International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-018-0878-6

ORIGINAL ARTICLE

Boosting galactic swarm optimization with ABC

Ersin Kaya[1] · **Sait Ali Uymaz**[1] · **Baris Kocer**[1]

Received: 17 October 2017 / Accepted: 19 September 2018

Abstract
Galactic swarm optimization (GSO) is a new global metaheuristic optimization algorithm. It manages multiple sub-populations to explore search space efficiently. Then superswarm is recruited from the best-found solutions. Actually, GSO is a framework. In this framework, search method in both sub-population and superswarm can be selected differently. In the original work, particle swarm optimization is used as the search method in both phases. In this work, performance of the state of the art and well known methods are tested under GSO framework. Experiments show that performance of artificial bee colony algorithm under the GSO framework is the best among the other algorithms both under GSO framework and original algorithms.

Keywords Galactic swarm optimization · Artificial bee colony algorithm · Swarm intelligence · Metaheuristic optimization algorithm

1 Introduction

The aim of the optimization is determining the parameters which maximizes or minimizes an objective function. These parameter may have discrete or continuous values. Real world optimization problems are hard because they may have enormous search space because of lots of parameters and lots of local minima points that make it more difficult to find global optima. Also some problems can't be implemented by linear functions. Examining all possible solutions for that kind of problems requires too many computing and so almost impossible. Metaheuristic methods are developed to search best possible solution in reasonable time. These methods are developed by inspiring different disciplines like biology, physics and sociology.

Most of the today's successful metaheuristic methods are inspired from swarm behavior of animals, biological systems and natural phenomena. The most important bio-inspired metaheuristic search algorithm genetic algorithms is developed in 1975 and inspired from evolution theory [12]. It is a fundamental algorithm that sometimes used to solve other algorithms problems like inverse problem of the support vector machines which is about splitting a given dataset into two clusters [36] and training extreme learning machines [1]. Researchers attempted to in 1983 collective behavior of the ant swarms give researchers an inspiration to develop ant colony optimization algorithm (ACO) [4]. Another similar algorithm particle swarm optimization [16] is inspired from collective behavior of fish schools and bird flocks in 1995 by Eberhart and Keneddy. Bees are another motivation source for metaheuristic search algorithm. For example artificial bee colony algorithm (ABC) [14] simulates the intelligence food gathering behavior of the bumble bees. Although the ABC method has good exploration ability, exploitation ability is weak. For this reason, in recent years there have been many studies conducted on behalf of the ABC method exploration ability. In a study by Li et al., a modification was proposed for the problem of slow convergence of ABC. Modification is based on the fact that in the population, quality individuals produce better individuals faster by doing gene exchange with some of them [19]. In a study by Cui et al., adaptive population modification for the ABC method was presented. In the case of inadequate food resources, the chance of reaching better food sources of better individuals at hand is increased by reducing the population by eliminating worse individuals. Thanks to this modification, the ABC method has a better exploitation ability [7]. In another study by Cui et al. A mutation operator was proposed to

✉ Ersin Kaya
ersinkaya@selcuk.edu.tr

[1] Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Konya Technical University, Konya, Turkey

Published online: 24 September 2018

increase the exploitation ability of the ABC method. This mutation operator was inspired by the mutation operator in the differential evolution (DE) algorithm. New individuals are created by using useful information obtained from randomly selected solutions in DE. In this study, the individuals are selected according to a ranking order, not random. The method has been compared with other ABC versions and the success has been achieved [5]. The DFSABC_elite method was developed by Cui et al. In 2016 to strengthen the ABC's slow convergence weakness. They have proposed a depth-first search framework to better balance between exploration and exploitation [6]. Microorganisms also have intelligence behavior that give inspiration. For example algae search for best living condition according to light source, food source etc. and this behavior inspired the artificial algae algorithm (AAA) [34]. The tree seed algorithm developed by Kiran is in the field of heuristic and population-based search. The new intelligent optimizer based on the relation between trees and their seeds for continuous optimization [17]. Another metaheuristic algorithm based on echolocation behavior of bats called bat algorithm is introduced by Yang [37].

Not only biology but also other natural phenomena like metal annealing inspired to simulated annealing algorithm (SA) by Kirkpatrick et al. In 1983 [18]. Differential evolution is another population based optimization algorithm by Price et al. [33]. It has many applications like determining feature weights in clustering problem [3]. Another example from nature can be given by gravitational search algorithm (GSA) which is inspired from universal gravitational force by Rashedi et al. [31]. Similarly galactic swarm optimization (GSO) is very new natural inspired algorithm by Muthiah-Nakarajan et al. [26] developed in 2016.

