IJST, Transactions of Electrical Engineering, Vol. 37, No. E1, pp 79-92
Printed in The Islamic Republic of Iran, 2013

AN ADAPTIVE MULTI-OBJECTIVE ARTIFICIAL BEE COLONY WITH CROWDING DISTANCE MECHANISM*

S. A. R. MOHAMMADI[1], M. R. FEIZI DERAKHSHI[2] AND R. AKBARI[3**]

[1]Dept. of Information Technology, Tabriz University, Tabriz, I. R. of Iran,
[2]Dept. of Computer Science Tabriz University, Tabriz, I. R. of Iran,
[3]Dept. of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz, I. R. of Iran
Email: akbari@sutech.ac.ir

Abstract– Artificial Bee Colony (ABC) is one of the recently introduced optimization methods based on intelligent behavior of honey bees. In this work, we propose an Adaptive Multi-Objective Artificial Bee Colony (A-MOABC) Optimizer which uses Pareto dominance notion and takes advantage of crowding distance and windowing mechanisms. The employed bees use an adaptive windowing mechanism to select their own leaders and alter their positions. Besides, onlookers update their positions using food sources presented by employed bees. Pareto dominance notion is used to show the quality of the food sources. Those employed or onlooker bees which find food sources with poor quality turn into scout bees in order to search other areas. The suggested method uses crowding distance technique in conjunction with the windowing mechanism in order to keep diversity in the external archive. The experimental results indicate that the proposed approach is not only thoroughly competitive compared to other algorithms considered in this work, but also finds the result with satisfactory precision.

Keywords– Multi-objective optimization, artificial bee colony, crowding distance, windowing mechanism

1. INTRODUCTION

Some kinds of optimization problems which are common in engineering have more than one objective. As we do not have a single solution for these sorts of problems, our goal is to find a set of solutions that represents a trade-off or balance between the objectives.

Different types of multi-objective optimization techniques have been proposed in literature. However, most of the multi-objective techniques, specially the recent ones, have been designed based on the notions such as Pareto-optimality and non-dominated solutions. These techniques are so-called Pareto-based methods which are described as follow:

Assume that we have a $\boldsymbol{D}$-dimensional search space $\boldsymbol{S}$. A multi-objective optimization problem tries to solve $\boldsymbol{k}$ conflicting objective functions simultaneously:

$$\boldsymbol{y_i = f_i(\vec{x})} \tag{1}$$

where $\boldsymbol{\vec{x} = (x_1, x_2, \dots, x_D)}$ is a $\boldsymbol{D}$-dimensional vector in the search space $\boldsymbol{S}$. The objective functions may be constrained or unconstrained. A constrained objective function should be optimized subject to the $\boldsymbol{m}$ constraints:

$$\boldsymbol{c_l(\vec{x}) \geq 0, \quad 1 \leq l \leq m} \tag{2}$$

*Received by the editors November 4, 2012; Accepted March 3, 2013.
**Corresponding author

Mathematically, a maximization problem can be simply converted to minimization. Hence, it can be assumed that objective functions should be minimized. By this assumption, the multi-objective optimization problem can be formulated as follows:

$$\boldsymbol{Minimize}\ \ \boldsymbol{y = f(\vec{x}) = \left(f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})\right)} \tag{3}$$

subject to:

$$\boldsymbol{c(\vec{x}) = \left(c_1(\vec{x}), c_2(\vec{x}), \dots, c_m(\vec{x})\right)} \tag{4}$$

In the multi-objective optimization, we have two or more conflicting objectives. Also, the multi-objective problems have more than one solution. Based on these, improving an objective is not possible unless at least one of the other conflicting objectives is sacrificed. The possible best results of the conflicting objectives are called Pareto-optimal set. A multi-objective optimization method tries to find the Pareto-optimal set. However, due to the complexities of the multi-objective problems, most of the algorithms have difficulty to find the true Pareto-optimal set. Hence, they try to find the Pareto-front. The Pareto-optimality is based on the dominance notion. For every two solution vector $\vec{z}_1$ and $\vec{z}_2$, it is said that $\vec{z}_1$ dominates $\vec{z}_2$ (denoted as $\vec{z}_1 \prec \vec{z}_2$) iff:

$$\boldsymbol{f_i(\vec{z}_1) \le f_i(\vec{z}_2) \quad \forall i: 1 \le i \le k} \tag{5}$$

$$\boldsymbol{\exists i \quad f_i(\vec{z}_1) < f_i(\vec{z}_2)} \tag{6}$$

Usually, the Pareto-based approaches maintain a set of non-dominated solutions in an external archive. A set of solutions $\boldsymbol{V}$ is known as a non-dominated set if every two solution members $\vec{v}_1$ and $\vec{v}_2$ do not dominate each other:

$$\boldsymbol{\vec{v}_i \nprec \vec{v}_j \quad \forall i,j: \quad 1 \le i,j \le H} \tag{7}$$

where $\boldsymbol{H}$ is the number of solution vectors. It is very difficult to determine the optimal Pareto-front due to the computational complexities and memory constraints. Hence, the meta-heuristic methods are preferred to solve the multi-objective optimization problems.

ABC is known as one of the meta-heuristic methods recently introduced by Karaboga and Basturk. The bee algorithms have been used in different fields of engineering [1]. In this work, the standard ABC is extended and an adaptive multi-objective ABC is introduced. In the proposed A-MOABC method, the population is divided into three different kinds of bees and an external archive is used to keep appropriate solutions. The leaders for each of the employed bees are chosen from this external archive to update their trajectories. A crowding distance technique is also used for non-dominated solutions in the external archive to estimate the density of solutions around it 3, 4]. The employed bees use a windowing mechanism to select their own leader [5]. The Onlookers use this information, which is found by the employed bees to adjust their trajectories. The proposed method is applied on a set of well-known multi-objective problems and compared with some state-of-the-art techniques.

The rest of this paper is organized as follows. A survey on the multi-objective methods is given in Section 2. In Section 3, the proposed algorithm is described in detail. The obtained experimental results are discussed in Section 4. Finally, Section 5 concludes this work.

