Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 4935 – 4956

doi:10.3906/elk-1404-45

Balancing exploration and exploitation by using sequential execution cooperation between artificial bee colony and migrating birds optimization algorithms

Hasan MAKAS*, Nejat YUMUŞAK
Department of Computer Engineering, Faculty of Computer and Information Sciences, Sakarya University, Sakarya, Turkey

Received: 03.04.2014 • **Accepted/Published Online:** 08.10.2015 • **Final Version:** 06.12.2016

Abstract: The artificial bee colony (ABC) algorithm is a metaheuristic search method inspired by bees' foraging behaviour. With its global search ability in scout bee phase, it can easily escape from local optimum traps in the problem space. Therefore, it is good at exploration. The migrating birds optimization (MBO) algorithm is another recent metaheuristic search method. It simulates birds' V flight formation, which minimizes energy consumption during flight. The MBO algorithm achieves a good convergence to the global optimum by using its own unique benefit mechanism. That is, it has a good exploitation capability. This paper aimed to combine the good exploration property of the ABC algorithm and the good exploitation property of the MBO algorithm via a sequential execution strategy. In the proposed method, firstly, the ABC algorithm runs. This enables solutions to escape from local optimum traps and orientates them to the region in which the global optimum exists. Then the MBO algorithm runs. It performs a good convergence to the global optimum. In the proposed method, some variants of the ABC algorithm and some other well-known optimization algorithms were tested via benchmark functions. It was seen in the experiments that the proposed method gave competitive benchmark test results considering both success rates and convergence performances.

Key words: Migrating birds optimization algorithm, artificial bee colony algorithm, sequential execution, cooperation of algorithms, metaheuristics

1. Introduction

Solving a problem that includes many parameters is a critical issue in terms of optimization. Finding the global optimum of a complex problem may be too difficult or even impossible for the experts, even if the scientific definitions and constrains of the problem are determined using clear mathematical formulas. The additional constraints of the real world sometimes make problems much more complex. For example, in production systems, producers may be obliged to reduce consumption of a specific material from time to time due to its unavailability or a lack of inventory. In these cases, one or more parameters of the corresponding problem may need to be fixed to certain values. Thus, it is most probably impossible to reach the global optimum for this problem. Therefore, some other suboptimal solutions are searched for via algorithms for these cases.

Metaheuristic algorithms are designed to solve a wide range of optimization problems, including those with many parameters in industry and services, in areas ranging from finance to production management and engineering. The prefix *meta* represents that these algorithms are high level heuristics, and they are generally applied to problems for which there is no specific method for solving [1]. Philosophically, they do not have to

*Correspondence: hasanmakas@gmail.com

adapt to the problem system in any depth. More simply, metaheuristics are interested in neither the structural appearance of the problem system nor its design purpose. Therefore, they are not problem specific and can be applied to all of the problems that need optimization. Although problem specific complete methods give more optimal solutions, they need exponential computation times for the worst problem cases. Hence, metaheuristics, which are approximate methods, have been getting more and more attention over the last 20 years in order to solve both combinatorial and numerical optimization problems [2].

Metaheuristics can be defined as iterative methods that mimic the exploitation and exploration behaviours of some agents. Most of them are neighborhood search methods that are inspired by nature. They constitute a large and important class among improvement algorithms. A metaheuristic explores the problem space globally and searches in the neighborhoods of the existing solutions locally to get new and better solutions. There should be a fine balance between local intensive exploitation and global exploration [3]. If an optimization algorithm is good at exploration but poor at exploitation, it can escape from local optimums successfully but it cannot achieve a good convergence to the global optimum. On the other hand, if an optimization algorithm is good at exploitation but poor at exploration, it is weak at escaping from local optimums. Therefore, it has a low chance of finding a global optimum if the corresponding problem has many local optimums. However, if it manages to escape from local optimums, it can perform well at converging towards a global optimum.

Some of the most popular metaheuristic algorithms that have been introduced by researchers so far are as follows. The genetic algorithm (GA) [4], the simulated annealing (SA) algorithm [5], the tabu search (TS) algorithm [6], the ant colony optimization (ACO) algorithm [7], the particle swarm optimization (PSO) algorithm [8], the differential evolution (DE) algorithm [9], the harmony search (HS) algorithm [10], the monkey search (MS) algorithm [11], the ABC algorithm [12], the firefly algorithm (FA) [13], the intelligent water drops (IWD) algorithm [14], the cuckoo search (CS) algorithm [15,16], the bat algorithm (BA) [17,18] and the MBO algorithm [19]. In parallel with these studies many researchers have developed new methodologies based on the existing algorithms, such as modified hybrid forms [20–32] and parallel running methods [33–38], with the aim of getting better optimization performances.

Normally, the ABC algorithm is good at exploration, but poor at exploitation [39–51]. Conversely, the MBO algorithm is good at exploitation, but needs to be enhanced in exploration [33]. In this study, we proposed a sequential execution methodology for the ABC and MBO algorithms to perform better optimization by balancing their exploration and exploitation characteristics. More simply, we combined the good exploration property of the ABC algorithm and the good exploitation property of the MBO algorithm.

Using 10 different benchmark test functions in experiments, we tested how well the proposed method achieves optimization, and we compared its test results with the results of the MBO, ABC, GA, PSO, and DE algorithms and 3 variants of the ABC algorithm. The paper is organized as follows. Section 2 gives the theoretical backgrounds of the ABC and MBO algorithms, describes the proposed method, introduces experimental setups, and explains the philosophy of benchmark test performance evaluation. Section 3 gives the experimental results and discusses them. Section 4 concludes the whole study and proposes future work.

2. Methods and background

2.1. The ABC algorithm

The ABC algorithm is inspired by bees' foraging behaviours. Possible solutions of the optimization problem are represented by food sources and the fitness of each solution is represented by its nectar amount. Half of the colony is considered to be employed bees and the remaining half of the colony is considered to be onlooker

bees [52]. Each of the employed bees is assigned to 1 different food source. Therefore, the total number of food sources is equal to half of the colony size. There are 3 calculation phases in iterations as seen in Figure 1 [53]: employed bee phase, onlooker bee phase, and scout bee phase.

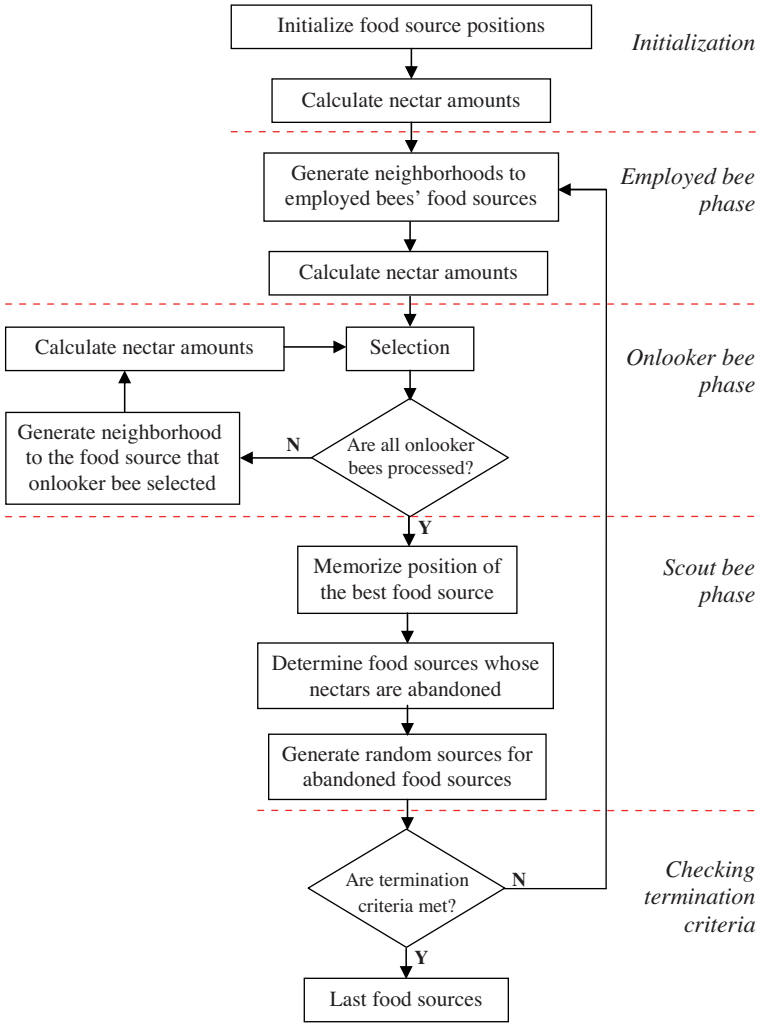

Figure 1. Flow diagram of the artificial bee colony algorithm [53].

At the beginning of the algorithm, it generates randomly distributed initial solutions to food sources by using

$$x_{ij} = x_j^{min} + rand. \left(x_j^{max} - x_j^{min}\right), \tag{1}$$

where x_{ij} is the position of the *ith* solution in the *jth* dimension, x_j^{min} and x_j^{max} are the minimum and maximum limit values for the *jth* dimension and rand is a uniform random number ranging from 0 to 1. After completing the food source initialization, the algorithm phases are run until the termination criteria are met.