Galactic swarm optimization algorithm mimics movement of galaxies and stars in the galaxies. In fact it is not a classical heuristic algorithm. It is a metaheuristic upper strategy that lets to user select the search strategy inside the framework. GSO is a two phase metaheuristic search strategy. In the first phase sub population created to work independently. In the second phase best solution that comes from sub populations are merged to create a super population. The main search is executed via super population. In the original work of the GSO, PSO is used because of its simplicity and ease of coding.

In this work GSO is implemented by other well-known successful optimization algorithms to investigate the performance with other algorithms. For the purpose ABC, AAA, DE and GA is used under GSO framework. Performance evaluation is not only made with GSO with different algorithms but algorithms themselves also used in comparisons. The results of the GSO_ABC were also compared with the results of the GRABC, GRGABC, GRCABC and DFSABC_elite methods presented by Li et al. in 2017.

Organization of the paper is GSO algorithm is explained well in Sect. 2, base search algorithms are introduced shortly in Sect. 3, experimental results and discussions about the results are presented in Sect. 4. In the last section total conclusion is made and talked about future work.

2 Galactic swarm optimization

Galactic swarm optimization mimics the movement of stars, galaxies and supergalaxies to search possible solutions in a certain search space. Just like stars in galaxies, galaxies also interact each other. GSO groups the agent in two level. First level of group represents the stars and the second level represents the galaxies. Each level is independent search mechanism expect second level's initial population because it recruited from first level's best solutions. Different search algorithms can be selected in each level. Authors decided to use PSO algorithm in all levels. So in the first level each sub population searches best solution using PSO then send it to upper level to create superswarm. Superswarm is used as initial population of another PSO run to search best solution. Multilayered struck of the GSO algorithm is expressed in (1):

$$\begin{aligned} &s_j^i \in S_i : j = 1, 2, \ldots, N \\ &b_i \in S_i : b_i = best(S_i) \\ &G = \bigcup_{i=1}^{M} b_i. \end{aligned} \tag{1}$$

In the original GSO algorithm, initial M sub population which consists of N solutions are randomly generated. s_j^i represents the jth solution of the ith subpopulation. S_i represents the ith subpopulation. b_i ($best(S_i)$) represents the best solution of the subpopulation S_i. Set G represents superpopulation that consists of the best solutions comes from subpopulations.

Phase 1 in Fig. 1 is run L1 times. The best solutions acquired from every subpop in phase 1 is used as initial population of phase 2. Phase 2 is run L2 times then best result found in phase 2 accepted as final solution of the epoch. Whole algorithm is run epoch count times then best results found so far in epochs accepted as the final result of the algorithm. Pseudo code of the GSO framework that is implemented by PSO is given in Fig. 2.

3 Base algorithms

In the original GSO developed by Muthiah-Nakarajan and Noel PSO is used in both phases. Performance of the original algorithm tested against other versions of the