2. LITERATURE SURVEY

In recent years, many population-based techniques such as multi-objective evolutionary algorithms (MOEAs), multi-objective particle swarm optimization algorithms (MOPSOs) and multi-objective artificial bee colony algorithms (MOABCs) have been designed for optimizing problems with more than one objective. Most studies on multi-objective optimizations concentrated on the Pareto-based approaches

and because of computational complexity, notable consideration is given to the evolutionary search methods to optimize multi-objective problems 6. Pareto-based approaches select non-dominated solutions based on the concept of Pareto dominance. These selected solutions are usually kept in an external archive [7-11].

Genetic Algorithms (GA) [12], Evolution Strategy (ES), Differential Evolution (DE), and Particle Swarm Optimization (PSO) have been used to design different classes of multi-objective optimization methods. These algorithm have been widely used and successfully extended to cope with the problems with more than one objective. Some interesting surveys on these methods have been presented in [13-16]. Reyes-Sierra and Coello Coello have presented a useful survey on the variants of PSO for multi-objective optimization [14]. In their study, PSO methods were categorized into six classes: aggregating, lexicographic, sub-population, Pareto-based, combined, and other approaches. The works presented in [17-20] are some of the representative multi-objective PSO methods. The variants of DE for multi-objective optimization problem have been considered by Mezura-Montes et al. in [13]. Based on their study, the variants of DE can be categorized as non-Pareto-based, Pareto-based using Pareto dominance or Pareto-ranking notions, and combined approaches. A categorization of multi-objective optimization methods for engineering problems has been given by Marler and Arora in 15]. Finally, a comprehensive survey of evolutionary-optimization based methods has been presented by Coello Coello in [16]. His work suggests that the evolutionary algorithms for handling multi-objective problems can be categorized as approaches that use aggregating functions, approaches not based on the notion of Pareto optimum, and Pareto-based approaches.

One of the most recently introduced evolutionary methods is Artificial Bee Colony (ABC) 21]. ABC algorithm can find solutions with great accuracy and it also has a satisfactory convergence speed in the single objective problems. These advantages could make this algorithm suitable for multi-objective optimizations. ABC has received much attention in recent years. It has been applied on many engineering problems. Due to the efficiency of ABC method, it has been extended by researchers to cope with multi-objective problems. The standard ABC has been used by Hedayatzadeh et al. to design multi-objective artificial bee colony (called MOABC) 11]. The MOABC is a Pareto-based approach which extends the standard ABC by employing an external archive. MOABC uses the ε-domination notion to maintain the non-dominated solutions in the archive. The performance of MOABC on a CEC'09 data sets has been investigated by Akbari et al. in [7]. Their study showed that the Pareto-based version of MOABC provide competitive performance compared to the other state-of-art algorithms. The concepts from the standard ABC have been used by Akbari and Ziarati to design a multi-objective bee swarm optimization algorithm (called MOBSO) [5]. The MOBSO has the ability to adaptively maintain an external archive of non-dominated solutions. Another multi-objective variant of ABC that uses the concept of Pareto-dominance and maintains the non-dominated solutions in an external archive presented in [22]. Some applications of multi-objective variants of ABC have been presented in [23-25]. A multi-objective technique for optimization of laminated composite components has been designed based on Vector Evaluated ABC (VEABC) [23]. A hybrid multi-objective ABC (HMABC) has been used by Zhang et al. for burdening optimization of copper strip production 24]. Their method solves a two-objective problem where the total cost of materials is minimized and the amount of waste material thrown into melting furnace is maximized. Also, a multi-objective variant of ABC (called MO-ABC) has been used to solve a real world frequency assignment problem in GSM networks [25].

3. THE PROPOSED METHOD

The proposed method is based on ABC algorithm. The population is divided into three different types of bees: employed, onlooker and scout bees. The employed bees find solutions or food sources and then present the positions and qualities of these food sources to the onlooker bees. Each of the onlooker bees decides to follow two of the employed bees according to the information received from employed bees. Also, the scout bees will explore the area randomly to find a new food source.

The pseudocode of the A-MOABC is given in Fig. 1. The A-MOABC is a Pareto-based algorithm, which keeps the non-dominated solutions in an external archive. This algorithm consists of four main parts: initialization, update bees, update the archive, and termination.

a) Initialization

In this phase, a variable *FoodNumber* is defined which stores number of food sources. The number of each group of employed bees and onlooker bees is set to the half of *FoodNumber* variable. There is also a trial variable which is defined and is used when one of the employed bees could not find a suitable source after a specific number of cycles that is defined by *trial* variable. If the cycles for each onlooker bee exceeded this limit, that onlooker bee turns into a scout bee. Each food source is initialized with the following equation:

$$x_{id} = Min_d + r_{[0,1]}(Max_d - Min_d) \tag{8}$$

where $\boldsymbol{x_i}$is a *D* dimensional vector, $\boldsymbol{r_{[0,1]}}$ is a random coefficient drawn from a normal distribution and $\boldsymbol{Max_d}$ and $\boldsymbol{Min_d}$ represent minimum and maximum values along dimension $\boldsymbol{d}$ respectively. The archive size $\boldsymbol{S_A}$ is also adjusted in the initializing phase.

b) Update mechanism

Each of the employed bees uses the external archive, which contains the best solutions found so far to update food sources. For this purpose each employed bee *i* would select two leaders from the archive to compute a temporary position called v_i. One of the leaders is selected by windowing mechanism and the other one is selected randomly. The reason behind selecting two types of leaders is that one of them which is selected by windowing mechanism will guide the employed bees to converge to the sparse parts of the Pareto set and the other one will help the employed bees to preserve diversity of search process over the solution space. One of the dimensions of the position v_i that is v_{id} is updated by the following equation:

$$v_{id} = x_{id} + w_1 r_{[-1,1]}(x_{id} - x_{kd}) + w_2 r_{[-1,1]}(x_{id} - x_{jd}) \tag{9}$$

where *i* represents the food sources which are going to be optimized, $\boldsymbol{k, j} \in \{\boldsymbol{1, 2, \dots, S_A}\}$ and $\boldsymbol{d} \in \{\boldsymbol{1, 2, \dots, D}\}$ are randomly chosen indices. It should be noticed that *k* has to be different from *i*. Index *j* will be selected with windowing mechanism. The coefficients w_1 and w_2 indicate which of the leaders should have a greater influence in guiding the employed bees.