In the employed bee phase, employed bees try to improve their solutions by generating new neighbor solutions via

$$\hat{x}_{ij} = x_{ij} + \varphi \cdot \left(x_{ij} - x_{kj}\right), \tag{2}$$

where x_{ij} is the position of the *ith* solution in the *jth* dimension, k is a randomly selected index value different from i, x_{kj} is the position of the *kth* solution in the *jth* dimension, ϕ is a random number ranging from –1 to 1, and $\hat{x}_{ij}$ is a generated new neighbor position in the *jth* dimension for the *ith* solution. If the generated neighbor $\hat{x}_{ij}$ exceeds a predefined space limit for the problem, it is shifted to the limit value exceeded. This shifting operation is also valid for all of the algorithms given in this paper. The fitness value of a source is

$$Fitness_i = \begin{cases} 1/(1+f_i), f_i \geq 0 \\ 1 + abs\left(f_i\right), f_i < 0 \end{cases}, \tag{3}$$

where f_i and $Fitness_i$ are cost and fitness values of the *ith* source, respectively. A greedy selection based on fitness values is employed for the selection between existing food sources and generated sources. Then selection probabilities by onlooker bees for updated sources are calculated by

$$p_i = Fitness_i \Big/ \sum_{j=1}^{SN} Fitness_j, \tag{4}$$

where p_i is the selection probability by onlooker bees for the *ith* source and SN is the total number of food sources.

In the onlooker bee phase, onlooker bees make their selections depending on the corresponding p_i probabilities by using a probabilistic selection mechanism, such as roulette wheel. Each onlooker generates a neighborhood to its selection by using Eq. (2), and it makes a greedy selection between the selected source and generated new source in order to improve its selection.

A failure counter is used for each source during implementations. The failure counter of the corresponding source is increased by 1 if no improvement is achieved after completing a neighborhood creation. Otherwise, it is set to zero. In the scout bee phase, whenever a failure counter exceeds a predefined limit value, the employed bee of that source is transformed into a scout bee. A scout bee makes a global search to find a new food source via Eq. (1). Then it is retransformed into an employed bee, and the corresponding failure counter is set to zero. The limit value for the failure counters is an important issue to be focused on. It can be determined by using Eq. (5) which depends on the problem dimension and the colony size [53].

$$FC_{Limit} = SN * D, \tag{5}$$

where SN is the total number of food sources, which is equal to half the colony size, and D is the problem dimension.

2.2. The MBO algorithm

The MBO algorithm is inspired by the V flight formation of birds like in Figure 2. A V shaped flight style gives rise to induced drag reduction. Therefore, it is an effective formation for birds to save energy [19]. Researchers have reported that each bird can achieve a reduction in induced drag as large as 65% in a V-shaped flight formation consisting of 25 members. This induced drag reduction results in a range increase of about 70% [54].

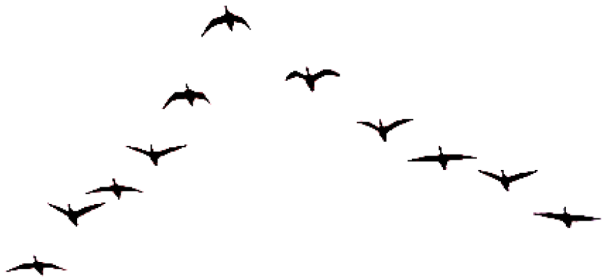

Figure 2. A typical V flight formation of birds.

The bird in Figure 3 is shown in the direction of flight. When the birds are in flight, a pair of vortices is created due to their wing movements as seen in Figure 3 [19,54]. Accordingly, trailing tip vortices from the left and the right wingtips rotate in clockwise and counterclockwise directions respectively [19,54]. Vortices create downwash and upwash regions for the birds flying behind. Since downwash increases the induced drag on a wing in flight, it is undesirable. However, upwash is beneficial since it decreases the induced drag on a wing in flight. For this reason, all of the birds except for the leader tend to locate in upwash regions of vortices, and they form a V-shaped formation. Thus, they get the benefit of upwashes and reduce their energy consumption. As a result, the leader bird in a V formation expends the most energy and the others get upwash benefit coming from the birds in front [19]. The parameters of the MBO algorithm are given in Table 1 and its flow diagram is given in Figure 4.

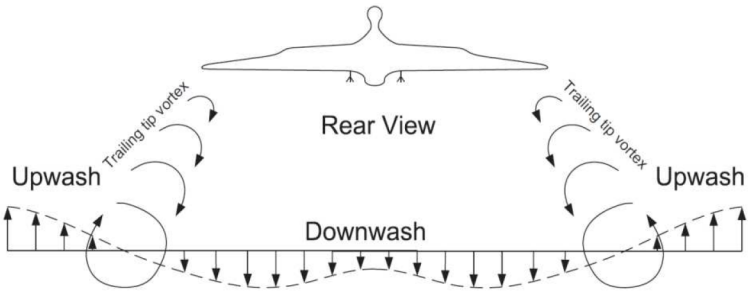

Figure 3. Upwash and downwash regions generated by trailing tip vortices [19,54].

Table 1. Parameters of the migrating birds optimization algorithm.

Parameter	Description
n	Number of solutions (flock size)
k	Total number of neighbor solutions to be considered (inverse of the speed)
x	Number of neighbor solutions to be shared with the next solution (upwash benefit)
m	Number of tours to change the leader
K	Maximum iteration or tour number

By sharing solutions, the MBO algorithm simulates the benefit mechanism in a real bird flock. Firstly, the population is initialized by using Eq. (1). After completing initialization, one of the solutions is chosen as leader and all of the solutions are placed in a hypothetical V formation arbitrarily. Starting with the first solution, which corresponds to the leader bird, and progressing on the lines towards the tails, the MBO algorithm aims to improve each solution by using its neighbor solutions [19].

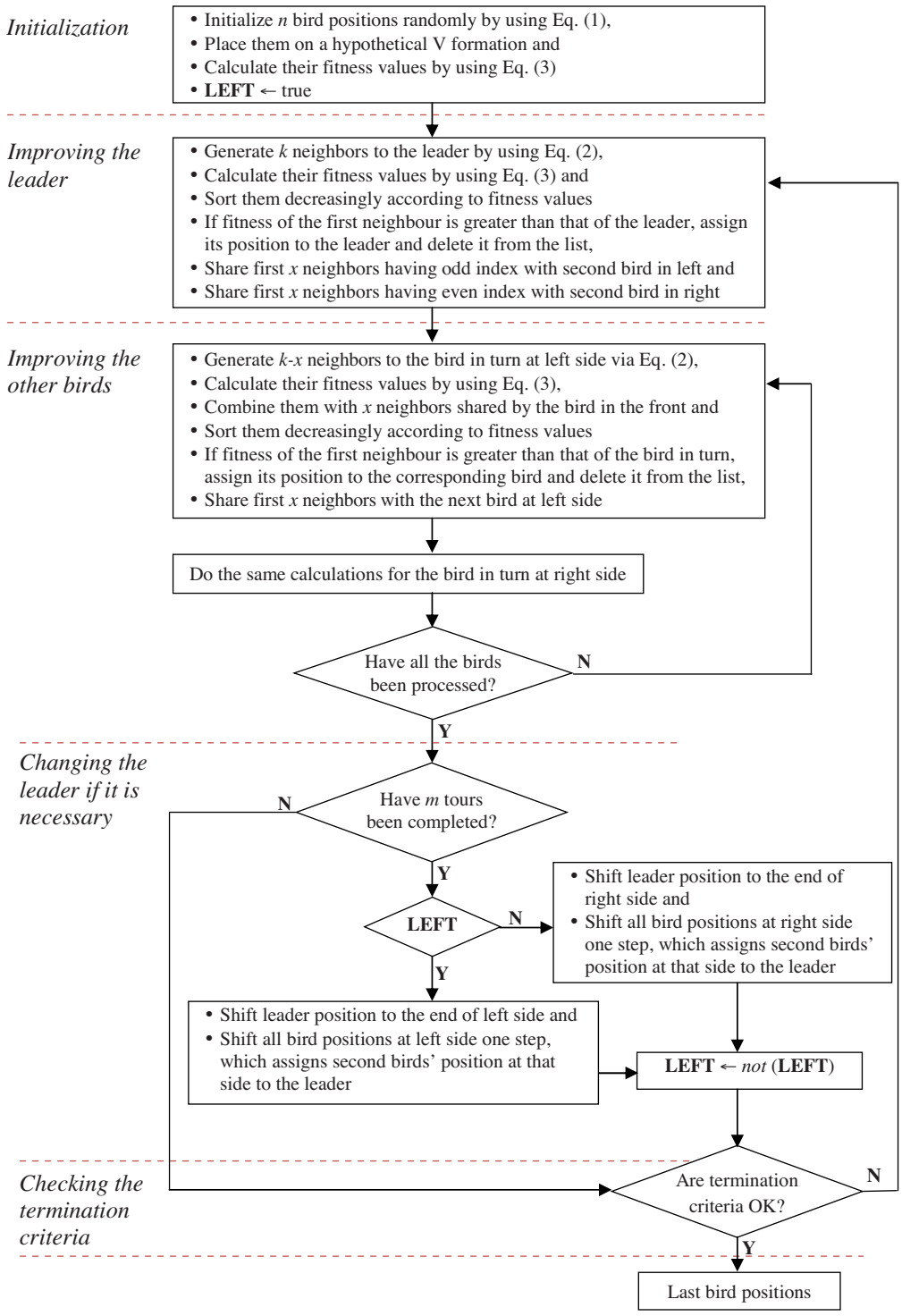

Figure 4. Flow diagram of the migrating birds optimization algorithm.

An improvement of the leader bird is performed in the second step. To do this, k neighbors are generated and their fitness values are calculated. The method given in Eq. (2) can be used for neighborhood creation. After creating the neighbors, if the fitness of the best neighbor solution is better than the fitness of the leader,

the position of this neighbor solution is assigned to the leader solution to improve its quality. Then $2\,x$ unused best neighbor solutions are shared with 2 birds in the second row.

Improvements of the other birds are performed in the third step. For each bird in turn, $(k-x)$ neighbor solutions are generated, and their fitness values are calculated. These generated neighbors, and x unused best neighbors coming from the bird in front, are combined to get $k$neighbors for the corresponding bird. Among these k neighbor solutions, if the fitness of the best neighbor solution is better than the fitness of the corresponding bird, the position of this neighbor solution is assigned to the corresponding solution to improve its quality. Then x unused best solutions are shared with the next bird. One iteration ends after completing improvement trials for all of the birds. Calculations in first 2 steps show that the leader bird spends the most energy (the most neighborhood creation), but the birds in other positions get a benefit from the birds in front (neighborhood sharing) and spend less energy.