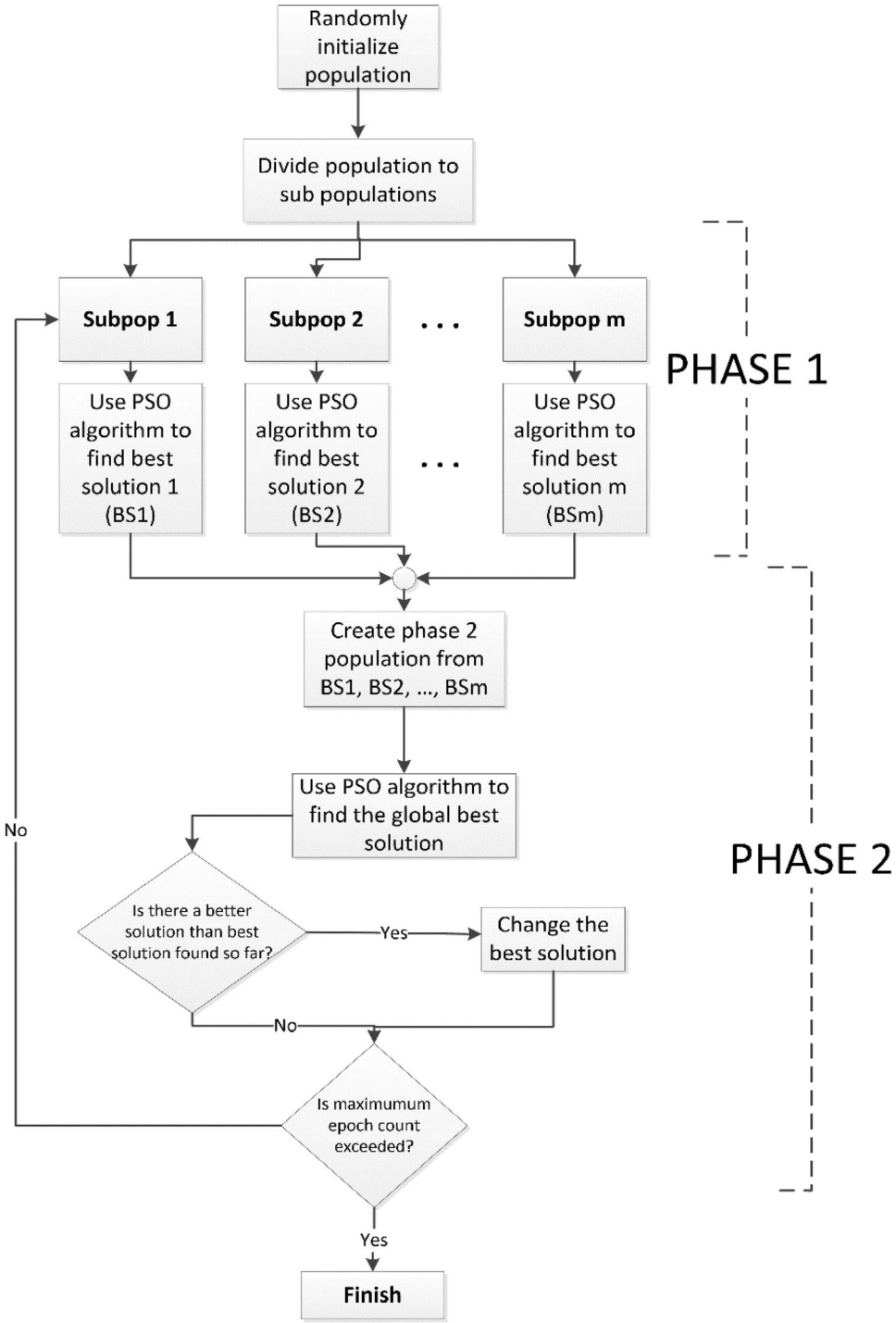


Fig. 1 Flowchart of the GSO algorithm

PSO and GSO with PSO surpassed. In this work evolution based algorithms such as GA and DE and food collecting behavior of the living organism inspired algorithms like ABC and AAA used in GSO and performance changes are measured. In short this work presents the performance of GSO algorithm with DE, AAA, GA and ABC.

Fig. 2 Implementation of GSO algorithm with PSO

```
Objective function f(x), x={x1,x2,...,xd}
Initialize phase 1 population (x1) in [xmin, xmax]^D
Initialize phase 1 variables (v1ij, p1ij, g1i) in [xmin, xmax]^D
Initialize phase 2 population (x2) in [xmin, xmax]^D
Initialize phase 2 variables (v2i, p2i, g2) in [xmin, xmax]^D
for EP = 1 to EPmax
    Start Phase 1
    for i = 1 to Number of Partitions
        for k=1 to Number of Iteration of Phase 1
            for j = 1 to Size of Partitions
                v1ij ← w1 v1ij + c1r1(p1ij – x1ij) + c2r2(g1i – x1ij)
                x1ij ← x1ij + v1ij
                if (f(x1ij) < f(p1ij)) then
                    p1ij ← x1ij
                    if (f(p1ij) < f(g1i)) then
                        g1i ← p1ij
                        if (f(g1i) < f(g2)) then
                            g2 ← g1i
                        end
                    end
                end
            end
        end
    end
    Initialize phase 2 population
    for j = 1 to Number of Partitions
    x2j ← g1j
    end

    Start Phase 2
    for k=1 to Number of Iteration of Phase 2
        for i = 1 to Number of Partitions
            v2i ← w2 v2i + c3r3(p2i – x2i) + c4r4(g2 – x2i)
            x2i ← x2i + v2i
            if (f(x2i) < f(p2i)) then
                p2i ← x2i
                if (f(p2i) < f(g2)) then
                    g2 ← p2i
                end
            end
        end
    end
end
Return g2
```