After computing v_i, the values of objective functions will be calculated. If the result dominates the old values, the new one will replace the old values and positions, otherwise the *trial* value is incremented by one.

The employed bees select one of their leaders with the windowing mechanism. The windowing mechanism was used by Akbari and Ziarati 5]. An employed bee chooses her leader with a probability p_k as follows:

$$P_k = \frac{f(\vec{x}_{A,j})}{\sum_{k=1}^{S_A} f(\vec{x}_{A,k})} \tag{10}$$

where *j* is the index of an archive member and *n* is the number of archive members, $\boldsymbol{f}(\vec{\boldsymbol{x}}_{\boldsymbol{A,j}})$ is the fitness value of the food source proposed by the archive member *j* which it depends on number of archive members around its neighborhood and is calculated as follows:

$$f(\vec{x}_{A,j}) = {}^{1}/_{N_{A,j}} \tag{11}$$

where $\boldsymbol{N_{A,j}}$ indicates number of members around the archive member *j*. To calculate neighbors, an adaptive windowing mechanism is used. Placing each archive member in the center of a window does this. The window length in *i*-th objective dimension is calculated as follows for a MOO problem with *n* objectives:

$$l_i = \frac{|Max_i - Min_i|}{S_A}, i \in \{1,2,\dots,n\} \tag{12}$$

where $\boldsymbol{Max_i}$ and $\boldsymbol{Min_i}$ respectively presents the maximum and minimum values of the *i*-th objective function. The $\mathbf{N_{A,j}}$ for the archive member *j* can be defined as the number of archive members existing in its local window. The roulette wheel approach is used for selecting a suitable archive member as a leader.

After employed bees finish optimizing their food sources, they come back to the hive and share their information with onlooker bees. Each of the onlooker bees chooses one of the food sources announced by the employed bees according to their probabilities. The probability of food source *k* is calculated as:

$$P_k = \frac{f(\vec{x}_k)}{\sum_{m=1}^{FoodNumber} f(\vec{x}_m)} \tag{13}$$

where $\boldsymbol{f}(\vec{\boldsymbol{x}}_{\boldsymbol{m}})$ is the fitness of the food source *m*. The fitness is calculated as the number of food sources that are dominated by the food source *m* divided by *FoodNumber*. In other words, it can be stated mathematically as:

$$f(\vec{x}_m) = \frac{dom(m)}{FoodNumber} \tag{14}$$

where function $\boldsymbol{dom}(\boldsymbol{m})$ returns the number of food sources dominated by food source *m*.

The onlookers use the roulette wheel approach in order to choose a food source presented by the employed bees. After onlookers select food sources, they calculate their new positions. If the new solution dominates the old one, then the position will be updated. This process is formulated as:

$$v_{id} = x_{id} + w_3 r_{[-1,1]}(x_{id} - x_{kd}) \tag{15}$$

where coefficient $\boldsymbol{w_3}$ controls the influence of information provided by an employed bee *k*.

After sending onlooker bees, the algorithm checks if any poor quality food sources exist to replace them with new ones. If the *trial* limits of any food sources are exceeded then scout bees try to find new ones by reinitiating them. If the old food source is dominated by the new one then it is replaced. The reinitiating formula is like the Eq. (1). This phase has two advantages, one is that the algorithm could get rid of poor quality food sources and the other is that it could avoid trapping the algorithm into the local optimum. The pseudo-code of this function is shown in Fig. 1.

c) Update archive

A fixed size archive is considered for this algorithm to hold the best non-dominated solutions. At first, we check solutions to see if they are feasible and satisfy all constraints. If a food source satisfies all constraints, it is a potential candidate to be stored in the archive and if it is not dominated by all the solutions in the archive it will be inserted into the archive. In this process if the food source dominates any solutions, those solutions will be deleted from the archive. We may face two situations in inserting candidate food sources into the archive. The first one is that we have enough places in the archive to insert candidate food sources. There is no problem with this situation but the problem occurs in another case.

The second one is that we have no place to insert candidate food sources, so we should use a technique to select a victim from the archive and replace the new solution with that victim. This algorithm uses crowding distance technique to overcome this problem.

```
Adaptive MOABC
    Initialization
    While (termination condition is not met)
        Call Send_Employed_Bees();
        Call Send_Onlooker_Bees();
        Call Send_Scout_Bees ();
        Call Update_Archive ();
        Call Crowding_distance ();
    End While
    Return Archive
```

```
Function Send_Employed_Bees()
    For i = 1 to FoodNumber
        Select a parameter d randomly
        Select Neighbor k from the archive randomly
        Select Neighbor j from the archive with windowing
               mechanism
        Calculate v_id using (5)
        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
```

```
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 randomly
    Calculate v_id using (11)
    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
```

```
Function Send_Scout_Bees ()
For i = 1 to FoodNumber
    If i exceeds maximum Trial
        x_i= initiate(i, S)
        Trial_i = 0
    End If
End For
```

```
Function Update_Archive ()
Set insertFlag = true
For i = 1 to FoodNumber
    If archive is empty
        Insert particle in pop into archive
    End If
    If archive is not empty
        For each particle in archive
            If both particles are infeasible
                Delete particle in archive
                Set insertFlag = false
            End If
            If particle in pop is infeasible
                Set insertFlag = false
            End If
            If particle in archive is infeasible
                Delete particle in archive
            End If
            If both are feasible
                If bee in pop dominates
                    Delete particle in archive
                End If
                If bee in archive dominates
                    Set insertFlag = false
                End If
            End If
            If insertFlag
                If archive is not yet full
                    Insert particle in pop into archive
                End If
                If archive is full
                    Select a victim by crowding distance
technique
                    Replace bee in pop with selected victim in
archive
                End If
            End If
        End For
End For
```

```
Function Crowding_distance ()
Initialize distance for every solution in archive to zero
For each objective m
    Sort archive members by that objective
    For each archive member i
        Compute distance_i = distance_i + ArcMemberValue_(i+1)m -
ArMemberValue_(i-1)m
    End For
    Set distance_o= distance_arcSize = maximum distance
End For
```

Fig. 1. Pseudocode of the adaptive MOABC

The estimation of density that surrounds a solution is called the crowding distance value [3]. As described in 3, to calculate crowding distance of solution *i* in the archive we compute the largest estimation of the cuboid encompassing solution *i* without including any other points. Figure 2 makes this description more illustrative. In this technique, for every objective function, solutions are sorted in ascending objective function values. Then for computing the crowding distance value of a single solution, the average distance of its two neighboring solutions will be computed. Infinite crowding distance values are given to the boundary solutions, which they have the lowest and highest objective function values so we do not sacrifice boundary solutions. The final crowding distance of a solution is calculated by summation of its crowding distance values in each objective function.