In the fourth step, if the leader bird completes a predefined number of iterations (m), then it is thought to be tired and changed. Changing the leader is performed by shifting it to the end of one side of the hypothetical V formation and assigning the second solution at that side to the leader position. It should be noted that if the leader changing operation is performed on one side (e.g., left leg of V formation), the next leader changing operation should be performed on the opposite side (e.g., right leg of V formation). We chose $m = 10$ as recommended in [19]. Steps from step 2 to step 4 are repeated until the predefined termination criteria are satisfied. The algorithm stops and gives its best solution as the resulting overall solution.

Parameters k and x have significant effects on algorithm performance. Therefore, they should be tuned appropriately. Parameter k is inversely proportional to flock flight speed. For small values of k, the flock is assumed to be flying at higher speeds. Higher speeds enable the algorithm to reduce the overall execution time. This is advantageous for low dimensional problems. However, if the parameter k is increased, the search depth of the algorithm increases. For high dimensional problem cases, although the execution time increases, k may need to be increased in order to get better solutions. The parameter x represents upwash benefits of trailing tip vortices in a real bird flock. The benefit mechanism of the MBO algorithm is defined as the number of good neighborhoods coming from the predecessor solution. High values of x may cause solutions to be more similar. A premature convergence may then happen [19]. In our implementations, we chose $x = 1$ to prevent the algorithm from premature convergence, and we chose $k = 3$ to reduce the algorithm execution time. The MBO algorithm makes $k+$ $(population\_\ size - 1) \times (k - x)$ fitness calculations for 1 cycle. On the other hand, the total number of fitness calculations for 1 cycle in other algorithms is equal to the population size. Therefore, the total number of fitness calculations for the MBO algorithm by using our k and x choices is approximately equal to 2 times that for the other algorithms. Hence, we set total number of iterations in the MBO algorithm to half of that in other algorithms in order to make a fair comparison between the algorithms.

2.3. Proposed method

Exploitation and exploration are 2 important properties of an optimization algorithm and there should be a fine balance between them. As given in Table 2, the exploration characteristic determines how well an algorithm performs the global search. On the other hand, the exploitation characteristic determines how well it performs a search on local regions and how close it can converge to minimums.

Logically, if an algorithm is good at exploitation it can converge to optimal points as many times as possible. Diversity of the population is reduced during these convergences. That is, the members of population become more similar. However, some suboptimal points may be converged upon instead of a global optimal.

Table 2. Exploration and exploitation balance in an optimization algorithm.

Exploration	Exploitation	Algorithm characteristic
Good	Good	Firstly, it finds the global optimum region easily. Then it performs a good convergence to the global optimum point as desired. Its performance is generally independent from initial population and problem characteristics.
Good	Poor	It finds the global optimum region easily, but it cannot perform a good convergence to the global optimum point. Its performance is generally independent from initial population and problem characteristics.
Poor	Good	It may not able to find the region including the global optimum. If it finds global optimum region, it can perform a good convergence to the global optimum. Its performance is highly dependent on the diversity of initial population.
Poor	Poor	It is not wanted and not meaningful for an optimization algorithm.

Unfortunately, for these cases, the algorithm cannot escape from local optimum traps. Some algorithms have several methods to prevent themselves from reaching this desperate situation. For instance, the GA uses mutation and the ABC algorithm uses scout bees to escape from local optimums. Both mutation and scout bee applications force the algorithm to perform random global exploration and to increase population diversity. Thus, the algorithm gets a new chance to escape from local optimums.

The ABC algorithm is easy to implement and has few parameters. These are population size and maximum iteration cycle. It is good at exploration owing to the use of scout bees for food sources whose nectars are abandoned. That is, if a solution cannot be improved anymore, improvement trials on it are stopped and a random search is performed.

On the other hand, the MBO algorithm gives competitive results to optimization problems and has 2 important parameters to be tuned for successful optimizations. These are k and x as mentioned in Section 2.2. Briefly, parameter k is related to local search depth and parameter x is related to benefit usage rate. In short, there is no powerful mechanism that supports or controls a randomized global search in the MBO algorithm.

In our proposed method, it is aimed to use the good characteristics of the ABC and MBO algorithms sequentially. Since we used the ABC and MBO algorithms sequentially, we called this method the sequential ABC and MBO (SABCMBO) algorithm. The method is illustrated in Figure 5, schematically. The reason why the ABC algorithm is used firstly is that even if the ABC algorithm cannot perform a good convergence to the global optimum, it manages to direct the initial solution set to the region in which the global optimum exists. After the ABC algorithm completes the optimization, the MBO algorithm uses the resultant solution set as its initial solution set. Since the MBO algorithm does not lose time (iterations) in exploring to find the region which includes the global optimum, it directly carries out convergence to the global optimum. To sum up, the ABC algorithm finds the region nearest to the global optimum and the MBO algorithm performs a good convergence to the global optimum. We used the same ABC and MBO parameters in experiments for the SABCMBO method. Experimental results given in Section 3 show that SABCMBO algorithm has competitive success rates in comparison with the ABC and MBO algorithms.

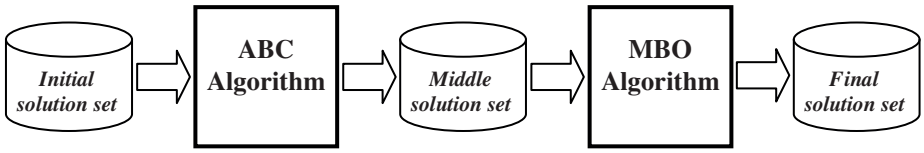

Figure 5. Schematic representation of the sequential artificial bee colony and migrating birds optimization method.

2.4. Benchmark functions and performance evaluation

A function that has only 1 optimum in the region to be searched is called unimodal. Otherwise, it is called multimodal. The performance of an optimization algorithm on a benchmark function depends on this characteristic of the benchmark function. We used 2, 5, 10, 30, and 50 dimensional versions of some unimodal and multimodal benchmark functions to test the optimization performances of the algorithms. These benchmark functions are given in Table 3.

Table 3. Benchmark functions used in experiments (n: problem dimension, C: characteristic, U: unimodal, M: multimodal).

Function	Definition	Ranges & parameters	Global minimum	C
Sphere [55]	$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-5.12,5.12]^n$	0	U
Rastrigin [56]	$f_2(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right]$	$[-5.12,5.12]^n$	0	M
Axis parallel hyper ellipsoid	$f_3(x) = \sum_{i=1}^{n} (i \cdot x_i^2)$	$[-5.12,5.12]^n$	0	U
Griewank [57]	$f_4(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$	0	M
Michalewicz [58]	$f_5(x) = -\sum_{i=1}^{n} \sin(x_i) \left[\sin\left(\frac{i \cdot x_i^2}{\pi}\right)\right]^{2m}$	$[0,\pi]^n$ $(m = 10)$	-1.801 $(n= 2)$ -4.687 $(n= 5)$:	M
Alpine [59]	$f_6(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	$[-10,10]^n$	0	M
Step [60]	$f_7(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	$[-100,100]^n$	0	U
Schwefel [61]	$f_8(x) = \sum_{i=1}^{n} \left[-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right]$	$[-500,500]^n$	$-418.9829n$	M
Ackley [62]	$f_9(x) = -a\exp\left(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)\right) + a + \exp(1)$	$[-32.\ 8, 32.\ 8]^n$ $a = 20,$ $b = 0.2,$ $c = \pi$	0	M
Schawefel	$f_{10}(x) = \max\{\ \lvert x_i \rvert\ ,\ 1 \le i \le n\}$	$[-100,100]^n$	0	U

It was aimed to make fair comparisons among the metaheuristics by using the same initial populations. Therefore, 100 different initial populations were created for each dimension by using Eq. (1), and they were stored in files. Each algorithm was executed 100 times for each dimension of each benchmark function. A new initial population from the stored files was selected for each execution. The algorithm parameters used in the implementations are given in Table 4, and average cost values of the final results from executions are given in Tables 5–9.

Table 4. Algorithm parameters.

Problem dimensions:		2	5	10	30	50
Error tolerance		0.005	0.01	0.011	0.5	0.6
Approximate fitness calculation		24,000	96,000	210,000	714,000	1,216,000
MBO	*Flock size (n)*	15	31	41	101	151
	Nb. to be considered (k)	3	3	3	3	3
	Nb. to be shared (x)	1	1	1	1	1
	Max cycle (K)	750	1500	2500	3500	4000
ABC	*Colony size (n)*	16	32	42	102	152
IABC	*Max cycle (K)*	1500	3000	5000	7000	8000
BSFABC						
GBGABC	*Colony size (n)*	16	32	42	102	152
	C constant	2	2	2	2	2
	Max cycle (K)	1500	3000	5000	7000	8000
PSO	*Swarm size (n)*	15	31	41	101	151
	Acceleration coef. (c)	1.494	1.494	1.494	1.494	1.494
	Max velocity limit (V_{max})	1	1	1	1	1
	Decreasing inertia (w)	$0.9 \rightarrow 0.4$	$0.9 \rightarrow 0.4$	$0.9 \rightarrow 0.4$	$0.9 \rightarrow 0.4$	$0.9 \rightarrow 0.4$
	Max cycle (K)	1500	3000	5000	7000	8000
DE	*Swarm size (n)*	15	31	51	101	151
	Amplification const. (F)	0.7	0.5	0.3	0.2	0.2
	Crossover constant (CR)	0.4	0.4	0.4	0.4	0.4
	Max cycle (K)	1500	3000	5000	7000	8000
GA	*Population size (n)*	16	32	42	102	152
	Mutation rate (CR)	0.05	0.05	0.05	0.05	0.05
	Max cycle (K)	1500	3000	5000	7000	8000
SABCMBO	*ABC part Colony size (n)*	16	32	42	102	152
	ABC part cycle (K_1)	750	1500	2500	3500	4000
	MBO part flock size (n)	15	31	41	101	151
	Nb. to be considered (k)	3	3	3	3	3
	Nb. to be shared (x)	1	1	1	1	1
	MBO part cycle (K_2)	375	750	1250	1750	2000

GA - genetic algorithm. PSO - particle swarm optimization algorithm. DE - differential evolution algorithm. ABC – artificial bee colony algorithm. MBO – migrating birds optimization algorithm. SABCMBO – sequential artificial bee colony migrating birds optimization algorithm. IABC - improved artificial bee colony algorithm. GBGABC - global best guided artificial bee colony algorithm. BSFABC - best so far artificial bee colony algorithm.