3.1 Particle swarm optimization

Particle swarm optimization is a result of inspiration from social behavior of bird flocks and fish schools by Kenedy and Eberhard [16]. It is a population based optimization algorithm. Randomly created individuals search for the best solution by the guidance of best individual in population based algorithms. Some population based algorithms like PSO also use the personal best solution found so far in search process. PSO is an easy to use algorithm because it does not has lot of parameters to tune and coding the PSO is not a complex task. Velocity of each individual is calculated by Eq. 2 and the new positions are calculated by Eq. 3:

$$v_i(t+1) = v_i(t) + c_1 r_1 (p_i(t) - x_i(t) + c_2 r_2 \big(g(t) - x_i(t)\big), \tag{2}$$

$$x_i(t+1) = x_i(t) + v_i(t), \tag{3}$$

where $v_i(t)$ is the velocity of particle i at iteration t. $x_i(t)$ is the position of particle i at iteration t. r_1 and r_2 are uniformly distributed random number in [0,1]. c_1 and c_2 are the acceleration constants which controls the search direction towards the personal best or global best solutions respectively. $p_i(t)$ is the personal best solution of ith particle at iteration t. $g(t)$ is the best solution found so far at iteration t.

Work flow of the PSO can be summarized in five steps:

1. Create random initial population.
2. Calculate every particles velocity.
3. Update every particles position using velocity vector.
4. Update personal best and global best values.
5. Return to step two until termination criteria met.

New PSO models are developed by changing some fundamental dynamics of the algorithm [20, 23, 25, 27, 29] but in this work the original PSO algorithm is used [16].

3.2 Differential evolution

Differential evolution is another population based algorithm developed by Storn and Price [33]. Like other population based algorithm, searching begins by creating random initial population. Then selection, crossover and mutation operators are applied to create new population. Individuals expressed as $X_i = \left(x_{1i}, x_{1i}, \ldots, x_{1D}\right)\ i = 1, 2, \ldots, NP$. NP represents individual count in the population, D represents the dimensionality. First of all mutation vector $V_i = \left(v_{1i}, v_{1i}, \ldots, v_{1D}\right)$ is created using Eq. 4:

$$V_i = X_{r_1} + F * \left(X_{r_2} - X_{r_3}\right), \quad r_1 \neq r_2 \neq r_3 \neq i, \qquad i = 1, 2, \ldots, NP, \tag{4}$$

where $F \in [0, 1]$ is scaling factor, $r_1, r_2, r_3 \in @1, 2, \ldots, NP@$ are randomly chosen numbers. New individual $U_i = \left(u_{1i}, u_{1i}, \ldots, u_{1D}\right)$ is created using crossover operator, X_i and V_i vectors. Equation 5 is used to create new individual.

$$U_{ij} = \begin{cases} v_{ij} & if\ (rand(0,1)\ \leq CR\ or\ j = rand(i)) \\ x_{ij} & if\ (rand(0,1)\ > CR\ and\ \ j \neq rand(i)) \end{cases} \tag{5}$$

where $j \in @1, 2, \ldots, D@, CR \in [0,\ 1]$ is crossover constant, $rand(i) \in [1, 2, \ldots, D]$ randomly chosen index.

It is decided to pass the new individual to new population by selection operator. Selection operator compares the fitness value of the new individual with old one and if there is an improvement then it passes the individual new generation.

Generating new populations via mutation, crossover and selection continues until a termination criteria met [28]. There a lot of work done it literature to increase the performance of DE [21, 22, 30, 32, 38]. In this work original DE is used [33].

3.3 Artificial algae algorithm

Artificial algae algorithm developed by Uymaz et al. by idealizing characteristic behaviors of algae in 2015 [34]. The fundamental species used in the algorithm is algae and population consist of artificial algae colonies as expressed in Eq. 6. Each artificial algae colony consist of algal cells that lives together Eq. 7:

$$Algae\ colony = \begin{bmatrix} x_1^1 & \cdots & x_1^D \\ \vdots & \ddots & \vdots \\ x_N^1 & \cdots & x_N^D \end{bmatrix}, \tag{6}$$

Fig. 3 Pseudo code of AAA

```
Start
    Helical Movement
        1. Determine light source
        2. Determine three algal cells
        3. Move the algal colony
    Evolutional Process
        1. Determine biggest and smallest algal colony
        2. Copy one of the cell from the biggest colony to the smallest colony
    Adaptation Process
        1. Determine the hungriest algal colony
        2. Adapt the algal colony
End
```

