_1, x_2, \ldots, x_D)$ with each food source.

The MOABC is a Pareto based algorithm with an external archive to store non-dominated solutions. Note that two solutions are non-dominated when in a minimization problem such that no one has less value in all objectives than the other one or vice versa. To include the latest solutions found in the archive and check if they dominate other solutions or other archive members or versa, we need an updating method. The MOABC uses ε-dominance method for this reason that will be explained in Section 3.5.

The schematic diagram of the proposed method is given in Fig. 1. The MOABC method is constituted of five parts: Initialization, Send Employed Bees, Send Onlooker Bees, Send Scout Bees and Update the Archive. These parts are explained in the following sub-sections respectively.

The MOABC algorithm receives the parameters required as inputs: population size (*population*), number of scout bees (*Scouts*), *Max_Trial*, test problem (*Problem*), and *Epsilon*. *Max_Trial* is the value that the algorithm will use to find out which food sources should be abandoned. *Epsilon* is the value that will be used to partition the external archive in the ε-dominance method.

3.1. Initialization

In the initialization phase, a new variable named *FoodNumber* will be defined to hold the number of food sources. Since each employed bee represents a food source and the number of employed and onlooker bees are equal, the *FoodNumber* can be set to half of the population. Hence *FoodNumber* is equal to the number

```
Function Send_Employed_Bees()
    For $i$ = 1 to FoodNumber
        Select a parameter $d$ randomly
        Select Neighbor $k$ from the archive randomly
        Calculate $v_{id} = x_{id} + w_1 rand[0,1](x_{id} - x_{kd})$
        If the new food source dominates the old one
            Update the position
        End If
        If the food source has not been improved
            Increment its Trial by 1
        End If
    End For
```

Fig. 2. Send employed bee function.

of employed or onlooker bees. Note that in a case that a food source could not get optimized in a number of cycles, its related employed bee will turn into a scout bee and after doing a random search, it will turn back to an employed bee again. That is why no population has been considered for the scout bees.

Next, each food source will initialize through a function named $init(i, S)$, in which i is the index of the food source and S is the search space that we have defined before. In this way, a D dimensional vector $x_i = (x_1, x_2, \ldots, x_D)$ will be assigned randomly to each food source i through the following equation:

$$x_{id} = LB_d + rand[0, 1]^* (UB_d - LB_d), \tag{11}$$

where $d \in \{1, 2, \ldots, D\}$, $rand[0, 1]$ is a random number selected from a normal distribution in the range of zero and one, and LB_d and UB_d are lower and upper bounds along the d dimension respectively. Finally a $Trial_i$ variable will be assigned to each food source i in order to find food sources to be abandoned in the next iterations. This variable will be explained later in Section 3.4.

3.2. Send employed bees

The pseudocode of *Send_Employed_Bee* module is given in Fig. 2. To optimize food sources by their employed bees, the algorithm uses the archive since it contains the best solutions found so far. For this reason, each employed bee i would select an archive member randomly to calculate a temporary position called v_{id}. The position v_{id} is a copy of the position vector of the food source and one of its dimensions will be updated through the following equation:

$$v_{id} = x_{id} + w_1 rand[0, 1] (x_{id} - x_{kd}), \tag{12}$$

where i represents the food sources which is going to be optimized, $k \in \{1, 2, \ldots, FoodNumber\}$ and $d \in \{1, 2, \ldots, D\}$ are randomly chosen indexes. Although k is determined randomly, it has to be different from i. The random number r_{id} is a real number chosen randomly in −1 to 1 span. It controls the production of neighbour food sources around x_{id} and represents the comparison of two food positions visually by a bee. The coefficient w_1 controls the importance of the food source k in the production of the new food source. As can be seen from Eq. (2), as the difference between the parameters of the 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. If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value.

After v_{id} has been updated, the value of objective functions will be calculated. If the result could dominate the related food sources results, then it will be replaced by the food source's vector. If not, the update was unsuccessful and *Trial* value of that food source will be incremented by one.

```
Function Send_Onlooker_Bees()
    Calculate probabilities for each food source
    For $i$ = 1 to FoodNumber
        Select a parameter $d$ randomly
        Select Neighbor $k$ from food sources using equation (13)
        Calculate $v_{id} = x_{id} + w_2 rand[0,1](x_{id} - x_{kd})$
        If the new food source dominates the old one
            Update the position
        End If
        If the food source has not been improved
            Increment its Trial by 1
        End If
    End For
```

Fig. 3. Send onlooker bee function.

3.3. Send onlooker bees

The pseudocode of *Send_Onlooker_Bees* module is given in Fig. 3. After all employed bees optimized their food source by considering the information provided by the external archive, they will come to the hive to share their information with onlooker bees. Hence, each onlooker bee needs a decision making process to choose one of the food sources advertised by the employed bees.

For this purpose, the probability for each food source k advertised by the corresponding employed bee will be calculated as follows:

$$p_k = \frac{fit\left(\vec{x}_k\right)}{\sum_{m=1}^{FoodNumber} fit\left(\vec{x}_m\right)}, \tag{13}$$

where $fit\left(\vec{x}_m\right)$ is the probability of the food source proposed by the employed bee k which is proportional to the quality of food source. In the MOABC, the quality of a food source depends on the number of food sources dominated by the food source k. This can be formulated using the following equation:

$$fit\left(\vec{x}_m\right) = \frac{dom\,(m)}{FoodNumber}, \tag{14}$$

where $dom\,(m)$ is the function that returns the number of food sources dominated by food source m.

After that, onlookers can use a roulette wheel to choose a food source advertised by the employed bee k based on its probability. After an onlooker chooses a food source, she will select an archive member randomly and will do same as an employed bee to update that food source. The onlooker bee updates its position using the following equation if the newly discovered food source dominates the old one:

$$v_{id} = x_{id} + w_2 rand[0, 1] (x_{id} - x_{kd}), \tag{15}$$

where coefficient w_2 controls the importance of information provided by an employed bee k.

Using this probabilistic approach will make food sources better optimized and make it more probabilistic to put those better food sources into Archive.

For this reason, each food source should be evaluated to give a quality so that the onlooker could choose between different qualities.

3.4. Send scout bees

The pseudocode of *Send_Scout_Bees* module is given in Fig. 4. In this step, the algorithm will find abandoned food sources to replace them with new ones. Each employed or onlooker bee that

```
Function Send_Scout_Bees ()
    For $i$ = 1 to FoodNumber
        If $i$ has the maximum Trial
            $x_i$= init($i$, S)
            Trial$_i$ = 0
        End If
    End For
```

Fig. 4. Send scout bee function.

```
Function Update_Archive ()
    Define the Grid on the Archive
    Check for dominated boxes and remove their related
          members
    For each box remaining in Grid
        If the box contains more than one member
            Identify dominated ones box and remove them
        End If
        If the box still contains more than one member
            Keep the closest one to the left corner and remove the
                 Others
        End If
    End For
```

Fig. 5. Update archive function.

tries to optimize a food source has two possible endings. First, she cannot get it done, in which *Trial* of that food source will get incremented by one. Second, it can get it done, in which *Trial* value will be reset to zero. In this way, if the Trial reaches the *Max_Trial* value, its employed bee will turn to a scout bee and after doing a random search, she will reset it to zero. The *Max_Trial* parameter is determined manually. In this work, the value of *Max_Trial* is set to 60.

A scout produces a new position randomly and replaces the abandoned food source if the new food source has better nectar. Assume that the abandoned source is x_i and $j \in \{1, 2, \dots, D\}$, then the scout discovers a new food source to be replaced with x_i. This operation can be defined as follows:

$$v_{id} = LB_d + rand[0, 1]^* (UB_d - LB_d) . \quad (16)$$

After each candidate source position v_{ij} is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source dominates the old ones, it is replaced with the new ones in the memory. Otherwise, the old one is retained in the memory.

In this way, the algorithm will get rid of two types of food sources: first the ones who will not get good results or the ones which have been trapped in a local optimum, second the ones who have found their optimum solution and can work on the other areas of the search space.