By using calculated average cost values, average absolute errors were calculated for the functions having a global minimum whose cost value is zero (except for f_5 and f_8), and average relative errors were calculated for the functions having a global minimum whose cost value is not zero (f_5 and f_8). Averages of these calculated errors were used as a performance indicator.

Reasonable tolerance values were determined depending on the problem dimensions, and these are given in Table 4. The error of each execution was compared with the corresponding tolerance value. Optimizations were assumed to be successful for errors lower than the determined tolerance values. The total numbers of successful optimizations among 100 executions were given as success rates in Tables 5–9. The averages of these success rates were used as another performance indicator in percentages.

Table 5. Test results for 2 dimensional benchmark functions (C: mean cost, E: mean error, R: success percentage and GM: global minimum).

		MBO	ABC	IABC	GBGABC	BSFABC	PSO	DE	GA	SABCMBO
f_1	C:	2.45E–306	1.08E–11	4.82E–15	5.82E–17	5.61E–17	2.51E–212	1.16E–235	2.49E–06	3.44E-173
	E:	2.45E–306	1.08E–11	4.82E–15	5.82E–17	5.61E–17	2.51E–212	1.16E–235	2.49E–06	3.44E-173
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_2	C:	1.39E–01	2.22E–02	1.55E–02	1.51E–03	0.00E+00	3.28E–01	5.97E–02	4.94E–04	0.00E+00
	E:	1.39E–01	2.22E–02	1.55E–02	1.51E–03	0.00E+00	3.28E–01	5.97E–02	4.94E–04	0.00E+00
$GM = 0$	R:	90	60	62	97	100	74	95	98	100
f_3	C:	5.36E–207	1.07E–10	3.49E–13	5.66E–17	5.97E–17	1.31E–212	4.88E–236	8.45E–07	1.30E-172
	E:	5.36E–207	1.07E–10	3.49E–13	5.66E–17	5.97E–17	1.31E–212	4.88E–236	8.45E–07	1.30E-172
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_4	C:	1.15E–02	4.07E–02	4.44E–02	2.78E–02	6.88E–17	2.50E–02	3.58E–03	1.47E–02	7.40E-05
	E:	1.15E–02	4.07E–02	4.44E–02	2.78E–02	6.88E–17	2.50E–02	3.58E–03	1.47E–02	7.40E-05
$GM = 0$	R:	34	4	6	5	100	32	58	20	100
f_5	C:	–1.80E+00	–1.80E+00	–1.80E+00	–1.23E+00	–1.23E+00	–1.72E+00	–1.80E+00	–1.80E+00	-1.80E+00
	E:	3.43E–03	1.68E–04	1.68E–04	3.16E–01	3.16E–01	4.43E–02	1.68E–04	1.62E–04	1.68E-04
$GM = –1.801$	R:	100	100	100	41	41	90	100	100	100
f_6	C:	6.83E–17	2.20E–04	2.59E–04	6.84E–05	7.36E–17	5.62E–05	8.33E–18	4.99E–05	3.66E-102
	E:	6.83E–17	2.20E–04	2.59E–04	6.84E–05	7.36E–17	5.62E–05	8.33E–18	4.99E–05	3.66E-102
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_7	C:	0.00E+00	3.35E–04	7.05E–05	5.59E–17	9.96E–12	0.00E+00	0.00E+00	1.17E–03	0.00E+00
	E:	0.00E+00	3.35E–04	7.05E–05	5.59E–17	9.96E–12	0.00E+00	0.00E+00	1.17E–03	0.00E+00
$GM = 0$	R:	100	99	100	100	100	100	100	94	100
f_8	C:	–8.17E+02	–8.38E+02	–8.38E+02	–8.38E+02	–7.24E+02	–6.84E+02	–8.03E+02	–1.59E+04	-8.38E+02
	E:	2.55E–02	2.27E–04	1.22E–04	3.75E–05	1.36E–01	1.84E–01	4.13E–02	1.80E+01	5.04E-06
$GM = –837.97$	R:	83	100	100	100	4	14	76	1	100
f_9	C:	1.21E–15	1.12E–02	7.17E–03	3.95E–05	8.88E–16	1.74E–15	8.88E–16	1.14E–02	8.88E-16
	E:	1.21E–15	1.12E–02	7.17E–03	3.95E–05	8.88E–16	1.74E–15	8.88E–16	1.14E–02	8.88E-16
$GM = 0$	R:	100	85	92	100	100	100	100	77	100
f_{10}	C:	8.84E–93	8.26E–02	8.42E–02	3.13E–03	7.91E–17	1.38E–106	2.39E–103	3.41E–02	2.13E-77
	E:	8.84E–93	8.26E–02	8.42E–02	3.13E–03	7.91E–17	1.38E–106	2.39E–103	3.41E–02	2.13E-77
$GM = 0$	R:	100	10	16	91	100	100	100	32	100
Mean error:		1.80E–02	1.58E–02	1.52E–02	3.49E–02	4.53E–02	5.81E–02	1.05E–02	1.81E+00	2.47E–05
Mean success rate:		90.7	75.8	77.6	83.4	84.5	81.0	92.9	72.2	100.0

GA - genetic algorithm. PSO - particle swarm optimization algorithm. DE - differential evolution algorithm. ABC – artificial bee colony algorithm. MBO – migrating birds optimization algorithm. SABCMBO – sequential artificial bee colony migrating birds optimization algorithm. IABC - improved artificial bee colony algorithm. GBGABC - global best guided artificial bee colony algorithm. BSFABC - best so far artificial bee colony algorithm.

Table 6. Test results for 5 dimensional benchmark functions (C: mean cost, E: mean error, R: success percentage and GM: global minimum).

		MBO	ABC	IABC	GBGABC	BSFABC	PSO	DE	GA	SABCMBO
f_1	C:	9.32E–35	8.24E–17	8.35E–17	8.11E–17	8.62E–17	0.00E+00	7.09E–252	1.76E–06	4.30E-74
	E:	9.32E–35	8.24E–17	8.35E–17	8.11E–17	8.62E–17	0.00E+00	7.09E–252	1.76E–06	4.30E-74
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_2	C:	1.39E–01	7.11E–17	0.00E+00	0.00E+00	0.00E+00	3.04E+00	0.00E+00	2.73E–04	0.00E+00
	E:	1.39E–01	7.11E–17	0.00E+00	0.00E+00	0.00E+00	3.04E+00	0.00E+00	2.73E–04	0.00E+00
$GM = 0$	R:	89	100	100	100	100	4	100	100	100
f_3	C:	1.34E–240	7.98E–17	8.61E–17	7.94E–17	7.97E–17	0.00E+00	3.37E–252	6.86E–06	1.56E-67
	E:	1.34E–240	7.98E–17	8.61E–17	7.94E–17	7.97E–17	0.00E+00	3.37E–252	6.86E–06	1.56E-67
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_4	C:	2.39E–02	4.62E–03	4.71E–03	4.41E–03	6.66E–17	2.95E–01	1.48E–04	2.56E–02	1.36E-03
	E:	2.39E–02	4.62E–03	4.71E–03	4.41E–03	6.66E–17	2.95E–01	1.48E–04	2.56E–02	1.36E-03
$GM = 0$	R:	29	94	96	89	100	1	100	8	100
f_5	C:	–4.66E+00	–4.69E+00	–4.69E+00	–4.68E+00	–4.69E+00	–4.15E+00	–4.68E+00	–4.69E+00	-4.69E+00
	E:	4.99E–03	1.40E–04	3.13E–04	9.75E–04	2.27E–04	1.15E–01	1.15E–03	1.33E–04	1.40E-04
$GM = –4.687$	R:	92	100	100	100	100	27	100	100	100
f_6	C:	6.71E–05	7.45E–14	4.10E–13	1.36E–10	1.44E–15	1.94E–09	1.26E–131	1.29E–04	9.29E-19
	E:	6.71E–05	7.45E–14	4.10E–13	1.36E–10	1.44E–15	1.94E–09	1.26E–131	1.29E–04	9.29E-19
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_7	C:	0.00E+00	8.43E–17	8.10E–17	8.22E–17	8.21E–17	0.00E+00	0.00E+00	5.37E–04	0.00E+00
	E:	0.00E+00	8.43E–17	8.10E–17	8.22E–17	8.21E–17	0.00E+00	0.00E+00	5.37E–04	0.00E+00
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_8	C:	–2.08E+03	–2.09E+03	–2.09E+03	–2.07E+03	–2.07E+03	–1.43E+03	–2.09E+03	–1.27E+04	-2.09E+03
	E:	7.92E–03	6.83E–06	6.89E–06	1.25E–02	9.87E–03	3.17E–01	2.83E–03	5.07E+00	6.89E-06
$GM = –2094.9$	R:	88	100	100	81	87	1	96	1	100
f_9	C:	1.65E–02	4.33E–15	4.37E–15	4.26E–15	8.88E–16	5.22E–15	8.88E–16	1.35E–02	3.62E-15
	E:	1.65E–02	4.33E–15	4.37E–15	4.26E–15	8.88E–16	5.22E–15	8.88E–16	1.35E–02	3.62E-15
$GM = 0$	R:	100	100	100	100	100	100	100	59	100
f_{10}	C:	1.84E–03	7.82E–04	3.75E–04	6.34E–16	1.01E–16	2.76E–162	7.30E–83	9.97E–02	2.29E-13
	E:	1.84E–03	7.82E–04	3.75E–04	6.34E–16	1.01E–16	2.76E–162	7.30E–83	9.97E–02	2.29E-13
$GM = 0$	R:	100	98	100	100	100	100	100	2	100
Mean error:		1.94E–02	5.55E–04	5.41E–04	1.79E–03	1.01E–03	3.77E–01	4.13E–04	5.21E–01	1.50E–04
Mean success rate:		89.8	99.2	99.6	97.0	98.7	63.3	99.6	67.0	100.0