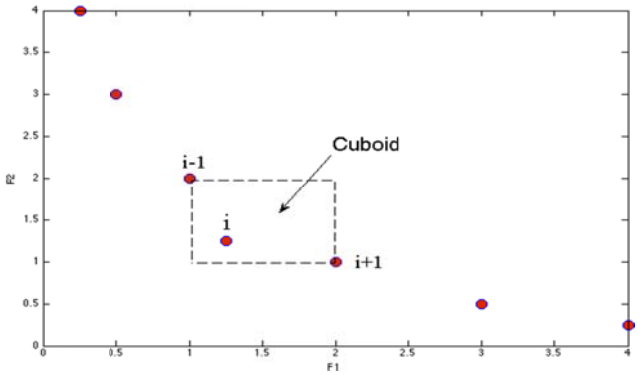


Fig. 2. Schematic computation of the crowding distance

d) Termination

There are different types of termination conditions such as iteration numbers, number of function evaluation, etc. that can be used to stop an evolutionary method. Here, we used the number of function evaluations as the termination condition. The algorithm terminates after the predefined number of function evaluations. After termination, the obtained Pareto-front by the algorithm is returned as the result.

4. EXPERIMENTAL STUDY

This section presents the experimental results to evaluate the performance of the A-MOABC compared to the other 14 multi-objective methods over a set of test problems introduced in CEC09 26] based on the IGD measure.

a) Performance metrics

IGD measure [27] is used in order to provide a quantitative evaluation for the performance of the proposed algorithm. The IGD measure is used as follows: consider P^* as Pareto optimal set, which is uniformly distributed over the objective space. Let A be a set of closed and approximated points to the Pareto front. The average distance from P* to A is formulated as:

$$IGD(A, P^*) = \frac{\sum_{\vartheta \in P^*} d(\vartheta, A)}{|P^*|}$$

where $\boldsymbol{d(\vartheta, A)}$ is the minimum Euclidian distance between $\boldsymbol{\vartheta}$ and the points in the approximated points A. Both the diversity and the convergence of the approximated set A could be measured using $\boldsymbol{IGD(A, P^*)}$.

b) Settings of parameters

The proposed algorithm has some parameters that should be tuned in order to obtain the best results. Some of these parameters are common among the investigated algorithms. The number of function

evaluation is set at 300000 in this experiment and the maximum number of archive is set at 100. Each algorithm is evaluated 30 times for each of the test problems and then an average over all the obtained results is taken. Each algorithm has some specific parameters. The proposed algorithm was examined with a population of size 50 and the iteration numbers were set at 6000. Hence, the number of function evaluations will be 300000. The values of coefficients w_1, w_2 and w_3 are set at 1.5, 1.0 and 1.5, respectively. It should be noted that, the parameters values for the proposed algorithm were obtained empirically.

c) Unconstrained test functions

In order to compare the performance of the proposed method in comparison with other methods, we use seven unconstrained test problems UF1 to UF7. These test problems have two objectives which they should be minimized. The mathematical formulation of the unconstrained problems are given below:

$$UF1:\begin{cases} f_1 = x_1 + \dfrac{2}{|J_1|}\sum_{j\in J_1}\left[x_j - \sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right)\right]^2, J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\} \\ f_1 = 1-\sqrt{x} + \dfrac{2}{|J_2|}\sum_{j\in J_2}\left[x_j - \sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right)\right]^2, J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\} \end{cases} \tag{16}$$

$$UF2:\begin{cases} f_1 = x_1 + \dfrac{2}{|J_1|}\sum_{j\in J_1} y_j^2, \quad J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\} \\ f_2 = 1-\sqrt{x_1} + \dfrac{2}{|J_2|}\sum_{j\in J_2} y_j^2, \quad J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\} \\ y_j = \begin{cases} x_j - \left[0.3x_1^2\cos\left(24\pi x_1 + \dfrac{4j\pi}{n}\right) + 0.6x_1\right]\cos\left(6\pi x_1 + \dfrac{j\pi}{n}\right) & j \in J_1 \\ x_j - \left[0.3x_1^2\cos\left(24\pi x_1 + \dfrac{4j\pi}{n}\right) + 0.6x_1\right]\sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right) & j \in J_2 \end{cases} \end{cases} \tag{17}$$

$$UF3:\begin{cases} f_1 = x_1 + \dfrac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\dfrac{20y_j\pi}{\sqrt{j}}\right) + 2\right), \quad y_j = x_j - x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}, \ j = 2,...,n \\ f_2 = 1-\sqrt{x_1} + \dfrac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\dfrac{20y_j\pi}{\sqrt{j}}\right) + 2\right), \quad J_1 \text{ and } J_2 \text{ are the same as those of UF1} \end{cases} \tag{18}$$

$$UF4:\begin{cases} f_1 = x_1 + \dfrac{2}{|J_1|}\sum_{j\in J_1} h(y_j), \quad y_j = x_j - \sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right), j = 2,...,n, \quad h(t) = \dfrac{|t|}{1+e^{2|t|}} \\ f_2 = 1 - x_2 + \dfrac{2}{|J_2|}\sum_{j\in J_2} h(y_j), \quad J_1 \text{ and } J_2 \text{ are the same as those of UF1} \end{cases} \tag{19}$$