3.5. Update archive

This algorithm is using a fixed size archive to hold the best solutions ever found. The pseudocode for updating the archive is given in Fig. 5. The archive is updated at each cycle of the algorithm as follows:

As stated before, to update archive in each cycle, ε-dominance method has been taken to use [15]. In ε-dominance method, a space with dimensions equal to number of the problem's objectives will be assumed. Each dimension will get sliced in an ε to ε size. This will break the space to boxes like squares, cubes or hypercubes for two, three and more than three objectives respectively. First, if a box that holds the solution(s) can dominate other boxes, those boxes will be removed. That means the solutions in those boxes will remove either. Then each box is examined to contain only one solution. Second, the dominated ones in each box will be eliminated. Next, if a box still has more than one solution, the solution with less distance from the left corner of the box will remain and the others will be removed. The use of ε-dominance guarantees that the retained solutions are non-dominated with respect to all solutions generated during the execution of the algorithm.

The ε-dominance approach for managing the external archive has been used extensively in previous works on multi-objective optimization algorithms. The PAES is one of the representative multi-objective methods which uses this approach [47]. In the PAES algorithm, a new candidate solution is rejected if it is dominated by any member of the archive, else all the dominated solutions by the new solution will be removed from the archive and the new solution will be added to the archive. If the candidate solution does not dominate any member of the archive and the archive is not full, the candidate will be added to the archive. If the archive is full, then the diversity of the archive is examined. If the candidate solution increases the diversity of the archive, it replaces a member of the archive residing in a crowded region.

The archiving process used by the MOABC is different from the archiving process of the PAES. The MOABC uses a grid based approach. In the MOABC, if a box that holds solution(s) can dominate other boxes, those boxes will be removed. That means the solutions in those boxes will remove either. Then each box is examined to contain only one solution. First, the dominated ones in each box will be eliminated. Next, if a box still has more than one solution, the solution with less distance from the left corner of the box will remain and the others will be removed.

3.6. Constraint handling

The proposed MOABC algorithm has the ability to solve both the constrained and unconstrained problems. To solve the constrained problems, the MOABC algorithm should provide a handling mechanism to cope with the infeasible solutions. A solution is called infeasible if it violates the constraints that should be satisfied. Different constraint handling methods have been used in multi-objective optimization. A study on the constrained multi-objective optimization has been presented by Qu and Suganthan in [48]. They have investigated three constraint handling methods along with their ensemble. In the MOABC algorithm, the superiority of the feasible solution method (SF) is used to cope with the constrained problems.

3.7. Termination

The steps *Update_Archive()*, *Send_Employed_Bee()*, *Send_Onlooker_Bee()*, and *Send_Scout_Bee()* iterate cycle by cycle according to the Fig. 1 until the termination condition is met. By termination of the MOABC algorithm, the external archive found by the population is returned as the output. In our implementation, the MOABC terminates after a predefined number of function evaluations. In order to facilitate the assessment of the performance of the proposed method, both the quantitative and qualitative results are provided.

4. Experimental results

This section contains the computational results obtained by the MOABC algorithm compared to the other 13 multi-objective methods over a set of test problems introduced in CEC09 [49].

Table 1
Mathematical representation of the unconstrained test problems.

Problem	Mathematical representation
UF1	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert} \sum_{j\in J_1} \left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2,\ f_1 = 1 - \sqrt{x} + \frac{2}{\lvert J_2\rvert} \sum_{j\in J_2} \left[x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right]^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$
UF2	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert} \sum_{j\in J_1} y_j^2,\ f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert} \sum_{j\in J_2} y_j^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ $y_j = \begin{cases} x_j - \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_1 \\ x_j - \left[0.3x_1^2 \cos\left(24\pi x_1 + \frac{4j\pi}{n}\right) + 0.6x_1\right] \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & j \in J_2 \end{cases}.$
UF3	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert} \left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1} \cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right),\ f_2 =$ $1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert} \left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} \cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ J_1 and J_2 are the same as those of UF1, $y_j = x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)},\ j = 2, \ldots, n$
UF4	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert} \sum_{j\in J_1} h(y_j),\ f_2 = 1 - x_2 + \frac{2}{\lvert J_2\rvert} \sum_{j\in J_2} h(y_j)$ J_1 and J_2 are the same as those of UF1, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2, \ldots, n,\ h(t) = \frac{\lvert t\rvert}{1+e^{2\lvert t\rvert}}$
UF5	$f_1 = x_1 + \left(\frac{1}{2N} + \varepsilon\right) \lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_1\rvert} \sum_{j\in J_1} h(y_j),\ f_1 = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right) \lvert\sin(2N\pi x_1)\rvert + \frac{2}{\lvert J_2\rvert} \sum_{j\in J_2} h(y_j)$ J_1 and J_2 are the same as those of UF1, $\varepsilon > 0,\ y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2, \ldots, n,$ $h(t) = 2t^2 - \cos(4\pi t) + 1$
UF6	$f_1 = x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \varepsilon\right) \sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_1\rvert} \left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1} \cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $f_2 = 1 - x_1 + \max\left\{0, 2\left(\frac{1}{2N} + \varepsilon\right) \sin(2N\pi x_1)\right\} + \frac{2}{\lvert J_2\rvert} \left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2} \cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $J_1$and J_2 are the same as those of UF1, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2, \ldots, n$
UF7	$f_1 = \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert} \sum_{j\in J_1} y_j^2,\ f_2 = 1 - \sqrt[5]{x_1} + \frac{2}{\lvert J_1\rvert} \sum_{j\in J_2} y_j^2$ J_1 and J_2 are the same as those of UF1, $y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right),\ j = 2, \ldots, n$

4.1. Performance metrics

In this work, we have used the IGD measure [22] in order to provide a quantitative assessment for the performance of the MOABC. The IGD measure is defined as follows: assume that P^* be a set of uniformly distributed points along the Pareto front in the objective space. Let A be a set of approximated points to the Pareto front. The average distance from P^* to A is defined using the following equation:

$$IGD\left(A, P^*\right) = \frac{\sum\limits_{\vartheta \in P^*} d(\vartheta, A)}{\lvert P^*\rvert}, \tag{17}$$

where $d(\vartheta, A)$ is the minimum Euclidean distance between ϑ and the points in A. If $\lvert P^*\rvert$ is large enough to represent the Pareto front very well, both the diversity and convergence of the approximated set A could be measured using $IGD(A, P^*)$. An optimization algorithm will try to minimize the value of $IGD(A, P^*)$ measure. To obtain smaller values of this measure, the approximated set A must be very close to the Pareto front and cannot miss any part of the whole Pareto front. The source code and data files for computing the IGD measure can be downloaded from [15].

4.2. Benchmarks

As the complexities of the optimization problems increase, we need to design more powerful methods to tackle with the complexities. It seems that the proposed method has the ability to cope with the complexities of the new problems. The CEC'09 benchmark presents a set of complex test problems. Several test functions from the CEC'09 are used in order to compare the performance of the proposed method. The performance of the MOABC algorithm is evaluated against the other algorithms over the UF1–UF10 and the CF1–CF7 test problems [49]. The mathematical representation of these test problems are given in Tables 1–3. The UF1–UF7 are unconstrained two objectives test problems and the UF8–UF10 are unconstrained three objectives test problems and the CF1–CF7 represents constrained two objective test problems. 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). A successful multi-objective optimization method should provide the ability to solve these problems with different types of Pareto fronts.

4.3. Experimental settings

To observe the performance of the proposed MOABC method in solving multi-objective problems, its performance is compared with the other methods which are presented in CEC'09 [49]. The proposed method is compared with AMGA [16], ClusteringMOEA [17], DECMOSA-SQP [18], DMOEADD [19], GDE3 [20], LiuLiAlgorithm [21], MOEAD [22], MOEADGM [23], MOEP [24], MTS [25], NSGAIILS [26], OMOEAII [27], and OWMOSaDE [28] methods on 10 unconstrained test problems and 7 constrained test problems [49]. The source code of the aforementioned algorithms can be downloaded from [15].