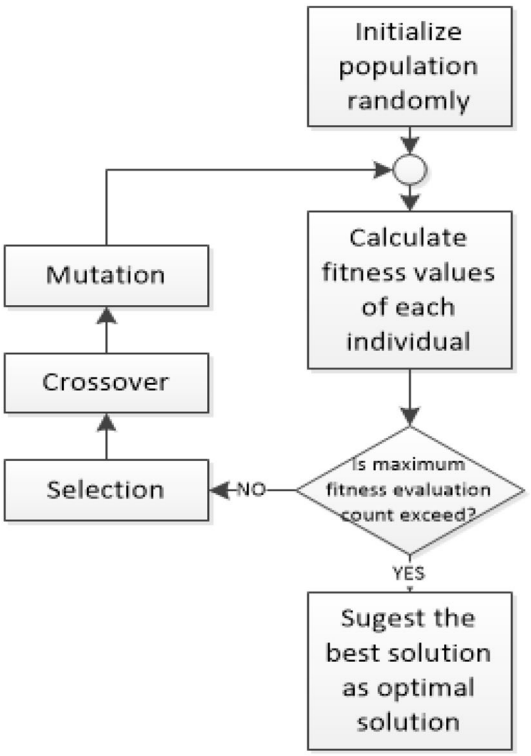


Fig. 4 Flowchart of the genetic algorithms

$$x_i = \left[x_i^1 , x_i^2 \cdots x_i^D \right] \quad i = 1, 2, \ldots, n, \tag{7}$$

where x_i^j, represents jth algal cell in ith algal colony. Every x_i represents a solution in the solution space. It is assumed that every algal cell moves together to the same direction.

Artificial algae algorithm constructed on three main concepts. Those are evolutional process, adaptation and helical movement. In evolutional process, if artificial algal cell gets enough light it groves and duplicates itself like mitotic division in biology. Otherwise if the algal cell cannot get enough light it lives for a while and eventually dies. Adaptation is a process that occurs when an algal colony stay alive but cannot grove enough then tries to change to be similar with the biggest algal colony in the environment. Not all algal colonies behave similarly. For example bigger algal colonies moves slower because of their friction surface in liquid environment. This situation increases local search abilities (exploration) in artificial algal colony. In the opposite way, smaller algal colonies have bigger step size so they are better in global search (exploitation) [35]. Primitive steps of AAA is depicted in Fig. 3.

3.4 Genetic algorithms

Genetic algorithms is developed by Holland [13]. It is based on evolution process which means adaptation of the fastest adapted species in nature. GA was born when the same process was applied in an optimization algorithm. Although this kind of metaheuristic can not grand to find the best solution, it can find near best solution in enormous search spaces where exhaustive search is not available.

Genetic algorithm creates a set of solutions (population) to mimics the evolution process. It measures the every solution's fitness value with a fitness evaluation function then it applies crossover and mutation operator to good solutions to create new population. It especially selects the strong individuals to crossover just happened in nature [2].

Mutation is changing random individuals' random gene. Genetic algorithm uses mutation not to trap local optimum solutions. The overall flowchart of GA is given in Fig. 4.

Fig. 5 Phases of the ABC

Step 1. Initialization: Employed bees initialized randomly by Eq.8.
Step 2. Every employed bee searches a better solution by selecting a random neighbor in Eq.9.
Step 3. Onlooker bee searches better food sources intensely around good food sources with high fitness values and better food sources are updated.
Step 4. If an employed bee exceeds the limit value it becomes the scout bee. Becoming a scout bee is the same as randomly initialization. If maximum fitness evaluation count is not reached algorithm returns to step 2.

Table 1 Parameters of base algorithms

Algorithm	Parameter
ABC	POPSIZE = 50, limit = FoodNumber*D/5, MaxIter = MFes/POPSIZE
AAA	POPSIZE = 50, ShearForce (Δ) = 2, LossOfEnergy (e) = 0.3, AdaptationParameter (A_p) = 0.5
DE	POPSIZE = 50, CR = 0.9, F = 1, strategy is DE/Best/1
GA	POPSIZE = 50, crossover rate = 0.9, mutation rate = 0.1
PSO	POSIZE = 50, c1 = c2 = 2.05