$$UF5:\begin{cases} f_1 = x_1 + \left(\dfrac{1}{2N} + \varepsilon\right)\left|\sin(2N\pi x_1)\right| + \dfrac{2}{|J_1|}\sum_{j\in J_1} h(y_j), \varepsilon > 0, y_j = x_j - \sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right), j = 2,...,n \\ f_2 = 1 - x_1 + \left(\dfrac{1}{2N} + \varepsilon\right)\left|\sin(2N\pi x_1)\right| + \dfrac{2}{|J_2|}\sum_{j\in J_2} h(y_j), h(t) = 2t^2 - \cos(4\pi t) + 1, J_1 \text{ and } J_2 \text{ are the same as those of UF1} \end{cases} \tag{20}$$

$$UF6:\begin{cases} f_1 = x_1 + \max\left\{0, 2\left(\dfrac{1}{2N} + \varepsilon\right)\sin(2N\pi x_1)\right\} + \dfrac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\dfrac{20y_j\pi}{\sqrt{j}}\right) + 2\right), y_j = x_j - \sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right), j = 2,...,n \\ f_2 = 1 - x_1 + \max\left\{0, 2\left(\dfrac{1}{2N} + \varepsilon\right)\sin(2N\pi x_1)\right\} + \dfrac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\dfrac{20y_j\pi}{\sqrt{j}}\right) + 2\right), J_1 \text{ and } J_2 \text{ are the same as those of UF1} \end{cases} \tag{21}$$

$$UF7:\begin{cases} f_1 = \sqrt[5]{x_1} + \dfrac{2}{|J_1|}\sum_{j\in J_1} y_j^2, y_j = x_j - \sin\left(6\pi x_1 + \dfrac{j\pi}{n}\right), j = 2,...,n \\ f_2 = 1 - \sqrt[5]{x_1} + \dfrac{2}{|J_2|}\sum_{j\in J_2} y_j^2, J_1 \text{ and } J_2 \text{ are the same as those of UF1} \end{cases} \tag{22}$$

Table 1 shows the results on unconstrained test problems. It can be seen from Table 1 that the proposed algorithm obtain rank one on UF2. It also has a competitive performance on the UF3 and UF7.

The A-MOABC showed satisfactory performance over UF1, UF4, UF5 and UF6 test problems and placed among the best algorithms. Lexicographic ordering is used to specify the overall rank of our algorithm among 15 algorithms in optimizing unconstrained test problems. The results show that our algorithm is the best algorithm among 15 algorithms for optimizing 7 unconstrained problems. The first rank shows the efficiency of the proposed approach in solving multi-objective optimization problems. It seems that using adaptive window, crowding distance mechanism, and different movement patterns provides the ability for the proposed method to provide competitive results over the unconstrained problems in comparison with the other methods investigated here.

Table 1. The IDG statistics over U1-U7 test problems

Test Problem / Algorithm	UF1	UF2	UF3	UF4	UF5	UF6	UF7
A-MOABC	**0.00501**	**0.00435**	**0.04843**	**0.02632**	**0.05627**	**0.04151**	**0.02613**
MOABC	0.00618	0.00484	0.05120	0.05801	0.07775	0.06537	0.05573
MOEAD	0.00435	0.00679	0.00742	0.06385	0.18071	0.00587	0.00444
GDE3	0.00534	0.01195	0.10639	0.02650	0.03928	0.25091	0.02522
MOEADGM	0.00620	0.00640	0.04290	0.04760	1.79190	0.55630	0.00760
MTS	0.00646	0.00615	0.05310	0.02356	0.01489	0.05917	0.04079
LiuLi Algorithm	0.00785	0.01230	0.01497	0.04350	0.16186	0.17555	0.00730
DMOEADD	0.01038	0.00679	0.03337	0.04268	0.31454	0.06673	0.01032
NSGAIILS	0.01153	0.01237	0.10603	0.05840	0.56570	0.31032	0.02132
OWMOSaDE	0.01220	0.00810	0.10300	0.05130	0.43030	0.1918	0.05850
ClusteringMOEA	0.0299	0.02280	0.05490	0.05850	0.24730	0.08710	0.02230
AMGA	0.03588	0.01623	0.06998	0.04062	0.09405	0.12942	0.05707
MOEP	0.05960	0.01890	0.09900	0.04270	0.22450	0.10310	0.01970
DECMOSA-SQP	0.07702	0.02834	0.09350	0.03392	0.16713	0.12604	0.02416
OMOEAII	0.08564	0.03057	0.27141	0.04624	0.16920	0.07338	0.03354

d) Constrained test functions

The experiments on the unconstrained test problems show that the proposed algorithm surpasses other methods. However, the performance of an optimization method may be affected by the constraints that should be satisfied by the solutions. In order to compare the performance of the proposed method in comparison with the other method, we use seven constrained test problems CF1 to CF7. As noted in Section 3, in our method, only the feasible solutions that satisfy all the constraints have the ability to insert to the external archive. Usually, an optimization method faces more challenges in solving constrained problems. Hence, the constraint handling method has an important effect in producing satisfactory results. All the constrained test problems used here have two objectives that should be minimized. The mathematical formulation of the constrained problems is given below:

$$CF1:\begin{cases} f_1(x)=x_1+\dfrac{2}{|J_1|}\displaystyle\sum_{j\in J_1}\left(x_j-x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}\right)^2, J_1=\{j|j \text{ is odd and } 2\le j\le n\} \\ f_2(x)=1-x_1+\dfrac{2}{|J_2|}\displaystyle\sum_{j\in J_2}\left(x_j-x_1^{0.5\left(1.0+\frac{3(j-2)}{n-2}\right)}\right)^2, J_2=\{j|j \text{ is even and } 2\le j\le n\} \end{cases} \quad (23)$$

Subject to: $f_1+f_2-a\left|\sin\left[N\pi(f_1-f_2+1)\right]\right|-1\ge 0$ where N is an integer and $a\ge\dfrac{1}{2N}$.