In the experiments, the number of function evaluations is set at 300 000. The maximum number of final solutions for computing the IGD measure is set at 100 and 150 for two-objective and three-objective problems. Each algorithm is evaluated independently 30 times for each test problem. The MOABC algorithm was tested with a population of size 20 and the iteration numbers was set at 15 000. The values of coefficients w_1 and w_2 are set at 0.7 and 0.8 respectively. The performance of the MOABC was evaluated under this configuration over all the test problems. For the other algorithms, the results reported by them are used here.

4.4. Performance analysis

4.4.1. Unconstrained problems

The CEC09's unconstrained problems are considered in this section. The average results on the UF1-UF10 test problems are

Table 2
Mathematical representation of the three-objectives unconstrained test problems (UF8–UF10).

Problem	Mathematical representation
UF8	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2$ $J_1 = \{j\lvert \le j \le n,$ and $j-1$ is a multiplication of $3\}$, $J_2 = \{j\lvert \le j \le n,$ and $j-2$ is a multiplication of $3\}$ $J_3 = \{j\lvert \le j \le n,$ and j is a multiplication of $3\}$
UF9	$f_1 = 0.5\left[\max\left\{0, (1+\varepsilon)(1-4(2x_1-1)^2)\right\} + 2x_1\right]x_2 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2$ $f_2 = 0.5\left[\max\left\{0, (1+\varepsilon)(1-4(2x_1-1)^2)\right\} + 2x_1\right]x_2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2$ $f_3 = 1 - x_2 + \frac{2}{\lvert J_3\rvert}\sum_{j\in J3}\left(x_j - 2x_2\sin\left(2\pi x_1 + \frac{j\pi}{n}\right)\right)^2,$ $J_1 = \{j\lvert \le j \le n,$ and $j-1$ is a multiplication of $3\}, J_2 = \{j\lvert \le j \le n,$ and $j-2$ is a multiplication of $3\}$ $J_3 = \{j\lvert \le j \le n,$ and j is a multiplication of $3\}$ and $\varepsilon = 0.1$
UF10	$f_1 = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{\lvert J_1\rvert}\sum_{j\in J1}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]$ $f_2 = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]$ $f_3 = \sin(0.5x_1\pi) + \frac{2}{\lvert J_3\rvert}\sum_{j\in J_3}\left[4y_j^2 - \cos\left(8\pi y_j\right) + 1\right]$ $J_1 = \{j\lvert \le j \le n,$ and $j-1$ is a multiplication of $3\}$, $J_2 = \{j\lvert \le j \le n,$ and $j-2$ is a multiplication of $3\}$ $J_3 = \{j\lvert \le j \le n,$ and j is a multiplication of $3\}$

given in Table 4. The MOABC algorithm produces competitive results on the UF1 test problem. The MOEAD and GD3 are two algorithms that produce better results compared to the MOABC algorithm. Also, competitive results are obtained by some of the algorithms such as MOEADGM and MTS algorithms.

Beside the quantitative comparisons of the investigated algorithms, the graphical representations of the Pareto fronts produced by the MOABC algorithm are given in Fig. 6. This figure shows the quality of the Pareto fronts produced by the MOABC algorithm. Fig. 6(a) shows the Pareto front produced by the MOABC algorithm over the UF1 test problem. The figure shows that the results produced not only have good convergence but also have appropriate distribution over the Pareto front in objective space.

The MOABC algorithm outperforms other algorithms when optimizing the UF2 test problem. The MOABC algorithm obtains the first rank on this test problem. Fig. 6(b) shows that the Pareto front produced has uniform distribution.

For the UF3 test problem, the MOABC algorithm obtains the fifth rank between 14 algorithms. The best convergence is obtained by the MOEAD algorithm. However, the MOABC algorithm has the ability to produce uniformly distributed Pareto fronts.

The MOABC algorithm obtains the eleventh rank on the UF4 test problem compared to the other 13 algorithms. The best results are found by the MTS and GD3 algorithms. Also, our empirical study showed that these algorithms have the smallest standard deviations which imply that they provide more robustness on this test problem compared to the other algorithms.

It seems that the UF5 presents a hard problem to solve. The UF5 has a discontinuous Pareto front. Most of the methods have difficulties in convergence to a Pareto front with uniform distribution. The proposed MOABC algorithm obtains the third rank between 14 algorithms. As can be seen from Fig. 6(e), the MOABC algorithm produces an archive where its members are uniformly distributed over the Pareto front.

The UF6 has a discontinuous Pareto front. 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. The MOEAD obtains the best performance on the UF6 test problem. The MOABC, MTS and DMOEADD algorithms produce competitive results on this test problem. Our empirical study showed that the MOEAD has the smallest standard deviation on the UF6 test problem. This property shows that MOEAD has more robustness in solving the UF6 test problem compared to the other investigated algorithms.

The MOABC algorithm obtains the twelfth rank on the UF7 test problem. The MOEAD, LiuLiAlgorithm, and MOEADGM algorithms obtain competitive performance on the UF7 test problems. As shown in Fig. 6(g), although the MOABC has good convergence over the optimal Pareto front, the top-left corner of the Pareto front is not successfully covered by the MOABC algorithm. Hence, the value of the IGD measure increases over the UF7 test problem.

The UF8 is the first three objectives test problem investigated in this paper. Usually, the complexity of multi-objective problems increases by the number of objectives to be optimized. The MOABC and MOEAD algorithm have competitive performance on this test problem. The best result is obtained by the MOEAD algorithm. The MOABC algorithm obtains the second rank. The quality of the approximated Pareto front is shown in Fig. 6(h). It is apparent from the results that the MOABC produced a set of solution points which have an appropriate distribution in the 3 dimensional objective space.

The MOABC algorithm has the ability to solve the UF9 test problem successfully. The second rank is obtained by our algorithm. The DMOEADD is the only algorithm which produces better performance than the MOABC algorithm. The quality of the approximated Pareto front can be seen in Fig. 6(i). The results show that the MOABC 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.

The MTS algorithm obtains the best result on the UF10 test problem. The MOABC produces competitive results compared to the MTS algorithm. The second rank is obtained by the MOABC algorithm. Fig. 6(j) shows the quality of the approximated Pareto front by the MOABC algorithm. The results show that the approximated Pareto front covers a large part of the objective space. However, compared to the approximated Pareto fronts of the UF8 and the UF9, the MOABC algorithm produces small number of points in the objective space. This problem occurs in some other test problems such as the UF4 and the UF6. It seems that this problem may be resolved by extending the number of function evaluations. The effect of number of function evaluations is given in Section 4.7.

4.4.2. Constrained problems

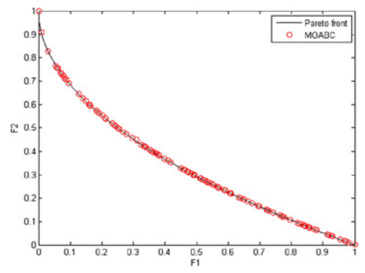
The constrained test problems are considered in this section. The average results of the investigated algorithms on the CF1–CF7 test problem are given in Table 5. The CF1 has a discontinuous

Table 3
Mathematical representation of the constrained test problems (CF1–CF7).