GA - genetic algorithm. PSO - particle swarm optimization algorithm. DE - differential evolution algorithm. ABC – artificial bee colony algorithm. MBO – migrating birds optimization algorithm. SABCMBO – sequential artificial bee colony migrating birds optimization algorithm. IABC - improved artificial bee colony algorithm. GBGABC - global best guided artificial bee colony algorithm. BSFABC - best so far artificial bee colony algorithm.

Table 7. Test results for 10 dimensional benchmark functions (C: mean cost, E: mean error, R: success percentage and GM: global minimum).

		MBO	ABC	IABC	GBGABC	BSFABC	PSO	DE	GA	SABCMBO
f_1	C:	1.02E–08	1.35E–16	1.61E–16	1.55E–16	9.51E–17	4.03E–246	1.00E–203	2.47E–06	4.88E–20
	E:	1.02E–08	1.35E–16	1.61E–16	1.55E–16	9.51E–17	4.03E–246	1.00E–203	2.47E–06	4.88E–20
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_2	C:	6.99E–01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.86E+00	5.07E–02	5.45E–04	0.00E+00
	E:	6.99E–01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.86E+00	5.07E–02	5.45E–04	0.00E+00
$GM = 0$	R:	51	100	100	100	100	1	93	100	100
f_3	C:	1.61E–07	1.45E–16	1.30E–16	1.32E–16	9.42E–17	1.18E–209	6.16E–93	2.02E–05	2.73E–20
	E:	1.61E–07	1.45E–16	1.30E–16	1.32E–16	9.42E–17	1.18E–209	6.16E–93	2.02E–05	2.73E–20
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_4	C:	5.01E–02	5.69E–03	5.12E–03	4.43E–03	7.22E–17	6.84E–01	7.95E–04	5.13E–02	2.64E–03
	E:	5.01E–02	5.69E–03	5.12E–03	4.43E–03	7.22E–17	6.84E–01	7.95E–04	5.13E–02	2.64E–03
$GM = 0$	R:	11	75	76	81	100	1	100	5	93
f_5	C:	–9.57E+00	–9.66E+00	–9.66E+00	–9.66E+00	–9.66E+00	–7.89E+00	–9.65E+00	–9.66E+00	–9.66E+00
	E:	9.46E–03	1.56E–05	2.01E–05	4.95E–05	5.86E–05	1.83E–01	1.34E–03	1.54E–04	1.57E–05
$GM = –9.66$	R:	69	100	100	100	100	2	99	1	100
f_6	C:	1.98E–04	1.29E–15	4.62E–16	6.65E–10	8.60E–14	1.65E–08	5.61E–185	3.45E–04	6.80E–18
	E:	1.98E–04	1.29E–15	4.62E–16	6.65E–10	8.60E–14	1.65E–08	5.61E–185	3.45E–04	6.80E–18
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_7	C:	8.60E–04	1.51E–16	1.25E–16	1.37E–16	1.65E–16	0.00E+00	0.00E+00	9.47E–04	1.43E–20
	E:	8.60E–04	1.51E–16	1.25E–16	1.37E–16	1.65E–16	0.00E+00	0.00E+00	9.47E–04	1.43E–20
$GM = 0$	R:	100	100	100	100	100	100	100	100	100
f_8	C:	–4.10E+03	–4.19E+03	–4.19E+03	–4.16E+03	–4.19E+03	–2.38E+03	–4.18E+03	–1.43E+04	–4.19E+03
	E:	2.16E–02	6.89E–06	6.89E–06	7.35E–03	5.72E–04	4.32E–01	2.28E–03	2.41E+00	6.89E–06
$GM = –4189.8$	R:	46	100	100	77	99	1	93	1	100
f_9	C:	1.04E–01	7.89E–15	8.31E–15	7.39E–15	8.88E–16	2.31E–02	3.87E–15	1.31E–02	6.15E–15
	E:	1.04E–01	7.89E–15	8.31E–15	7.39E–15	8.88E–16	2.31E–02	3.87E–15	1.31E–02	6.15E–15
$GM = 0$	R:	93	100	100	100	100	99	100	47	100
f_{10}	C:	6.80E–01	5.89E–11	8.11E–08	3.18E–15	3.55E–14	2.42E–62	1.72E–01	1.45E–01	7.91E–06
	E:	6.80E–01	5.89E–11	8.11E–08	3.18E–15	3.55E–14	2.42E–62	1.72E–01	1.45E–01	7.91E–06
$GM = 0$	R:	86	100	100	100	100	100	83	1	100
Mean error:		1.57E–01	5.71E–04	5.15E–04	1.18E–03	6.31E–05	1.12E+00	2.27E–02	2.62E–01	2.67E–04
Mean success rate:		75.6	97.5	97.6	95.8	99.9	60.4	96.8	55.5	99.3

GA - genetic algorithm. PSO - particle swarm optimization algorithm. DE - differential evolution algorithm. ABC – artificial bee colony algorithm. MBO – migrating birds optimization algorithm. SABCMBO – sequential artificial bee colony migrating birds optimization algorithm. IABC - improved artificial bee colony algorithm. GBGABC - global best guided artificial bee colony algorithm. BSFABC - best so far artificial bee colony algorithm.

Table 8. Test results for 30 dimensional benchmark functions (C: mean cost, E: mean error, R: success percentage and GM: global minimum).

		MBO	ABC	IABC	GBGABC	BSFABC	PSO	DE	GA	SABCMBO
f_1	C :	1.03E–15	5.09E–16	5.22E–16	5.44E–16	3.10E–16	1.73E–64	4.11E–210	1.07E–05	6.93E-19
	E :	1.03E–15	5.09E–16	5.22E–16	5.44E–16	3.10E–16	1.73E–64	4.11E–210	1.07E–05	6.93E-19
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_2	C :	7.78E–01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.83E+01	7.98E–02	2.25E–03	0.00E+00
	E :	7.78E–01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	3.83E+01	7.98E–02	2.25E–03	0.00E+00
$GM = 0$	R :	56	100	100	100	100	1	93	100	100
f_3	C :	8.23E–05	5.05E–16	5.03E–16	5.19E–16	3.14E–16	2.83E–68	5.80E–26	1.64E–04	4.77E-19
	E :	8.23E–05	5.05E–16	5.03E–16	5.19E–16	3.14E–16	2.83E–68	5.80E–26	1.64E–04	4.77E-19
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_4	C :	4.50E–03	9.88E–17	9.21E–17	7.66E–17	7.40E–05	2.11E–02	0.00E+00	2.24E–02	0.00E+00
	E :	4.50E–03	9.88E–17	9.21E–17	7.66E–17	7.40E–05	2.11E–02	0.00E+00	2.24E–02	0.00E+00
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_5	C :	–2.94E+01	–2.96E+01	–2.96E+01	–2.96E+01	–2.96E+01	–2.21E+01	–2.55E+01	–2.96E+01	-2.96E+01
	E :	1.61E–02	2.24E–02	2.24E–02	2.25E–02	2.18E–02	2.37E–01	1.20E–01	2.23E–02	2.24E-02
$GM = –28.98$	R :	100	100	100	100	100	100	100	100	100
f_6	C :	1.45E–04	6.75E–16	6.48E–16	4.76E–10	3.30E–14	4.44E–02	6.04E–05	1.21E–03	2.16E-15
	E :	1.45E–04	6.75E–16	6.48E–16	4.76E–10	3.30E–14	4.44E–02	6.04E–05	1.21E–03	2.16E-15
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_7	C :	2.10E–03	5.21E–16	5.11E–16	5.35E–16	5.00E–16	2.58E–32	0.00E+00	4.48E–03	3.31E-19
	E :	2.10E–03	5.21E–16	5.11E–16	5.35E–16	5.00E–16	2.58E–32	0.00E+00	4.48E–03	3.31E-19
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_8	C :	–1.25E+04	–1.26E+04	–1.26E+04	–1.26E+04	–1.26E+04	–5.95E+03	–1.26E+04	–5.33E+04	-1.26E+04
	E :	6.50E–03	3.87E–05	3.87E–05	3.87E–05	3.87E–05	5.27E–01	3.19E–04	3.24E+00	3.87E-05
$GM = –12569$	R :	100	100	100	100	100	29	100	1	100
f_9	C :	2.49E–14	3.25E–14	3.33E–14	3.04E–14	2.37E–14	1.46E–01	4.51E–15	1.52E–02	2.38E-14
	E :	2.49E–14	3.25E–14	3.33E–14	3.04E–14	2.37E–14	1.46E–01	4.51E–15	1.52E–02	2.38E-14
$GM = 0$	R :	100	100	100	100	100	90	100	100	100
f_{10}	C :	4.56E+00	6.92E–02	5.55E–02	7.53E–04	1.53E+00	9.18E–06	5.78E–01	4.55E–01	3.59E-01
	E :	4.56E+00	6.92E–02	5.55E–02	7.53E–04	1.53E+00	9.18E–06	5.78E–01	4.55E–01	3.59E-01
$GM = 0$	R :	72	100	100	100	6	100	74	71	83
Mean error:		5.37E–01	9.17E–03	7.80E–03	2.32E–03	1.55E–01	3.93E+00	7.78E–02	3.76E–01	3.81E–02
Mean success rate:		92.8	100.0	100.0	100.0	90.6	82.0	96.7	87.2	98.3

GA - genetic algorithm. PSO - particle swarm optimization algorithm. DE - differential evolution algorithm. ABC – artificial bee colony algorithm. MBO – migrating birds optimization algorithm. SABCMBO – sequential artificial bee colony migrating birds optimization algorithm. IABC - improved artificial bee colony algorithm. GBGABC - global best guided artificial bee colony algorithm. BSFABC - best so far artificial bee colony algorithm.