Table 2 Benchmark functions

No. of funct.	Name	Search Range	C	Function
F1	Sphere	[−100,100]	US	$f_1(\vec{X}) = \sum_{i=1}^{n} x_i^2$
F2	Elliptic	[−100,100]	UN	$f_2(\vec{X}) = \sum_{i=1}^{n} (10^6)^{(i-1)/(n-1)} x_i^2$
F3	SumSquares	[−10,10]	US	$f_3(\vec{X}) = \sum_{i=1}^{n} i x_i^2$
F4	SumPower	[−10,10]	MS	$f_4(\vec{X}) = \sum_{i=1}^{n} \lvert x_i \rvert^{(i+1)}$
F5	Schwefel2.22	[−10,10]	UN	$f_5(\vec{X}) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$
F6	Schwefel2.21	[−100,100]	UN	$f_6(\vec{X}) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$
F7	Step	[−100,100]	US	$f_7(\vec{X}) = \sum_{i=1}^{n} \left(\lfloor x_i + 0.5 \rfloor \right)^2$
F8	Quartic	[−1.28,1.28]	US	$f_8(\vec{X}) = \sum_{i=1}^{n} i x_i^4$
F9	QuarticWN	[−1.28,1.28]	US	$f_9(\vec{X}) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$
F10	Rosenbrock	[−10,10]	UN	$f_{10}(\vec{X}) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + \left(x_i - 1 \right)^2 \right]$
F11	Rastrigin	[−5.12,5.12]	MS	$f_{11}(\vec{X}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10 \right]$
F12	Non-Continuous Rastrigin	[−5.12,5.12]	MS	$f_{12}(\vec{X}) = \sum_{i=1}^{n} \left[y_i^2 - 10\cos(2\pi y_i) + 10 \right]$ $y_i = \begin{cases} x_i & \lvert x_i \rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \lvert x_i \rvert \ge \frac{1}{2} \end{cases}$
F13	Griewank	[−600,600]	MN	$f_{13}(\vec{X}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$
F14	Schwefel2.26	[−500,500]	UN	$f_{14}(\vec{X}) = 418.98*\mathrm{n} - \sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert} \right)$
F15	Ackley	[−32,32]	MN	$f_{15}(\vec{X}) = -20\exp\left\{ -0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right\} - \exp\left\{ \frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i) \right\} + 20 + e$
				$f_{16}(\vec{X}) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$
F16	Penalized1	[−50,50]	MN	$y_i = 1 + \frac{1}{4}(x_i + 1) \quad u_{x_i,a,k,m} = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(x_i - a)^m & x_i < -a \end{cases}$
F17	Penalized2	[−50,50]	MN	$f_{17}(\vec{X}) = \frac{1}{10}\{\sin^2(\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] +$ $(x_n - 1)^2 [1 + \sin^2(2\pi x_{i+1})]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$
F18	Alpine	[−10,10]	MS	$f_{18}(\vec{X}) = \sum_{i=1}^{n} \lvert x_i \cdot \sin(x_i) + 0.1 \cdot x_i \rvert$
F19	Levy	[−10,10]	MN	$f_{19}(\vec{X}) = \sum_{i=1}^{n-1} (x_i - 1)^2 [1 + \sin^2(3\pi x_{i+1})] + \sin^2(3\pi x_1) +$ $\lvert x_n - 1 \rvert [1 + \sin^2(3\pi x_n)]$

Table 2 (continued)

No. of funct.	Name	Search Range	C	Function
F20	Weierstrass	[−0.5,0.5]	MN	$f_{20}(\vec{X}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k\max} [a^k \cos(2\pi b^k (x_i + 0.5))] \right) - D \sum_{k=0}^{k\max} [a^k \cos(2\pi b^k 0.5)]$ $a = 0.5,\ b = 3,\ k_{\max} = 20$
F21	Schaffer	[−100,100]	MN	$f_{21}(\vec{X}) = 0.5 + \frac{\sin^2\left(\sqrt{\sum_{i=1}^{n} x_i^2}\right) - 0.5}{\left(1 + 0.001 * \left[\sum_{i=1}^{n} x_i^2\right]\right)^2}$
F22	Shifted Sphere	[−100,100]	US	$f_{24}(\vec{X}) = \sum_{i=1}^{n} z_i^2 \quad z = x - o$
F23	Shifted Rastrigin	[−5.12,5.12]	MS	$f_{25}(\vec{X}) = \sum_{i=1}^{n} \left[z_i^2 - 10\cos(2\pi z_i) + 10 \right] \quad z = x - o$
F24	Shifted Griewank	[−600,600]	MN	$f_{26}(\vec{X}) = \frac{1}{4000} \sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 \quad z = x - o$
F25	Shifted Ackley	[−32,32]	MN	$f_{27}(\vec{X}) = -20\exp\left\{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} z_i^2}\right\} - \exp\left\{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i)\right\}$ $z = x - o$
F26	Shifted Alpine	[−10,10]	MN	$f_{28}(\vec{X}) = \sum_{i=1}^{n} \left\lvert z_i \cdot \sin(z_i) + 0.1 \cdot z_i \right\rvert \quad z = x - o$