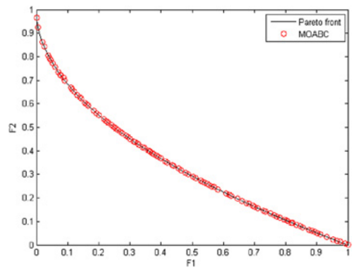
Problem	Mathematical representation
CF1	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}\right)^2,\ f_2(x) = 1 - x_1 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}\right)^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ Subject to: $f_1 + f_2 - a\lvert\sin[N\pi(f_1 - f_2 + 1)]\rvert - 1 \ge 0$ where N is an integer and $a \ge \frac{1}{2N}$
CF2	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}\left(x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2,\ f_2 = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}\left(x_j - \cos\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ Subject to: $\frac{t}{1+e^{4\lvert t\rvert}} \ge 0$ where $t = f2 + \sqrt{f_1} - a\sin\left[N\pi\left(\sqrt{f_1} - f_2 + 1\right)\right] - 1$
CF3	$f_1 = x_1 + \frac{2}{\lvert J_1\rvert}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right),\ f_2 = 1 - x_1^2 + \frac{2}{\lvert J_2\rvert}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right)$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\},\ y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \ldots, n$ Subject to: $f_2 + f_1^2 - a\left\lvert\sin\left[N\pi\left(f_1^2 - f_2 + 1\right)\right]\right\rvert - 1 \ge 0$
CF4	$f_1 = x_1 + \sum_{j\in J_1} h_j(y_j),\ f_2 = 1 - x_1 + \sum_{j\in J_2} h_j(y_j)$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\},\ y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \ldots, n,$ $h_2(t) = \begin{cases} \lvert t\rvert \text{ if } t < \frac{3}{2}\left(1 - \frac{\sqrt{2}}{2}\right) \\ 0.125 + (t-1)^2 \text{ otherwise} \end{cases},\ h_j(t) = t^2 \text{ for } j = 3, 4, \ldots, n$ Subject to: $\frac{t}{1+e^{4\lvert t\rvert}} \ge 0$ where $t = x_2 + \sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - 0.5x_1 + 0.25$
CF5	$f_1 = x_1 + \sum_{j\in J_1} h_j(y_j),\ f_2 = 1 - x_1 + \sum_{j\in J_2} h_j(y_j)$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\},\ y_j =$ $\begin{cases} x_j - 0.8x_1\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_1 \\ x_j - 0.8x_1\sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_2 \end{cases}.$ $h_2(t) = \begin{cases} \lvert t\rvert \text{ if } t < \frac{3}{2}\left(1 - \frac{\sqrt{2}}{2}\right) \\ 0.125 + (t-1)^2 \text{ otherwise} \end{cases},\ h_j(t) = 2t^2 - \cos(4\pi t) + 1 \text{ for } j = 3, 4, \ldots, n$ Subject to: $x_2 - 0.8x_1\sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - 0.5x_1 + 0.25 \ge 0$
CF6	$f_1 = x_1 + \sum_{j\in J_1} y_j^2,\ f_2 = (1 - x_1)^2 + \sum_{j\in J_2} y_j^2$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$ $y_j = \begin{cases} x_j - 0.8x_1\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) + 0.6x_1 \text{ if } & j \in J_1 \\ x_j - 0.8x_1\sin\left(6\pi x_1 + \frac{j\pi}{n}\right) + 0.6x_1 \text{ if } & j \in J_1 \end{cases}.$ Subject to: $x_2 - 0.8x_1\sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - sign\left(0.5(1 - x_1) - (1 - x_1)^2\right)\sqrt{\left\lvert 0.5(1 - x_1) - (1 - x_1)^2\right\rvert} \ge 0$ $x_4 - 0.8x_1\sin\left(6\pi x_1 + \frac{4\pi}{n}\right) - sign\left(0.25\sqrt{1 - x_1} - 0.5(1 - x_1)\right)\sqrt{0.25\sqrt{1 - x_1} - 0.5(1 - x_1)} \ge 0$
CF7	$f_1 = x_1 + \sum_{j\in J_1} h_j(y_j),\ f_2 = (1 - x_1)^2 + \sum_{j\in J_2} h_j(y_j)$ $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\},\ J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\},\ y_j = \begin{cases} x_j - \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_1 \\ x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_2 \end{cases}.$ $h_2(t) = h_4(t) = t^2,\ h_j(t) = 2t^2 - \cos(4\pi t) + 1 \text{ for } j = 3, 5, 6, \ldots, n$ Subject to: $x_2 - \sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - sign\left(0.5(1 - x_1) - (1 - x_1)^2\right)\sqrt{\left\lvert 0.5(1 - x_1) - (1 - x_1)^2\right\rvert} \ge 0$ $x_4 - \sin\left(6\pi x_1 + \frac{4\pi}{n}\right) - sign\left(0.25\sqrt{1 - x_1} - 0.5(1 - x_1)\right)\sqrt{0.25\sqrt{1 - x_1} - 0.5(1 - x_1)} \ge 0$

Table 4
The IGD statistics over UF1–UF10.

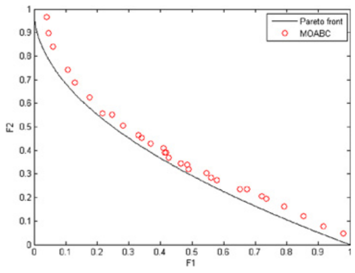
Algorithm	UF1	UF2	UF3	UF4	UF5	UF6	UF7	UF8	UF9	UF10
MOABC	**0.00618**	**0.00484**	**0.05120**	**0.05801**	**0.077758**	**0.06537**	**0.05573**	**0.06726**	**0.06150**	**0.19499**
MOEAD	0.00435	0.00679	0.00742	0.06385	0.18071	0.00587	0.00444	0.05840	0.07896	0.47415
GDE3	0.00534	0.01195	0.10639	0.02650	0.03928	0.25091	0.02522	0.24855	0.08248	0.43326
MOEADGM	0.00620	0.00640	0.04290	0.04760	1.79190	0.55630	0.00760	0.24460	0.18780	0.5646
MTS	0.00646	0.00615	0.05310	0.02356	0.01489	0.05917	0.04079	0.11251	0.11442	0.15306
LiuLi Algorithm	0.00785	0.01230	0.01497	0.04350	0.16186	0.17555	0.00730	0.08235	0.09391	0.44691
DMOEADD	0.01038	0.00679	0.03337	0.04268	0.31454	0.06673	0.01032	0.06841	0.04896	0.32211
NSGAIILS	0.01153	0.01237	0.10603	0.05840	0.56570	0.31032	0.02132	0.08630	0.07190	0.84468
OWMOSaDE	0.01220	0.00810	0.10300	0.05130	0.43030	0.1918	0.05850	0.09450	0.09830	0.74300
ClusteringMOEA	0.0299	0.02280	0.05490	0.05850	0.24730	0.08710	0.02230	0.23830	0.29340	0.41110
AMGA	0.03588	0.01623	0.06998	0.04062	0.09405	0.12942	0.05707	0.17125	0.18861	0.32418
MOEP	0.05960	0.01890	0.09900	0.04270	0.22450	0.10310	0.01970	0.42300	0.34200	0.36210
DECMOSA-SQP	0.07702	0.02834	0.09350	0.03392	0.16713	0.12604	0.02416	0.21583	0.14111	0.36985
OMOEAII	0.08564	0.03057	0.27141	0.04624	0.16920	0.07338	0.03354	0.19200	0.23179	0.62754



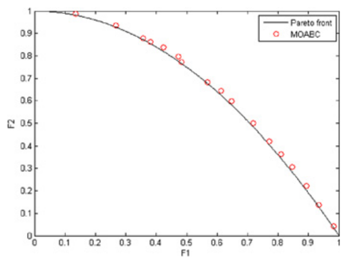
(a) UF1.



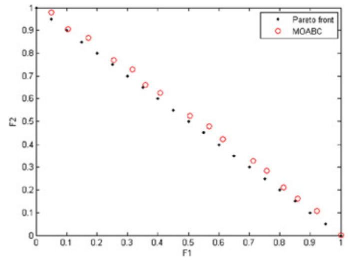
(b) UF2.



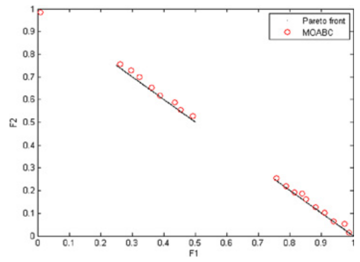
(c) UF3.



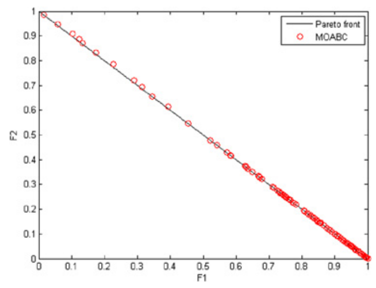
(d) UF4.