Table 9. Test results for 50 dimensional benchmark functions (C: mean cost, E: mean error, R: success percentage and GM: global minimum).

		MBO	ABC	IABC	GBGABC	BSFABC	PSO	DE	GA	SABCMBO
f_1	C :	3.77E–06	9.32E–16	9.32E–16	9.78E–16	7.79E–16	8.07E–30	9.47E–189	1.17E–04	5.90E-18
	E :	3.77E–06	9.32E–16	9.32E–16	9.78E–16	7.79E–16	8.07E–30	9.47E–189	1.17E–04	5.90E-18
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_2	C :	6.87E–01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.36E+01	1.99E–02	2.03E–02	0.00E+00
	E :	6.87E–01	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.36E+01	1.99E–02	2.03E–02	0.00E+00
$GM = 0$	R :	60	100	100	100	100	1	99	100	100
f_3	C :	2.58E–04	8.97E–16	9.08E–16	9.49E–16	7.50E–16	1.31E+00	1.78E–187	2.51E–03	5.68E-18
	E :	2.58E–04	8.97E–16	9.08E–16	9.49E–16	7.50E–16	1.31E+00	1.78E–187	2.51E–03	5.68E-18
$GM = 0$	R :	100	100	100	100	100	98	100	100	100
f_4	C :	1.81E–02	1.13E–16	9.66E–17	9.55E–17	1.31E–16	1.02E–02	0.00E+00	6.16E–02	0.00E+00
	E :	1.81E–02	1.13E–16	9.66E–17	9.55E–17	1.31E–16	1.02E–02	0.00E+00	6.16E–02	0.00E+00
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_5	C :	–4.94E+01	–4.96E+01	–4.96E+01	–4.96E+01	–4.94E+01	–3.67E+01	–3.19E+01	–4.96E+01	-4.96E+01
	E :	2.34E–02	2.74E–02	2.74E–02	2.73E–02	2.35E–02	2.40E–01	3.40E–01	2.65E–02	2.74E-02
$GM = –48.3$	R :	100	100	100	100	100	100	100	100	100
f_6	C :	7.39E–04	3.49E–15	2.40E–15	8.49E–09	7.38E–08	9.88E–04	7.61E–03	4.65E–03	7.77E-16
	E :	7.39E–04	3.49E–15	2.40E–15	8.49E–09	7.38E–08	9.88E–04	7.61E–03	4.65E–03	7.77E-16
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_7	C :	2.35E–04	9.09E–16	8.95E–16	9.28E–16	8.59E–16	2.73E–25	0.00E+00	4.39E–02	8.02E-18
	E :	2.35E–04	9.09E–16	8.95E–16	9.28E–16	8.59E–16	2.73E–25	0.00E+00	4.39E–02	8.02E-18
$GM = 0$	R :	100	100	100	100	100	100	100	100	100
f_8	C :	–2.08E+04	–2.09E+04	–2.09E+04	–2.09E+04	–2.09E+04	–9.03E+03	–2.09E+04	–8.92E+04	-2.09E+04
	E :	5.48E–03	6.89E–06	6.89E–06	6.89E–06	3.68E–03	5.69E–01	6.89E–06	3.26E+00	6.89E-06
$GM = –20949$	R :	100	100	100	100	100	73	100	1	100
f_9	C :	8.80E–03	6.00E–14	6.04E–14	5.47E–14	5.82E–14	1.90E–01	7.78E–15	3.96E–02	4.52E-14
	E :	8.80E–03	6.00E–14	6.04E–14	5.47E–14	5.82E–14	1.90E–01	7.78E–15	3.96E–02	4.52E-14
$GM = 0$	R :	100	100	100	100	100	86	100	100	100
f_{10}	C :	5.93E+00	6.68E–01	5.23E–01	1.58E–01	1.61E+01	5.90E–02	3.15E–01	1.76E+00	4.91E-01
	E :	5.93E+00	6.68E–01	5.23E–01	1.58E–01	1.61E+01	5.90E–02	3.15E–01	1.76E+00	4.91E-01
$GM = 0$	R :	63	49	69	100	1	100	91	1	80
Mean error:		6.67E–01	6.95E–02	5.51E–02	1.86E–02	1.62E+00	7.60E+00	6.83E–02	5.22E–01	5.18E–02
Mean success rate:		92.3	94.9	96.9	100.0	90.1	85.8	99.0	80.2	98.0

GA - genetic algorithm. PSO - particle swarm optimization algorithm. DE - differential evolution algorithm. ABC – artificial bee colony algorithm. MBO – migrating birds optimization algorithm. SABCMBO – sequential artificial bee colony migrating birds optimization algorithm. IABC - improved artificial bee colony algorithm. GBGABC - global best guided artificial bee colony algorithm. BSFABC - best so far artificial bee colony algorithm.

3. Results

The PSO, DE, GA, MBO, and ABC algorithms and 3 modified versions of the ABC algorithm were used in benchmark function tests. ABC variants are improved ABC (IABC) [51], global best guided ABC (GBGABC) [40], and best so far ABC (BSFABC) [50]. The benchmark function test results are given in Tables 5–9. The mean cost, mean error, and mean success rate values achieved by the algorithms for 2, 5, 10, 30, and 50 dimensional benchmark functions in these tables were calculated by using the methods mentioned in the previous section. The approximate fitness calculations mentioned in Section 2.2 are also given in Table 4 for each dimension. The results can be summarized as follows.

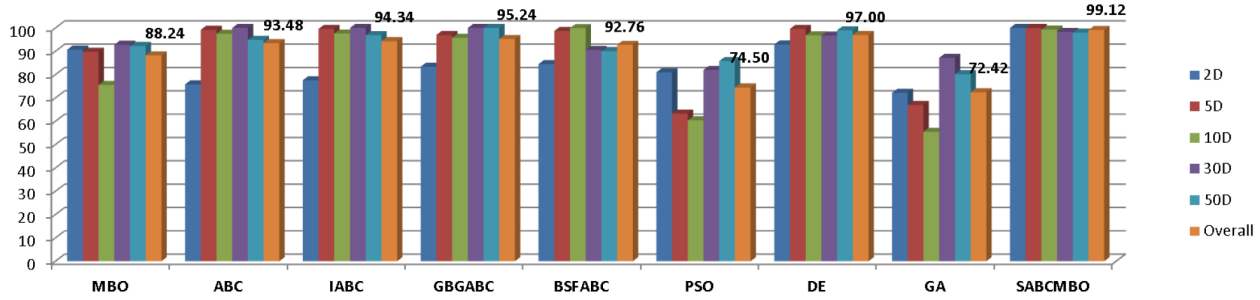

Figure 6. Success rate percentages of the algorithms over problem dimensions.

- The MBO algorithm achieves good convergence performance by using its own unique benefit mechanism. Its performance is highly competitive for unimodal benchmark functions, but it needs more exploration for multimodal benchmark functions. It is vulnerable to falling into local minimums for these functions. Data given in Tables 5–9 prove this fact. Its success rates are low for high dimensional multimodal functions because the number of local minimums increases proportional to the problem dimension. Flock size (n), neighborhood to be considered (k), neighborhood to be shared (x), and iteration (K) values used are given in Table 4 for each dimension.

- The ABC algorithm generally performs reasonable optimizations by giving high success rates. It has a good exploration characteristic that emerges in the scout bee phase. Although it finds reasonably optimal solutions to benchmark functions in all dimensions, it cannot achieve good convergences to the global minimums as desired. Therefore, its exploitation capability needs to be improved. Colony size (n) and iteration (K) values used are given in Table 4 for each dimension.

- The IABC, GBGABC, and BSFABC algorithms have characteristics similar to the ABC algorithm but improved. Their parameters for each dimension are given in Table 4.

- The PSO algorithm gives good optimization results for unimodal benchmark functions in all dimensions. Its exploitation property is good. However, results for the multimodal functions are not good. This is caused by its weak exploration property and its dependence on initial population quality. Population size (n), acceleration coefficient (c), inertia weight (w), maximum velocity limit ($V_{\max}$), and iteration (K) values used are given in Table 4 for each dimension.

- Since there is a fine balance between exploration and exploitation capabilities of the DE algorithm, it achieves reasonable optimizations in all dimensions. Population size (n), amplification constant (F), crossover rate (CR), and iteration (K) values used are given in Table 4 for each dimension.