$$CF2:\begin{cases} f_1=x_1+\dfrac{2}{|J_1|}\displaystyle\sum_{j\in J_1}\left(x_j-\sin\left(6\pi x_1+\frac{j\pi}{n}\right)\right)^2, J_1=\{j|j \text{ is odd and } 2\le j\le n\} \\ f_2=1-\sqrt{x_1}+\dfrac{2}{|J_2|}\displaystyle\sum_{j\in J_2}\left(x_j-\cos\left(6\pi x_1+\frac{j\pi}{n}\right)\right)^2, J_2=\{j|j \text{ is even and } 2\le j\le n\} \end{cases} \quad (24)$$

Subject to: $\frac{t}{1+e^{4|t|}} \geq 0$ where $t = f2 + \sqrt{f_1} - a\sin\left[N\pi\left(\sqrt{f_1} - f_2 + 1\right)\right] - 1$.

$$CF3: \begin{cases} f_1 = x_1 + \frac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right), J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}, J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\} \\ f_2 = 1 - x_1^2 + \frac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j\pi}{\sqrt{j}}\right) + 2\right), y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2,...,n \end{cases} \quad (25)$$

Subject to: $f_2 + f_1^2 - a\left|\sin\left[N\pi\left(f_1^2 - f_2 + 1\right)\right]\right| - 1 \geq 0$.

$$CF4: \begin{cases} f_1 = x_1 + \sum_{j\in J_1} h_j(y_j), J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}, J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\} \\ f_2 = 1 - x_1 + \sum_{j\in J_2} h_j(y_j), y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2,...,n, h_2(t) = \begin{cases} |t| & \text{if } t < \frac{3}{2}(1 - \frac{\sqrt{2}}{2}) \\ 0.125 + (t-1)^2 & \text{otherwise} \end{cases}, h_j(t) = t^2, \text{for j = 3,4,…,n} \end{cases} \quad (26)$$

Subject to: $\frac{t}{1+e^{4|t|}} \geq 0$ where $t = x_2 + \sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - 0.5x_1 + 0.25$.

$$CF5: \begin{cases} f_1 = x_1 + \sum_{j\in J_1} h_j(y_j), h_2(t) = \begin{cases} |t| & \text{if } t < \frac{3}{2}(1 - \frac{\sqrt{2}}{2}) \\ 0.125 + (t-1)^2 & \text{otherwise} \end{cases} \\ f_2 = 1 - x_1 + \sum_{j\in J_2} h_j(y_j), h_j(t) = 2t^2 - \cos(4\pi t) + 1, \text{for j = 3,4,…,n} \\ J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}, J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\}, y_j = \begin{cases} x_j - 0.8x_1\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_1 \\ x_j - 0.8x_1\sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_2 \end{cases} \end{cases} \quad (27)$$

Subject to: $x_2 - 0.8x_1\sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - 0.5x_1 + 0.25 \geq 0$.

$$CF6: \begin{cases} f_1 = x_1 + \sum_{j\in J_1} y_j^2, J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}, J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\} \\ f_2 = (1 - x_1)^2 + \sum_{j\in J_2} y_j^2, y_j = \begin{cases} x_j - 0.8x_1\cos\left(6\pi x_1 + \frac{j\pi}{n}\right) + 0.6x_1 & \text{if } j \in J_1 \\ x_j - 0.8x_1\sin\left(6\pi x_1 + \frac{j\pi}{n}\right) + 0.6x_1 & \text{if } j \in J_1 \end{cases} \end{cases} \quad (28)$$

Subject to: $x_2 - 0.8x_1\sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - sign\left(0.5(1-x_1) - (1-x_1)^2\right)\sqrt{\left|0.5(1-x_1) - (1-x_1)^2\right|} \geq 0$

$$x_4 - 0.8x_1\sin\left(6\pi x_1 + \frac{4\pi}{n}\right) - sign\left(0.25\sqrt{1-x_1} - 0.5(1-x_1)\right)\sqrt{0.25\sqrt{1-x_1} - 0.5(1-x_1)} \geq 0.$$

$$CF7: \begin{cases} f_1 = x_1 + \sum_{j\in J_1} h_j(y_j), J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}, J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\} \\ f_2 = (1 - x_1)^2 + \sum_{j\in J_2} h_j(y_j), y_j = \begin{cases} x_j - \cos\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_1 \\ x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_2 \end{cases}, h_2(t) = h_4(t) = t^2, h_j(t) = 2t^2 - \cos(4\pi t) + 1, \text{for j = 3,5,6,…,n} \end{cases} \quad (29)$$

Subject to: $x_2 - \sin\left(6\pi x_1 + \frac{2\pi}{n}\right) - sign\left(0.5(1-x_1) - (1-x_1)^2\right)\sqrt{\left|0.5(1-x_1) - (1-x_1)^2\right|} \geq 0$

$$x_4 - \sin\left(6\pi x_1 + \frac{4\pi}{n}\right) - sign\left(0.25\sqrt{1-x_1} - 0.5(1-x_1)\right)\sqrt{0.25\sqrt{1-x_1} - 0.5(1-x_1)} \geq 0.$$

The experimental results on the constrained test problem are shown in Table 2. The results represent that the proposed A-MOABC method has good performance on CF2, CF3 and CF5 and placed among the best algorithms for optimizing those constrained test problems. The presented algorithm demonstrates

competitive performance on CF1, CF4, CF6 and CF7. The Lexicographic ordering shows that the overall rank of the proposed algorithm among 9 algorithms for optimizing 7 constrained test problem is 3.

Table 2. The IDG statistics over CF1-CF7 test problems

Test Problem / Algorithm	CF1	CF2	CF3	CF4	CF5	CF6	CF7
A-MOABC	**0.01279**	**0.00282**	**0.07451**	**0.00892**	**0.02885**	**0.02512**	**0.03270**
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	1000000	0.15265	0.41275	0.14782	0.26049

e) The effect of colony size

The previous experiments showed that the proposed method provides competitive results compared to the other methods. It seems that the performance of A-MOABC may be affected by the colony size. In this experiment, the effect of colony size on the performance of the proposed method is considered. Here, the number of function evaluation is fixed at 300000. Hence, by changing the size of colony, the number of iterations is changed. The effect of colony size is represented in Fig. 3. The results show that the size of the colony has a positive effect on the performance of the proposed method. Based on the results, the proposed method with small colony has the worst performance. The competitive results are obtained when the size of the colony is set at range [50, 90]. The performance of the method decreases when the colony size is set at 100. It seems that the performance of the proposed method decreases when the colony size is larger than 90. The lexicographic ordering shows that the best results are obtained when the colony size is set at 70.