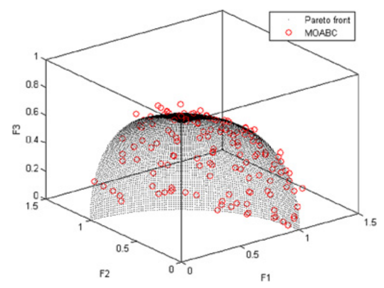
(e) UF5.



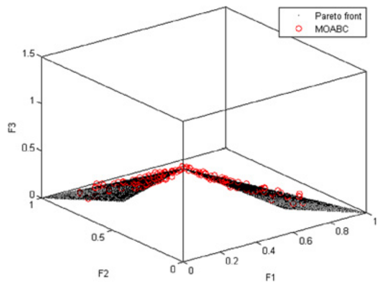
(f) UF6.



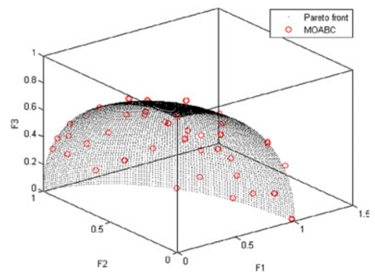
(g) UF7.



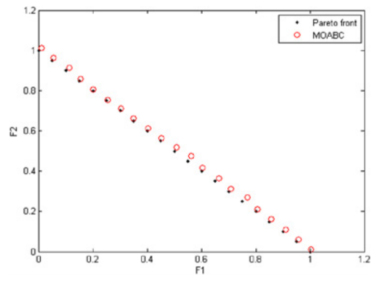
(h) UF8.



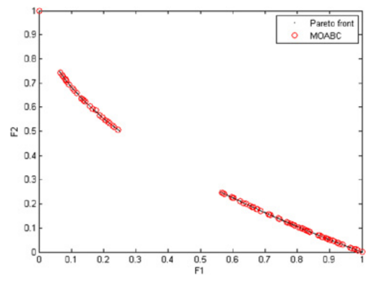
(i) UF9.



(j) UF10.



(k) CF1.



(l) CF2.

Fig. 6. The Pareto fronts obtained by the MOABC algorithm on the test problems UF1–UF10.

Pareto front. The MOABC obtains third rank on this test problem. The best average performance is obtained by the LiuLiAlgorithm.

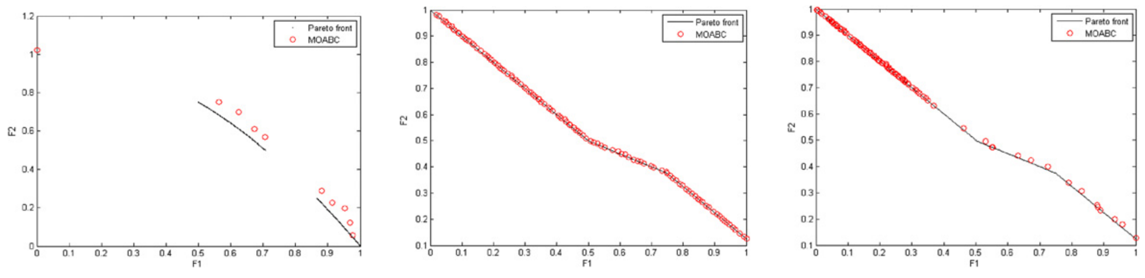
The average best result on the CF2 is obtained by the DMOEADD algorithm. The MOABC algorithm obtains the fourth rank on the CF1 test problem. The quality of the approximated Pareto front in Fig. 6(l) shows that the MOABC successfully converges to the optimal Pareto front. However, we can see discontinuities in the produced solutions of the MOABC algorithm.

Our empirical study showed that most of the algorithms have difficulty in solving the CF3 test problem. The DMOEADD and MOABC algorithms have competitive performance on this test problem. They respectively obtain the first and second ranks. The MOABC produces a small number of solutions for this test problem.

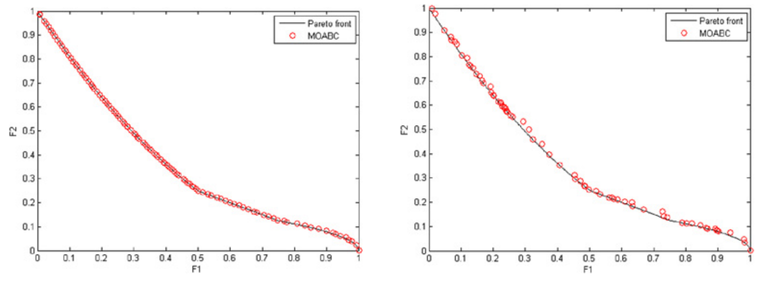
The CF4 test problem is successfully solved by the MOABC algorithm. The MOABC obtains the first rank. Fig. 6(n) shows that the MOABC algorithm produces a set of solutions which are uniformly distributed over the Pareto front.

The DMOEADD algorithm surpasses other algorithms in solving the CF5 test problem. The MOABC algorithm obtains the third rank between eight algorithms. Fig. 6(o) shows that the MOABC successfully converges to the optimal Pareto front. However, most of the produced solutions are gravitated to the left corner of the Pareto front and a uniform distribution of the solutions is not obtained by the MOABC algorithm.

There is a large difference between the performance of the MOABC algorithm and other investigated algorithms on the CF6 test problem. The MOABC successfully solves this test problem and obtains the first rank. As can be seen from Fig. 6(p), the MOABC successfully converges to the optimal Pareto front and its approximation has good distribution.



(m) CF3. (n) CF4. (o) CF5.



(p) CF6. (q) CF7.

Fig. 6. (*continued*)

Table 5
The IGD statistics over CF1–CF7.

Algorithm	CF1	CF2	CF3	CF4	CF5	CF6	CF7
MOABC	**0.00992**	**0.01027**	**0.08621**	**0.00452**	**0.06781**	**0.00483**	**0.01692**
GDE3	0.02940	0.01597	0.12750	0.00799	0.06799	0.06199	0.04169
MOEADGM	0.01080	0.00800	0.51340	0.07070	0.54460	0.20710	0.53560
MTS	0.01918	0.02677	0.10446	0.01109	0.02077	0.01616	0.02469
LiuLi Algorithm	0.00085	0.00420	0.18290	0.01423	0.10973	0.01394	0.10446
DMOEADD	0.01131	0.00210	0.05630	0.00699	0.01577	0.01502	0.01905
NSGAIILS	0.00692	0.01183	0.23994	0.01576	0.18420	0.02013	0.23345
DECMOSA-SQP	0.10773	0.09460	1000 000	0.15265	0.41275	0.14782	0.26049

The MOABC and DMOEADD algorithms have competitive performance on the CF7 test problem. However, the MOABC algorithm surpasses the DMOEADD algorithm and obtains the first rank on the CF6 test problem. From Fig. 6(p), we can see that although the produced solutions of the MOABC have no uniform distribution, it has the ability to successfully converge to the optimal solutions.

4.4.3. *Overall performance*

The performance of the proposed algorithm in comparison with other algorithms on the unconstrained and constrained test problems is investigated in Sections 4.4.1 and 4.4.2. The results showed that the proposed algorithm has the ability to provide competitive performance on the most of test problems. In this section, we use the lexicographic ordering to determine the overall rank of the proposed algorithm between 14 algorithms in the optimization of unconstrained problems and 8 algorithms in optimizing constrained problems. Table 6 shows the ranking of the best five algorithms in optimizing unconstrained problems and best three algorithms in optimizing constrained test problems.

The results show that the MOABC algorithm is the second best algorithm among 14 algorithms investigated for optimizing 10 unconstrained test problems. Also, the results show that the MOABC algorithm has better performance on the constrained test problems and it obtains the first rank.

The overall performance shows that the proposed algorithm which is designed based on intelligent behaviours of honey bees can be used as an alternative for optimizing problems with multiple objectives.

Table 6
List of best algorithms for optimizing unconstrained and constrained test problems.