- Focusing on mean error values, the solution quality of the GA can be said to be lower than the others. Both its exploration and exploitation capabilities are at a medium level. Thus, it falls into local minimums

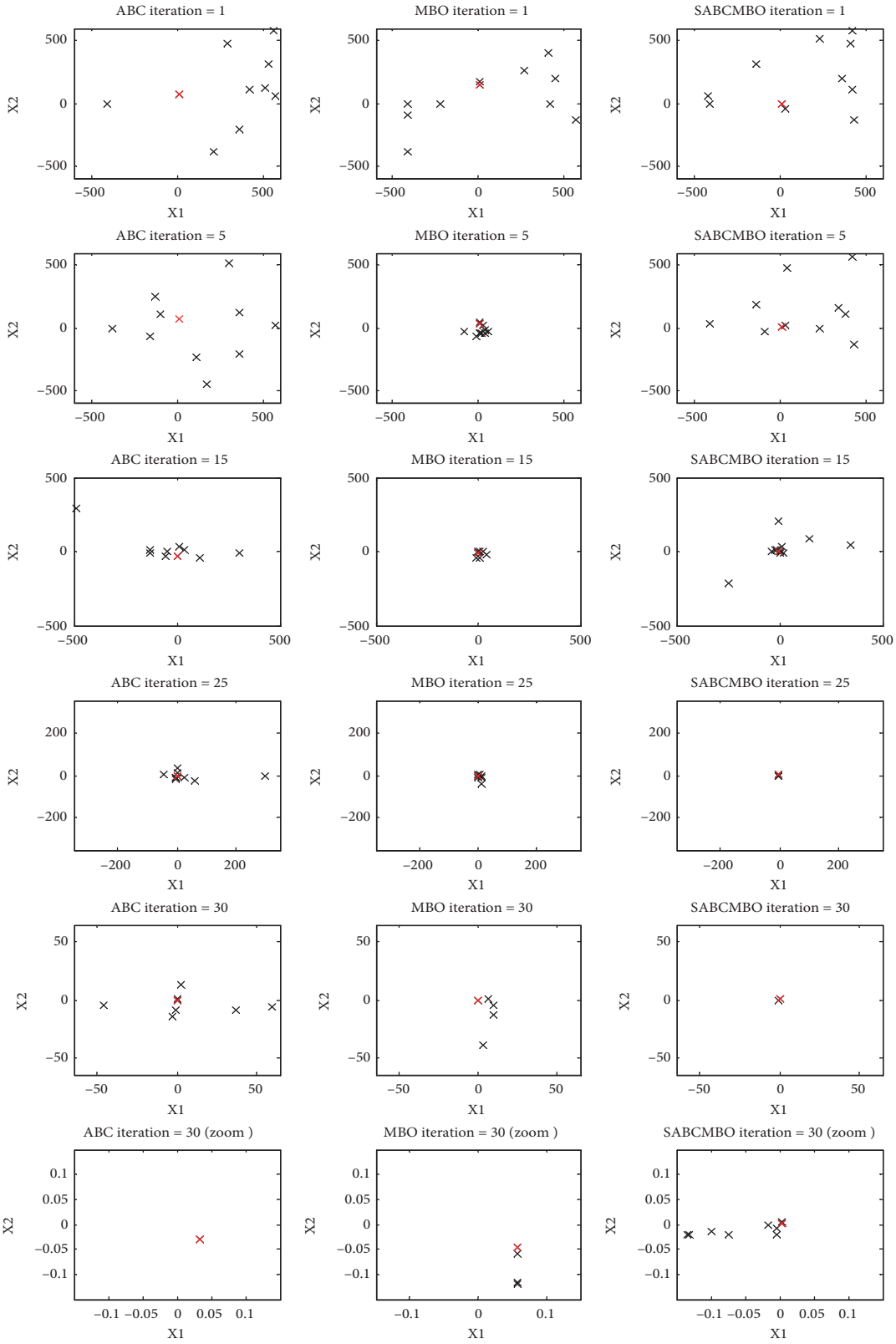

Figure 7. An example to show the convergence in artificial bee colony, migrating birds optimization, and sequential artificial bee colony and migrating birds optimization algorithms.

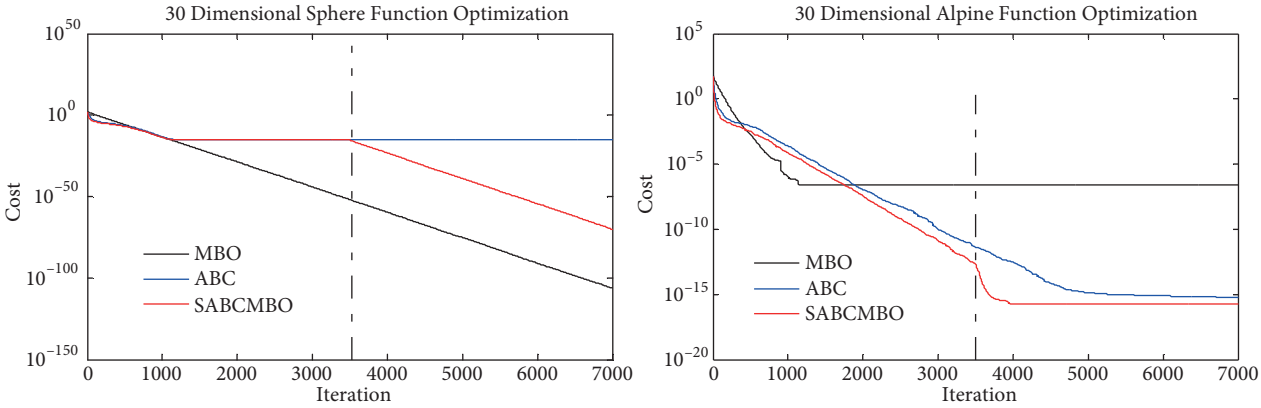

Figure 8. Average optimization performances of the algorithms on 30D functions.

for some high dimensional functions, and it cannot converge to global minimums as desired. Population size (n), mutation rate (CR), and iteration (K) values used are given in Table 4 for each dimension.

- The SABCMBO method has good characteristics both in exploration and in exploitation. Figure 6 summarizes the success rate distributions of the algorithms over problem dimensions. SABCMBO has competitive high success rates in all dimensions. It uses the powerful exploration capability of the ABC algorithm in the first half of its iterations. Hence, it directs the population towards the region in which the global minimum exists. Then it uses the good exploitation capability of the MBO algorithm to perform stable convergence in the second half of the iterations. Although success rate stabilities of the other algorithms change depending on parameter selection, SABCMBO keeps its success rate stable as seen in Figure 6. Total iterations of the ABC part (K_1) and total iterations of the MBO part (K_2) given in Table 4 are selected to get determined the approximate fitness calculation.

- Consequently, the SABCMBO method has high success rates and better convergence capacity. Figure 7 proves this fact graphically. This figure summarizes an experimental comparison between the ABC, MBO, and SABCMBO algorithms. In this small experiment, swarm sizes were 20 for the ABC algorithm and the ABC part of the SABCMBO algorithm, and 11 for the MBO algorithm and the MBO part of the SABCMBO algorithm. Since the number of fitness calculations per iteration in the MBO algorithm is approximately 2 times that in the ABC algorithm (for the parameters $k = 3$ and $x = 1$), the swarm size of the MBO algorithm was defined as approximately half of that in the ABC algorithm to get equal number of fitness calculations for a fair comparison. This experiment was performed on a 2 dimensional Griewank function, and the algorithms were run for 30 iterations. The positions of the population members after iterations 1, 5, 15, 25, and 30 are shown in Figure 7. Although the Griewank function problem space is defined in the range of [–600, 600] for all dimensions, all members shift towards the global minimum at (0,0) by improving their fitness values. Red positions on graphs represent the best positions that have been achieved by the populations so far. It is clearly seen that the algorithms are successful in moving the population members towards the global minimum. Analyzing the SABCMBO method in detail, it is clearly seen that SABCMBO directs most of the members towards the region including the global minimum via the ABC optimization method. Then it performs good convergences for these members using the MBO method. The graphs at the bottom row zoom to the region in the range of [–0.15, 0.15] for all dimensions. The best position achieved by the population in the SABCMBO method is closer to global minimum than that in the ABC and MBO algorithms. Finally, 1 member by the ABC algorithm, 4 members by the MBO algorithm, and 9 members by the SABCMBO method could be collected inside the zoomed region. This simple experiment supports the main motivation of this study.

- In giving the convergence characteristics, Figure 8 shows the optimization performances of the algorithms on 30 dimensional unimodal sphere and multimodal Alpine functions. Cost vs. iteration graphs in Figure 8 were obtained by averaging the cost vs. iteration trends of executions in experiments. It can be seen in the graphs that the ABC algorithm performs a reasonable optimization, but it goes into saturation for the selected unimodal function and cannot manage a good convergence. On the other hand, the MBO algorithm falls into local minimums for the selected multimodal function. Combining the good characteristics of the ABC and MBO algorithms, the SABCMBO algorithm performs a preoptimization in the ABC optimization step; then it performs a quick convergence in the MBO step.

4. Conclusions and future work

Exploration and exploitation are 2 important issues for an optimization algorithm. If an algorithm is good at exploration, it can escape from local optimum traps and have more chance to move its solutions towards the global optimum. In addition to good exploration characteristic, if it is good at exploitation, it can make a good approach to the global optimum. Considering these optimization facts, in this study, it was possible to combine the good characteristics of the ABC and MBO algorithms by establishing a sequential execution cooperation. In the SABCMBO algorithm, first the population is directed to the region in which the global optimum exists using the ABC optimization; then a good convergence to the global optimum is achieved using MBO. In short, the good exploration property of the ABC algorithm and the good exploitation property of the MBO algorithm are used sequentially to find the global optimum point.

If initial populations stack around local optimums, algorithms having weak exploration capability generally tend to these local optimums instead of the global optimum. The sequential cooperation mechanism in our proposed method eliminates this negative effect of the initial population. Half of the predefined approximate fitness calculations are performed by the ABC part and the remaining half is performed by the MBO part. Therefore, total execution time is kept constant in our proposed method.

Similarly, a parallel cooperation mechanism may be tried as a future work to get better optimization performance, both in terms of success rate and convergence.

References

[1] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Informa Sciences 2013; 237: 82-117.