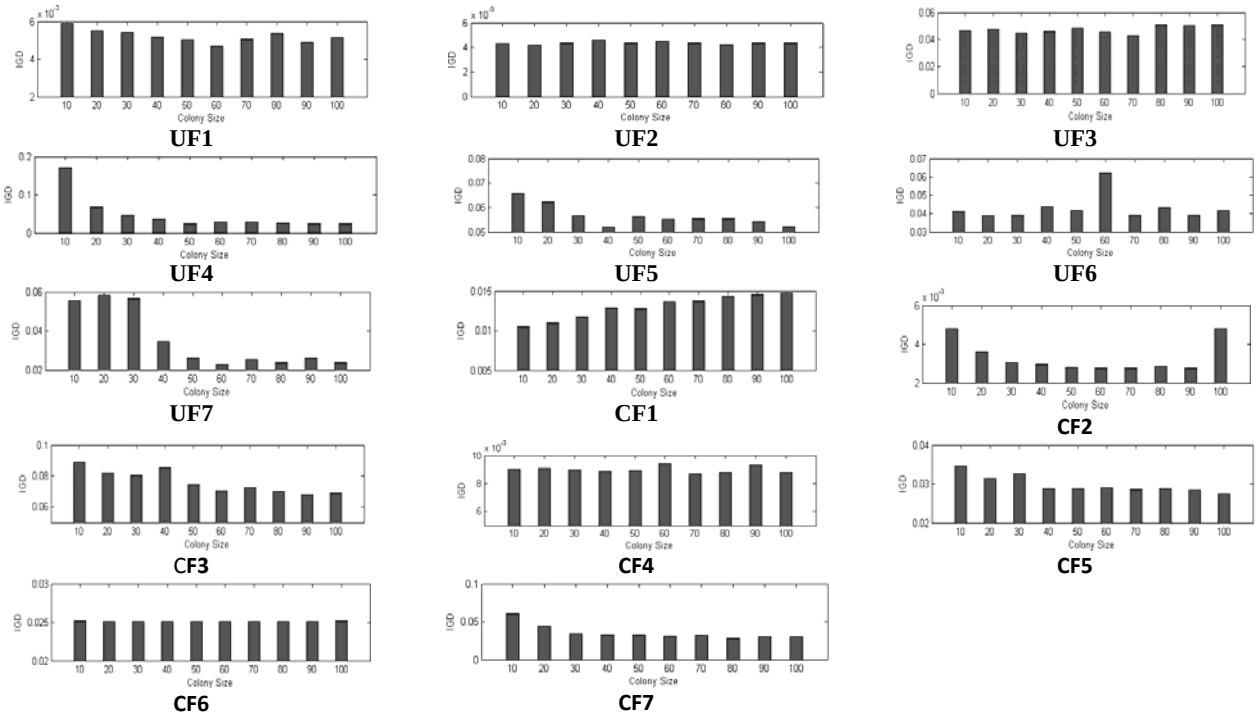
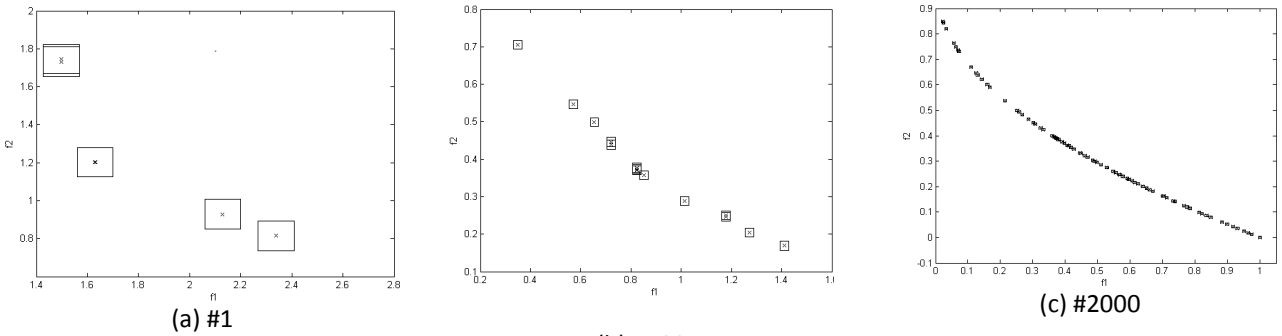


Fig. 3. The effect of colony size on the performance of A-MOABC

f) The effect of adaptive window

As mentioned in Section 3, the windowing mechanism is used by the employed bees in order to select their leaders from the archive members. The size of the window is adaptively adjusted throughout iterations. As described in Section 3, the size of window along the *i*-th dimension of solution space is adaptively adjusted based on $\boldsymbol{Max_i}$ and $\boldsymbol{Min_i}$ and number of non-dominated solutions in the archive. For UF1, the actual values for $\boldsymbol{Max_i}$ and $\boldsymbol{Min_i}$ are 1.0 and 0.0 respectively. The changes on the window size throughout iterations are presented in Fig. 4 for iterations 1, 100, and 2000. From the figure, it can be seen that the size of the window decreases as the algorithm proceeds. A window in a more crowded region contains more than one archive member. Hence, the fitnesses of those archive members decrease and the chance to select them as a leader for employed bees decreases too. In this way, the employed bees are encouraged to move to the sparse regions. The effect of this mechanism can be seen by comparing Fig. 4c with Fig. 4a and Fig. 4b. The sparse regions are covered as the algorithm proceeds. In general, it seems that the windowing mechanism helps the crowding distance mechanism to provide an archive in which the members are distributed uniformly.