3.5 Artificial bee colony algorithm

Artificial bee colony algorithm is developed by Karaboğa [14]. Every possible solution is represented by an employed bee in ABC. Employed bees are employed to give information about the food source. In nature they give information by waggle dance and in ABC they give the information by fitness value. Half of the population is employed bee and the other half is onlooker bee. As happens in nature onlooker bees prefer bigger and better food source. In ABC onlooker bees searche better solution around the employed bees. If a food source can't be improved after predefined number of searches (limit) then it becomes to scout bee [15]. Running of ABC is illustrated in Fig. 5 step by step:

$$x_{ij} = L_j + (U_j - L_j) \times r, \tag{8}$$

where $i = \{1, 2, 3, \ldots, FN\}, j = \{1, 2, 3, \ldots, D\}$ and where U_j jth. Dimension upper bound and L_j is the lower bound the jth dimension, r is a random number in [0, 1] range. FN employed bee count, D is the dimension count:

$$x_{ij}(t+1) = x_{ij}(t) + (x_{ij}(t) - x_{kj}) \times r, \tag{9}$$

where $i = \{1, 2, 3, \ldots, FN\}, j = \{1, 2, 3, \ldots, D\}, k \neq i$. In Eq. 9, x_{ij} represent ith employed bees jth parameter. The j value is selected random and then kth neighbor is selected randomly which obeys $k \neq i$. After determining a random r value in [0,1] $x_{ij}(t+1)$ is calculated.

Onlooker bees calculates the probability of the solution to update by Eq. 10. If the probability is higher than a random number in [0,1] then the solution is selected [11]:

$$P_i = \frac{F_i}{\sum_{j=1}^{FN} F_j}. \tag{10}$$

4 Experimental results

In this work AAA, DE, GA, PSO and ABC are selected as base methods for experiments. In addition, the GSO_ABC method was also compared with the ABC versions GRABC, GRGABC,GRCABC and DFSABC_elite, which were developed in 2017. Because of the difficulty of comparing and presenting test results of all these algorithms and their GSO variants we decided to eliminate some of the methods that has the worst performance. For the purpose every base method is compared to its GSO variant and test results showed that AAA, DE and GA are better than their GSO variants. So experimental results are gathered from GSO_PSO, GSO_ABC, AAA, DE and GA methods. All parameters of the algorithm is depicted in Table 1. It is not reasonable to compare algorithms over number of iteration. Because each algorithm has different iteration cost. For this reason, it is more convenient to use FEs to compare algorithms [24]. Because of all algorithms used in the