[2] Blum C, Roli A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv 2003; 35: 268-308.

[3] Yan TS, Tao YQ, Cui DW. Research on handwritten numeral recognition method based on improved genetic algorithm and neural network. In: Proceedings of international conference on wavelet analysis and pattern recognition; 02-04 November 2007; Beijing, China. pp. 1271-1276.

[4] Holland JH. Adaptation in Natural and Artificial Systems. Ann Arbor, Michigan, USA: University of Michigan Press, 1975.

[5] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983; 220: 671-680.

[6] Glover F. Future paths for integer programming and links to artificial intelligence. Comput Oper Res 1986; 13: 533-549.

[7] Dorigo M. Optimization, learning and natural algorithms. PhD, Politecnico di Milano, Milano, Italy, 1992.

[8] Eberhart RC, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micromachine and Human Science; 4–6 October 1995; Nagoya, Japan. pp. 39-43.

[9] Storn R, Price K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997; 11: 341-359.

[10] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001; 76: 60-68.

[11] Mucherino A, Seref O. A novel meta-heuristic search for global optimization. In: Proceedings of the Conference on Data Mining, System Analysis and Optimization in Biomedicine; 28–30 March 2007; Gainesville, FL, USA. pp. 162-173.

[12] Karaboğa D. An idea based on honey bee swarm for numerical optimization. Technical Report TR06. Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[13] Yang XS. Firefly algorithm. In: Yang XS, editor. Nature - Inspired Metaheuristic Algorithms. United Kingdom: Luniver Press, 2010. pp. 81-90.

[14] Shah-Hosseini H. The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. International Journal of Bio-inspired Computation 2009; 1: 71-79.

[15] Yang XS, Deb S. Cuckoo search via lévy flights. In: Proceeings of World Congress on Nature & Biologically Inspired Computing (NaBIC); 09–11 December 2009; Coimbatore, India. pp. 210-214.

[16] Yang XS, Deb S. Engineering optimisation by cuckoo search. Int J Math Modelling & Num Optimisation 2010; 1: 330-343.

[17] Yang XS. A new metaheuristic bat-inspired algorithm. In: Cruz C, González JR, Krasnogor N, Pelta DA, Terrazas G, editors. Nature Inspired Cooperative Strategies for Optimization - NICSO (Studies in Computational Intelligence; 284). Granada, Spain: Springer, 2010. pp. 65-74.

[18] Yang XS. Bat algorithm for multi-objective optimisation. Int J BioInspired Computation 2011; 3: 267-274.

[19] Duman E, Uysal M, Alkaya AF. Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem. Inform Sciences 2012; 217: 65-77.

[20] Pandian SMV, Thanushkodi K. Considering transmission loss for an economic dispatch problem without valve-point loading using an EP-EPSO algorithm. Turk J Elec Eng & Comp Sci 2012; 20: 1259-1267.

[21] Mutluer M, Bilgin O. Application of a hybrid evolutionary technique for efficiency determination of a submersible induction motor. Turk J Elec Eng & Comp Sci 2011; 19: 877-890.

[22] Jaddi NS, Abdullah S. Hybrid of genetic algorithm and great deluge algorithm for rough set attribute reduction. Turk J Elec Eng & Comp Sci 2013; 21: 1737-1750.

[23] Bu TM, Yu SN, Guan HW. Binary-coding-based ant colony optimization and its convergence. J Comput Sci Technol 2004; 19: 472-478.

[24] Hu XM, Zhang J, Li Y. Orthogonal methods based ant colony search for solving continuous optimization problems. J Comput Sci Technol 2008; 23: 2-18.

[25] Chu ZF, Xia YS, Wang LY. Cell mapping for nanohybrid circuit architecture using genetic algorithm. J Comput Sci Technol 2012; 27: 113-120.

[26] Li J, Liu X. Melt index prediction by RBF neural network optimized with an MPSO-SA hybrid algorithm. Neurocomputing 2011; 74: 735-740.

[27] Pop PC, Matei O, Sitar CP. An improved hybrid algorithm for solving the generalized vehicle routing problem. Neurocomputing 2013; 109: 76-83.

[28] Sun Y, Zhang L, Gu X. A hybrid co-evolutionary cultural algorithm based on particle swarm optimization for solving global optimization problems. Neurocomputing 2012; 98: 76-89.

[29] Yi H, Duan Q, Liao TW. Three improved hybrid metaheuristic algorithms for engineering design optimization. Appl Soft Comp 2013; 13: 2433-2444.

[30] Yaghini M, Karimi M, Rahbar M. A hybrid metaheuristic approach for the capacitated p-median problem. Appl Soft Comp 2013; 13: 3922-3930.

[31] Liefooghe A, Verel S, Hao JK. A hybrid metaheuristic for multi-objective unconstrained binary quadratic programming. Appl Soft Comp 2014; 16: 10-19.

[32] Cavuslu MA, Karakuzu C, Karakaya F. Neural identification of dynamic systems on FPGA with improved PSO learning. Appl Soft Comp 2012; 12: 2707-2718.

[33] Makas H, Yumusak N. New cooperative and modified variants of the migrating birds optimization algorithm. In: Proceedings of the International Conference on Electronics, Computer and Computation (ICECCO); 07–09 November 2013; Ankara, Turkey. pp. 176-179.

[34] Fornarelli G, Giaquinto A. An unsupervised multi-swarm clustering technique for image segmentation. Swarm and Evolutionary Computation 2013; 11: 31-45.

[35] Wang Y, Wang H, Lei X, Jiang Y, Song X. Flood simulation using parallel genetic algorithm integrated wavelet neural networks. Neurocomputing 2011; 74: 2734-2744.

[36] Baños R, Ortega J, Gil C, Fernández A, Toro F. A simulated annealing-based parallel multiobjective approach to vehicle routing problems with time windows. Expert Syst Appl 2013; 40: 1696-1707.

[37] Yusof R, Khalid M, Hui GT, Yusof SM, Othman MF. Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm. Appl Soft Comp 2011; 11: 5782-5792.

[38] Yu B, Yang Z, Sun X, Yao B, Zeng Q, Jeppesen E. Parallel genetic algorithm in bus route headway optimization. Appl Soft Comp 2011; 11: 5081-5091.

[39] Li G, Niu P, Xiao X. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. Appl Soft Comp 2012; 12: 320-332.

[40] Zhu G, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization. Appl Math and Comput 2010; 217: 3166-3173.

[41] Xiang W, An M. An efficient and robust artificial bee colony algorithm for numerical optimization. Computers & Operations Research 2013; 40: 1256-1265.

[42] Babayigit B, Ozdemir R. A modified artificial bee colony algorithm for numerical function optimization. In: Proceedings of IEEE Symposium on Computers and Communications (ISCC); 01–04 July 2012; Cappadocia, Turkey. pp. 245-249.

[43] Gao W, Liu S. A modified artificial bee colony algorithm. Computers & Operations Research 2012; 39: 687-697.

[44] Sharma TK, Pant M. Golden search based artificial bee colony algorithm and its application to solve engineering design problems. In: Proceedings of International Conference on Advanced Computing & Communication Technologies (ACCT); 7–8 Jan. 2012; Rohtak, India. pp. 156-160.

[45] Zhang Y, Zeng P, Wang Y, Zhu B, Kuang F. Linear weighted Gbest-guided artificial bee colony algorithm. In: Proceedings of International Symposium on Computational Intelligence and Design (ISCID); 28–29 October 2012; Hangzhou, China. pp. 155-159.

[46] Fister I, Fister I Jr., Zumer V. Memetic artificial bee colony algorithm for large-scale global optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC); 10–15 June 2012; Brisbane, Australia. pp. 1-8.

[47] Tsai PW, Pan JS, Liao BY, Chu SC. Enhanced artificial bee colony optimization. Int J Innov Comput I 2009; 5: 5081-5092.

[48] Kang F, Li J, Ma Z. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Inform Sciences 2011; 181: 3508-3531.

[49] Gao W, Liu S. A modified artificial bee colony algorithm. Computers & Operations Research 2012; 39: 687-697.

[50] Banharnsakun A, Achalakul T, Sirinaovakul B. The best-so-far selection in artificial bee colony algorithm. Appl Soft Comp 2011; 11: 2888-2901.

[51] Mustaffa Z, Yusof Y, Kamaruddin SS. Gasoline price forecasting: an application of LSSVM with improved ABC. Procedia - Social and Behavioral Sciences 2014; 129: 601-609.

[52] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Appl Math Comput 2009; 214: 108-132.

[53] Akay B. Performance analysis of artificial bee colony algorithm on numerical optimization problems. PhD, Erciyes University, Kayseri, Turkey, 2009.

[54] Lissaman PBS, Schollenberger CA. Formation flight of bird. Science 1970; 168: 1003-1005.

[55] De Jong K. An analysis of the behavior of a class of genetic adaptive systems. PhD, University of Michigan, Michigan, USA, 1975.

[56] Rastrigin LA. Systems of Extremal Control. Moscow, Russia: Nauka, 1974.

[57] Griewank AO. Generalized descent for global optimization. J Optimiz Theory App 1981; 34: 11-39.

[58] Michalewicz Z. Genetic Algorithms C Data Structures D Evolution Programs. New York, NY, USA: Springer, 1992.

[59] Rahnamyan S., Tizhoosh HR, Salama NMM. A novel population initialization method for accelerating evolutionary algorithms. Computers and Mathematics with Applications 2007; 53: 1605-1614.

[60] Bäck T, Schwefel, HP. An overview of evolutionary algorithms for parameter optimization. Evol Comput 1993; 1: 1-23.

[61] Schwefel HP. Numerical Optimization of Computer Models. Chichester, UK: Wiley, 1981.

[62] Ackley DH. A Connectionist Machine for Genetic Hill Climbing. Norwell, MA, USA: Kluwer Academic Publishers, 1987.