(a) #1 (b) #100 (c) #2000

Fig. 4. The changes of window size throughout iterations

5. CONCLUSION

In this paper we present an adaptive multi-objective algorithm based on ABC. The performance of the proposed algorithms depends on a crowding distance technique to control proximity of the external archive members and a windowing mechanism for employed bees for selecting their leaders to guide them and make their trajectories better. The population in the proposed algorithm used different flying patterns which help the algorithm to maintain a trade-off between exploration and exploitation. The properties of the presented algorithm can make it a better method in front of other algorithms due to the mechanism for keeping diversity and also an adaptive efficient movement pattern. The experimental study showed that the presented algorithm is competitive compared to other algorithms investigated in this work. Also, the experiments showed that the crowding distance mechanism and windowing mechanism provide the ability for the algorithm to provide a Pareto-front where its members are distributed uniformly.

REFERENCES

1. Bahmanifirouzi, B., Farjah, E., Niknam, T. & Azad Farsani, E. (2012). A new hybrid HBMO-SFLA algorithm for multi-objective distribution feeder reconfigurtion problem considering distributed generator units. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, Vol. 36, No. E1, pp. 51-66.
2. Mohammadi, S. A., Akbari, R. & Mohammadi, S. H. (2012). An efficient method based on ABC for optimal multilevel thresholding. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, Vol. 36, No. E1, pp. 37-49.

3. Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *In IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 2, pp. 182–197.
4. Raquel, C. R. & Naval, Jr. P. C. (2005). An effective use of crowding distance in multiobjective particle swarm optimization. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO- 2005)*, pp. 257–264, Washington, DC, USA, June 2005. ACM Press.
5. Akbari, R. & Ziarati, K. (2012). Multi-objective bee swarm optimization. *International Journal of Innovative Computing, Information and Control (IJICIC)*, Vol. 8, No. 1 (B), pp. 715-726.
6. CoelloCoello, C. A., Pulido, G. T. & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 8, No.3, pp. 256-279.
7. Akbari, R., Hedayatzadeh, R., Ziarati, K. & Hasanizadeh, B. (2012). A multi-objective artificial bee colony algorithm. *Journal of Swarm and Evolutionary Computation*, Elsevier, Vol. 2, No. 1, pp. 39-52.
8. Zhou, A., Qu, B. Y., Li, H., Zhao, S. Z., Suganthan, P. N. & Zhang, Q. (2011). Multiobjective evolutionary algorithms: a survey of the state-of-the-art. *In Journal of Swarm and Evolutionary Computation*, Vol. 1, No.1, pp. 32-49.
9. Guliashki, V., Toshev, H. & Korsemov, C. (2009). Survey of evolutionary algorithms used in multiobjective optimization. *Journal of Problems of Engineering Cybernetics and Robotics*, Vol. 60, pp. 42-54.
10. Leong, W. F. & Yen, G. G. (2008). PSO-based multiobjective optimization with dynamic population size and adaptive local archives. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 38, No. 5.
11. Hedayatzadeh, R., Hasanizadeh, B., Akbari, R. & Ziarati, K. (2010). A multi-objective artificial bee colony for optimizing multi-objective problems. *3th International Conference on Advanced Computer Theory and Engineering, ICACTE*, Vol. 5, pp. 271-281.
12. Shivaie, M., Sepasian, M. S. & Sheikheleslami, M. K. (2011). Multi–objective transmission expansion planning using fuzzy–genetic algorithm. *Iranian Journal of Science and Technology, Transactions of Electrical Engineering*, Vol. 35, No. E2, pp. 141-159.
13. Mezura-Montes, E., Reyes-Sierra, M., & Coello, C. A. C. (2008). *Multi-objective optimization using differential evolution: a survey of the state-of-the-art*. Advances in differential evolution, Springer Berlin Heidelberg, pp. 173-196.
14. Reyes-Sierra, M. & Coello Coello, C. A. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, Vol. 2, No. 3, pp. 287–308.
15. Marler, R. T. & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Struct Multidisc Optim*, Vol. 26, pp. 369–395.
16. Coello, C. A. C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information systems*, Vol. 1, No. 3, pp. 129-156.
17. Coello Coello, C. A., Pulido, G. T. & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 3, pp. 256–279.
18. Ho, S. L., Shiyou, Y., Guangzheng, N., Lo, Edward W. C. & Wong, H. C. (2005). A particle swarm optimization based method for multiobjective design optimizations. *IEEE Transactions on Magnetics*, Vol. 41, No. 5, pp. 1756–1759.
19. Li, X. (2003). A non-dominated sorting particle swarm optimizer for multiobjective optimization. In Erick Cant´u-Paz et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2003)*, pp. 37–48. Springer. Lecture Notes in Computer Science Vol. 2723.
20. Raquel, V. & Naval, Jr. Prospero C. (2005). An effective use of crowding distance in multiobjective particle swarm optimization. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 257–264, Washington, DC, USA, ACM Press.
21. Karaboga, D. & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, Vol. 39, pp. 459–471.

22. Zou et al., (2011). Solving multiobjective optimization problems using artificial bee colony algorithm, *Discrete Dynamics in Nature and Society*, Vol. 2011, Article ID 569784, 37 pages, doi:10.1155/2011/569784.
23. Omkar, S. N., Senthilnath, J., Khandelwal, R., Narayana Naik, G. & Gopalakrishnan, S. (2011). Artificial bee colony (ABC) for multi-objective design optimization of composite structures. *Applied Soft Computing*, Vol. 11, Issue 1, pp. 489-499.
24. Zhang, H., Zhu, Y., Zou, W. & Yan, X. (2012). A hybrid multi-objective artificial bee colony algorithm for burdening optimization of copper strip production. *Applied Mathematical Modelling*, Vol. 36, Issue 6, pp. 2578–2591.
25. Maximiano, M. da S., Vega-Rodríguez, M. A., Gómez-Pulido, J. A. & Sánchez-Pérez, J. M. (2012). A new multiobjective artificial bee colony algorithm to solve a real-world frequency assignment problem. *Neural Computing & Applications*, doi: 10.1007/s00521-012-1046-7.
26. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P. N., Liu, W. & Tiwari, S. (2008). 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.
27. 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) 203-208.