Unconstrained		Constrained	
Rank	Algorithm	Rank	Algorithm
1	MTS	**1**	**MOABC**
2	**MOABC**	2	DMOEADD
3	MOEAD	3	LiuLiAlgorithm
4	DMOEADD		
5	LiuLiAlgorithm		

4.5. *Statistical test*

When two algorithms have to be compared on a given set of problem instances, one can use the performance of an algorithm to determine if its solution has better quality than the solution produced by the other methods for the same problem instance. One would like to use the algorithm which produces the best solutions. The Wilcoxon rank sum tests are conducted between the quality of the solution of the proposed algorithm and the best results found by the other algorithms. This test is used to determine whether the proposed algorithm is better than the other algorithms or not. The results of the 30 test runs for the MOABC and the best algorithm between other ones are used. We performed the test at a significance level of $\alpha = 0.025$. The h values in Table 7 show the results of the significance comparisons. We say that the proposed algorithm statistically has better performance than other algorithms if the p-value between its result and the best results from the other algorithms is smaller than the significance level. The one values of h show that the MOABC algorithm has statistically

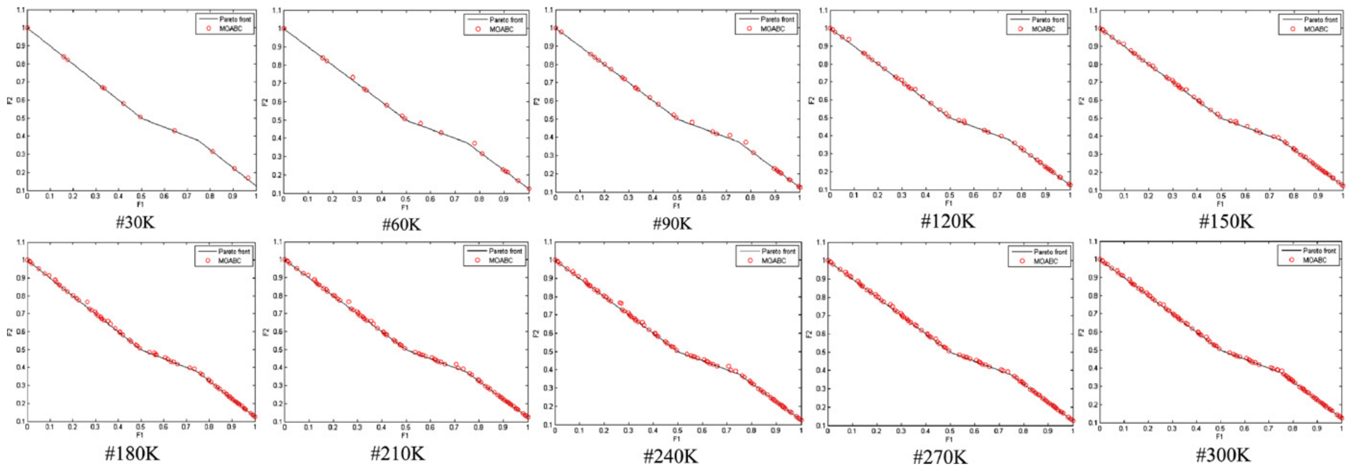


Fig. 7. Convergence behaviour of the MOABC algorithm throughout function evaluations.

Table 7
The results of the Wilcoxon rank sum test.

Function	*h*/best method	Function	*h*/best method
UF1	0/MOEAD	CF1	0/LiuLiAlgorithm
UF2	1	CF2	0/DMOEADD
UF3	0/MOEAD	CF3	0/DMOEADD
UF4	0/MTS	CF4	1
UF5	0/MTS	CF5	0/DMOEADD
UF6	0/MOEAD	CF6	1
UF7	0/MOEAD	CF7	1
UF8	0/MOEAD		
UF9	0/DMOEADD		
UF10	0/MTS		

Table 8
The IGD statistics over UF1–UF10.

Function	Best	Mean	Worst	Stdev.
UF1	0.00483	0.00494	0.00503	0.00006
UF2	0.00486	0.00500	0.00510	0.00007
UF3	0.01387	0.01737	0.02540	0.00428
UF4	0.01971	0.02017	0.02063	0.00036
UF5	0.00138	0.00177	0.00207	0.00020
UF6	0.00803	0.00906	0.01027	0.00075
UF7	0.00644	0.00826	0.01206	0.00164
UF8	0.05420	0.05739	0.06117	0.00202
UF9	0.03291	0.03427	0.03601	0.00104
UF10	0.10526	0.11969	0.16723	0.02156

Table 9
The IGD statistics over CF1–CF7.

Function	Best	Mean	Worst	Stdev.
CF1	0.00652	0.00811	0.00906	0.00076
CF2	0.00241	0.00244	0.00247	0.00002
CF3	0.02171	0.03432	0.05212	0.01429
CF4	0.00323	0.00351	0.00358	0.00013
CF5	0.00392	0.00423	0.00447	0.00025
CF6	0.00422	0.00425	0.00426	0.00001
CF7	0.00488	0.00496	0.00504	0.00007

better performance than other algorithms on a test function of the predefined dimensions. The zero values imply that the proposed algorithms do not have statistically better performance than other algorithms. For the 0 cases, the best performing algorithm is given. It is apparent from the results that the MOABC algorithm has significantly better performance than other algorithms on the UF2, CF4, CF6, and CF7 test problems.

4.6. *Convergence analysis*

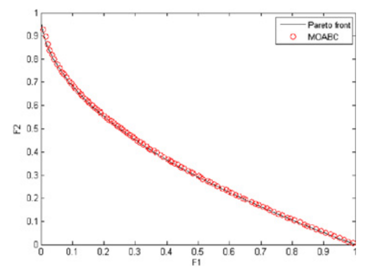
In this section, the behaviour of the MOABC in converging to the optimal Pareto front is studied. The convergence behaviour of the MOABC algorithm on the CF4 test problem is presented in Fig. 7. The results show that the MOABC algorithm continuously improves its performance throughout function evaluations. Although the MOABC 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. This means that the MOABC algorithm has the ability to avoid form trapping to local optima in the first times of function evaluations. 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. The MOABC algorithm helps the individuals to achieve this goal by allowing them to adjust their flying trajectories using the social knowledge provided by other individuals.

4.7. *The effects of the number of function evaluations*

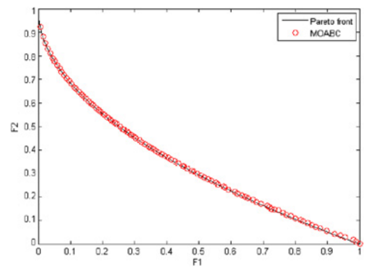
Section 4.4 showed that the proposed MOABC algorithm has competitive performance compared to the other investigated algorithms. Although, the MOABC provide good distribution over the Pareto front, in some test problems such as the UF4 and the UF6, it has difficulties in generating a large number of solutions. It seems that the MOABC needs more function evaluations to overcome this deficiency. In this section, we show that if the MOABC has the ability to provide more accurate solutions under extended numbers of function evaluations or not. For this purpose, the number of function evaluations is set at 60M and the convergence behaviour of the MOABC is evaluated. In this experiment, the population size is set at 200 and the iteration number is set at 300 000. For the other parameters of the MOBAC algorithm, the settings given in Section 4.3 are used.

Tables 8 and 9 show the effects of the extending number of function evaluations on the performance of the MOABC algorithm. The results show that the performance of the MOABC algorithm is significantly improved by extending the number of function evaluations. The qualities of the Pareto fronts produced for the test problems are shown in Fig. 8. Compared to the Pareto fronts given in Fig. 6, the new results show that the MOABC algorithm strongly improves its performance under the new configuration. Except to the CF3 test problem, the Pareto fronts produced over all the other test problems have almost uniform distribution.

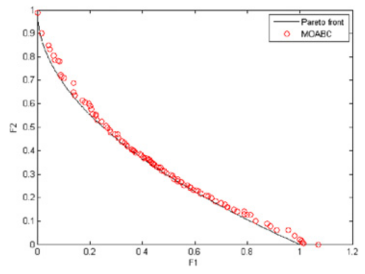
In general, we can see that the MOABC can obtain better performance by increasing the number of individuals in the population and the number of iterations. Our empirical study showed that the MOABC obtains best results when the number of



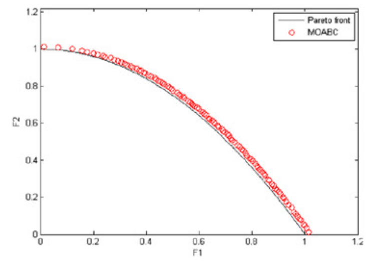
(a) UF1.



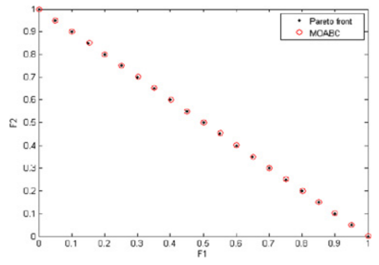
(b) UF2.



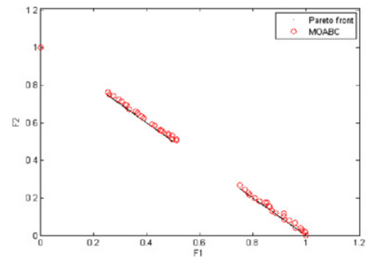
(c) UF3.



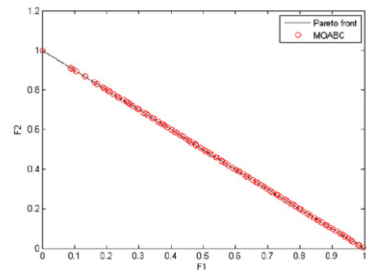
(d) UF4.



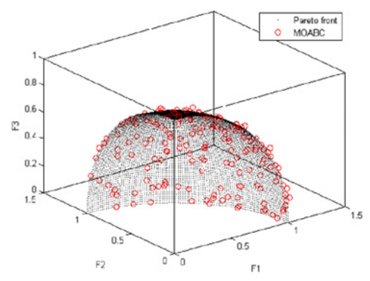
(e) UF5.



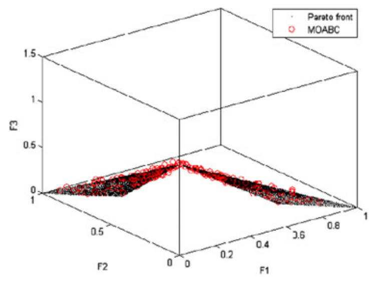
(f) UF6.



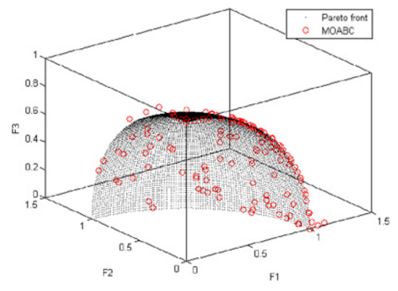
(g) UF7.



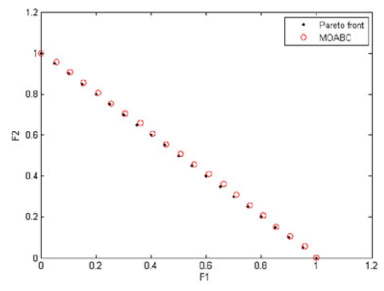
(h) UF8.



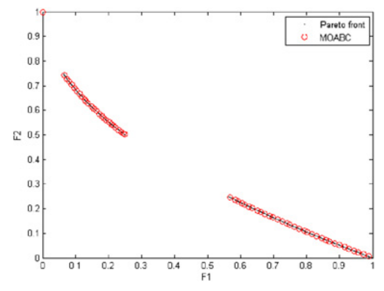
(i) UF9.



(j) UF10.



(k) CF1.



(l) CF2.

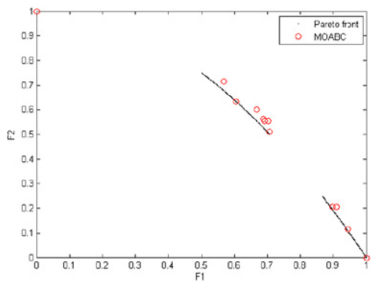
Fig. 8. The effects of number of function evaluations on the qualities of the Pareto fronts obtained by the MOABC algorithm.

individuals is selected between 150 and 200 and the number of iterations is larger than 50K.

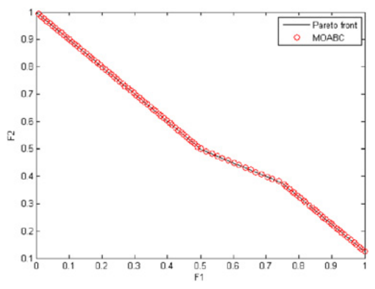
It seems that this type of behaviour occurs due to the grid-based archiving process. The archiving process used by the MOABC algorithm enforces each cell to contain only one solution. Although, this mechanism improves the diversity of the solutions provided by the population, most of the solutions located at the crowded portions of the Pareto front will be removed. Hence, the algorithm needs more iteration to compensate this deficiency.

5. Conclusion

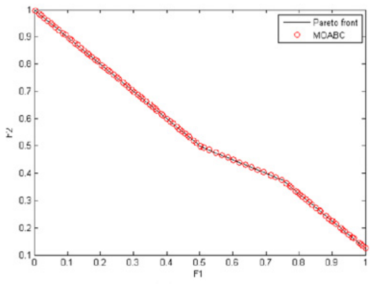
In this paper, we have presented a multi-objective optimization method based on intelligent behaviours of honey bees. The effectiveness of the MOABC depends on two factors: a population of different bee types with an efficient decision making process, and a grid for controlling the diversity over the external archive. A population with different flying patterns help the algorithm to increase the diversity of solutions. This may provide the ability for the algorithm to maintain an appropriate balance between exploration and exploitation. The MOABC with properties such as an effective trajectory adjusting strategy, and an efficient way for maintaining the diversity over the Pareto front, can be used as an alternate way for optimizing multi-objective problems. The comparative study showed that the proposed strategy produced results that are highly competitive with respect to the generational distance and spacing metrics.



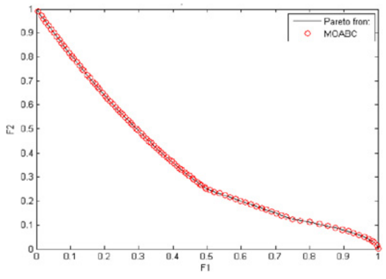
(m) CF3.



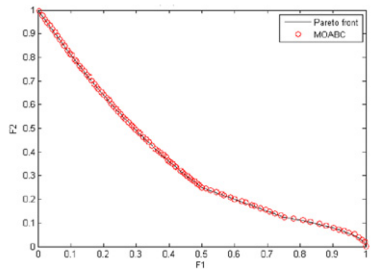
(n) CF4.



(o) CF5.



(p) CF6.



(q) CF7 .

Fig. 8. (*continued*)

References

[1] S. Agrawal, Y. Dashora, M.K. Tiwari, Y.J. Son, Interactive particle swarm: a Pareto-adaptive metaheuristic to multiobjective optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part A (Systems and Humans) 38 (2) (2008) 258–277.
[2] W.F. Leong, G.G. Yen, PSO-based multiobjective optimization with dynamic population size and adaptive local archives, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 38 (5) (2008) 1270–1293.
[3] C.A. Coello Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 256–279.
[4] D. Srinivasan, T.H. Seow, Particle swarm inspired evolutionary algorithm (ps-ea) for multiobjective optimization problem, Proceeding of Congress on Evolutionary Computation 3 (2003) 2292–2297.
[5] S. Mostaghim, J. Teich, Covering Pareto-optimal fronts by subswarms in multi-objective particle swarm optimization, Proceeding of Congress on Evolutionary Computation 2 (2004) 1404–1411.
[6] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[7] N. Srinivas, K. Deb, Multiobjective function optimization using nondominated sorting genetic algorithms, Evolutionary Computation 2 (3) (1995) 221–248.
[8] S.Z. Zhao, P.N. Suganthan, Two-lbests based multi-objective particle swarm optimizer, Engineering Optimization 43 (1) (2011) 1–17.
[9] A. Zhou, B.Y. Qu, H. Li, S.Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state-of-the-art, Journal of Swarm and Evolutionary Computation 1 (1) (2011) 32–49.
[10] V. Guliashki, H. Toshev, C. Korsemov, Survey of evolutionary algorithms used in multiobjective optimization, Journal of Problems of Engineering Cybernetics and Robotics 60 (2009) 42–54.
[11] M.R. Sierra, C.A. Coello Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, International Journal of Computational Intelligence Research 2 (3) (2006) 287–308.
[12] S. Omkar, N. Senthilnath, J.R. Khandelwal, G. Narayana Naik, S. Gopalakrishnan, Artificial bee colony (ABC) for multi-objective design optimization of composite structures, Journal of Applied Soft Computing 11 (1) (2011) 489–499.
[13] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (2007) 459–471.
[14] B. Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Systems with Applications 37 (2010) 5682–5687.
[15] K. Deb, M. Mohan, S. Mishra, Evaluating the epsilon-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions, Evolutionary Computation 13 (4) (2005) 501–525.
[16] S. Tiwari1, G. Fadel1, P. Koch, K. Deb, Performance assessment of the hybrid archive-based micro genetic algorithm (AMGA) on the CEC09 test problems, in: Proceeding of Congress on Evolutionary Computation.
[17] Y. Wang, C. Dang, H. Li, L. Han, J. Wei, A clustering multi-objective evolutionary algorithm based on orthogonal and uniform design, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 2927–2933.
[18] A. Zamuda, J. Brest, B. Boskovic, V. Zumer, Differential evolution with self-adaptation and local search for constrained multiobjective optimization, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 192–202.
[19] M. Liu, X. Zou, Y. Chen, Z. Wu, Performance assessment of DMOEA-DD with CEC 2009 moea competition test instances, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 2913–2918.
[20] S. Kukkonen, J. Lampinen, Performance assessment of generalized differential evolution with a given set of constrained multi-objective test problems, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 1943–1950.
[21] H. Liu, X. Li, The multiobjective evolutionary algorithm based on determined weight and sub-regional search, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 1928–1934.
[22] Q. Zhang, W. Liu, H. Li, The performance of a new version of MOEA/D on CEC09 unconstrained mop test instances, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 203–208.
[23] C.M. Chen, Y. Chen, Q. Zhang, Enhancing MOEA/D with guided mutation and priority update for multi-objective optimization, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 209–216.
[24] B.Y. Qu, P.N. Suganthan, Multi-objective evolutionary programming without non-domination sorting is up to twenty times faster, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 2934–2939.
[25] L.Y. Tseng, C. Chen, Multiple trajectory search for unconstrained/constrained multi-objective optimization, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 1951–1958.
[26] K. Sindhya, A. Sinha, K. Deb, K. Miettinen, Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 2919–2926.
[27] S. Gao, S. Zeng, B. Xiao, L. Zhang, Y. Shi, X. Tian, Y. Yang, H. Long, X. Yang, D. Yu, Z. Yan, An orthogonal multi-objective evolutionary algorithm with lower-dimensional crossover, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 1959–1964.
[28] V.L. Huang, S.Z. Zhao, R. Mallipeddi, P.N. Suganthan, Multi-objective optimization using self-adaptive differential evolution algorithm, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2009, pp. 190–194.
[29] W.F. Leong, G.G. Yen, PSO-based multiobjective optimization with dynamic population size and adaptive local archives, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 38 (5) (2008) 1270–1293.
[30] G.G. Yen, W.F. Leong, Dynamic multiple swarms in multiobjective particle swarm optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part A (Systems and Humans) 39 (4) (2009) 1013–1027.
[31] R. Akbari, A. Mohammadi, K. Ziarati, A novel bee swarm optimization algorithm for numerical function optimization, Communications in Nonlinear Science and Numerical Simulation 15 (9) (2010) 3142–3155.
[32] R. Akbari, K. Ziarati, A powerful bee swarm optimization algorithm, in: IEEE 13th International Multitopic Conference, 2010, pp. 1–6.

[33] D.T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, M. Zaidi, The bees algorithm, a novel tool for complex optimization problems, in: Proc 2nd Int Virtual Conf on Intelligent Production Machines and Systems, IPROMS 2006, Oxford: Elsevier (2006) 454–459.
[34] D. Teodorovic, M. Dell Orco, Bee colony optimization—a cooperative learning approach to complex transportation problems, in: Advanced OR and AI Methods in Transportation 51–60.
[35] W. Zou, Y. Zhu, H. Chen, H. Shen, A novel multi-objective optimization algorithm based on artificial bee colony, in: Genetic and Evolutionary Computation Conference, GECCO'11, 2011, pp. 103–104.
[36] R. Hedayatzadeh, B. Hasanizadeh, R. Akbari, Koorush Ziarati, A multi-objective artificial bee colony for optimizing multi-objective problems, in: 3rd International Conference on Advanced Computer Theory and Engineering, ICACTE 5, 2010, pp. 271–281.
[37] F. Zeng, J. Decraene, M.Y.H. Low, P. Hingston, C. Wentong, Z. Suiping, M. Chandramohan, Autonomous bee colony optimization for multi-objective function, in: Proceeding of Congress on Evolutionary Computation, CEC'09, 2010, pp. 1–8.
[38] J.D. Schaffer, Multi-objective optimization with vector evaluated genetic algorithms, Ph.D. Thesis, Vanderbilt University, Nashville, USA, (1984).
[39] S.N. Omkar, D. Mudigere, G. Narayana Naik, S. Gopalakrishna, Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures, Computers & Structures 86 (2008) 1–14.
[40] K. Atashkari, N. NarimanZadeh, A.R. Ghavimi, M.J. Mahmoodabadi, F. Aghaienezhad, Multi-objective optimization of power and heating system based on artificial bee colony, in: International Symposium on Innovations in Intelligent Systems and Applications, INISTA, 2011, pp. 64–68.
[41] J.Q. Li, Q.K. Pan, K.Z. Gao, Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems, International Journal of Advanced Manufacturing Technology 55 (9–12) (2011) 1159–1169.
[42] A. Rios, M. Vega-Rodriguez, M.A. Prieto-Castrillo, Multi-objective artificial bee colony for scheduling in grid environments, in: IEEE Symposium on Swarm Intelligence, SIS, 2011, pp. 1–7.
[43] Low, M.Y.H. Chandramohan, M. Chwee Seng Choo, Application of multi-objective bee colony optimization algorithm to automated red teaming, in: Proceedings of the 2009 Winter Simulation Conference, WSC, 2009, pp. 1798–1808.
[44] D. Pham, A. Ghanbarzadeha, Multi-objective optimization using the bees algorithm, in: Proceedings of the 3rd International Virtual Conference on Intelligent Production Machines and Systems, IPROMS 2007, Whittles, Dunbeath, Scotland, 2007.
[45] F. Sayadi, M. Ismail, N. Misran, K. Jumari, Multi-objective optimization using the bees algorithm in time-varying channel for mimo mc-cdma systems, European Journal of Scientific Research 33 (3) (2009) 411–428.
[46] N. Leeprechanon, P. Polratanasak, Multiobjective bees algorithm with clustering technique for environmental/economic dispatch, in: International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (2010) 621–625.
[47] J.D. Knowles, D.W. Corne, Approximating the nondominated front using the Pareto archived evolution strategy, Evolutionary Computation 8 (2000) 149–172.
[48] B.Y. Qu, P.N. Suganthan, Constrained multi-objective optimization algorithm with ensemble of constraint handling methods, Engineering Optimization 43 (4) (2011) 403–434.
[49] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the congress on evolutionary computation (CEC'09) 2009 special session and competition, Working Report, CES-887, School of Computer Science and Electrical Engineering, University of Essex, 2008.