Table 3 Test results for D = 30

F. no	GSO_ABC		AAA		DE		GA		GSO_PSO	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	**3.31E−144**	**1.47E−143**	2.85E−30	2.20E−30	6.56E−07	2.20E−06	1.74E−10	2.09E−10	7.32E−06	3.12E−05
F2	**6.50E−125**	**2.11E−124**	2.71E−27	2.05E−27	0.34750633	1.69287975	0.00012581	0.00059655	32144.0249	154038.267
F3	**4.41E−150**	**2.16E−149**	4.80E−31	3.61E−31	5.18E−07	1.86E−06	4.24E−11	1.00E−10	0.00020192	0.00108737
F4	**1.28E−219**	**0**	2.56E−105	1.37E−104	9.72E−09	4.31E−08	2.15E−13	9.84E−13	1.05086386	5.65907214
F5	**4.30E−49**	**1.12E−48**	4.38E−19	2.42E−19	5.67E−07	4.56E−07	2.68E−07	1.64E−07	0.00447173	0.02408102
F6	**1.79E−114**	**9.71E−115**	0.37660027	0.074352	18.9726967	7.75657925	4.53524	0.92476879	0.00298854	0.01609376
F7	**0**	**0**	**0**	**0**	1.23333333	4.5802717	**0**	**0**	**0**	**0**
F8	**0**	**0**	3.01E−58	3.58E−58	2.42E−12	7.82E−12	1.60E−23	3.31E−23	1.31E−14	7.07E−14
F9	**9.88E−06**	**9.05E−06**	0.01744896	0.00499193	0.29714278	0.29256877	0.00290952	0.00110701	0.00010312	0.0001335
F10	23.799953	**4.4441348**	**9.35873029**	14.1411476	33.1762729	29.6104771	42.266763	31.9020994	24.060124	10.4226565
F11	**0**	**0**	0.0331653	0.17860061	49.5156633	15.4728933	2.52137443	1.27731082	0.00014558	0.00078399
F12	**0**	**0**	**0**	**0**	66.9666667	19.1058165	1.77998041	1.15629066	0.83335535	4.48763325
F13	**0**	**0**	**0**	**0**	0.46947274	2.3197325	0.0142663	0.01191975	0.0031628	0.01582361
F14	3069.15133	357.614355	**3.94793333**	**21.2602716**	1274.34743	463.464795	343.686294	210.430297	3121.62433	783.706026
F15	**0**	**0**	1.82E−14	5.09E−15	5.99775678	4.48753974	2.26E−06	1.14E−06	0.4605	2.474
F16	1.56E−11	1.74E−11	**6.03E−32**	**5.03E−32**	0.22207964	0.35833355	8.00E−10	4.28E−09	0.0744318	0.0625672
F17	1.24E−08	6.10E−08	**5.78E−31**	**5.04E−31**	0.33189331	0.75197367	0.00073249	0.00274074	1.39989396	0.55547701
F18	2.75E−07	4.84E−07	**1.77E−07**	**3.07E−07**	3.24E−07	6.17E−07	4.66E−07	6.55E−07	0.12440889	0.47001456
F19	3.71E−07	1.33E−06	**1.61E−31**	**8.67E−32**	1.50423434	2.66848383	4.52E−08	1.11E−07	9.09079417	6.64117394
F20	**0**	**0**	**0**	**0**	3.979601	2.01541658	9.51E−05	5.27E−05	0.00786657	0.04236276
F21	0.00874432	**0.00291477**	0.13251897	0.03265387	0.47648457	0.04032984	0.4062922	0.0428002	**0.00156467**	0.00684735
F22	0.00092509	0.0015565	**3.12E−30**	**2.53E−30**	1.41E−05	7.03E−05	3.43E−10	5.65E−10	5257.94657	12142.5881
F23	35.89349	5.43968339	**0**	**0**	46.9257433	10.6975057	3.34986892	1.86088466	149.047897	38.8694943
F24	0.02727738	0.02595301	**0**	**0**	0.308226	0.68120565	0.01083303	0.01047054	29.3521493	28.5488283
F25	1.89455807	0.81153102	**1.89E−14**	**4.97E−15**	3.62812247	2.9721862	3.84E−06	2.12E−06	12.201282	4.1837373
F26	0.0409	0.0303	**8.57E−08**	**1.16E−07**	8.37E−07	2.38E−06	2.50E−07	4.72E−07	0.2135	1.0591

experiment have different maximum iteration numbers, comparison made based on a predefined fitness evaluation count. 225,000, 450,000 and 750,000 maximum fitness evaluation numbers are selected for 30, 60 100 dimensions respectively.

All algorithms used in GSO have the parameters following; subpopsize (M) = 10, number of subpops (N) = 5, iteration count for phase 1 (L1) = D*10, iteration count for phase 2 (L2) = L1*5, number of epoch (EPmax) = 5. For ABC it is meaningless to set subpop value (N) to 5 is meaningless because ABC uses its half population to store the positions of the potential solution and other half search around them. So it was decided to set N = 10 but total fitness evaluation count was kept same.

26 benchmark functions are collected from [8–10]. Constraints, characteristics and global optimum points of the benchmark functions are shown in Table 2.

In Table 2, these functions are presented and in column C, the characteristic of the function is indicated as M for multimodal functions, U for unimodal functions, S for separable functions and N non-separable functions.

In Table 3 for D = 30, GSO_ABC outperforms in first 15 functions are except F10 and F14. Especially while GSO_ABC finds 3.31E−144 for F1 most the nearest competitor AAA can find 2.85E−30. For F7, all methods expect DE found the optimum solution zero. Although for F12, F13 and F20; GSO_AAA and GSO_ABC hit the zero, AAA outperform in functions between F16–F19 and F22–F26 and F14. Optimum solution zero can only be found by GSO_ABC for F8, F11 and F15. The best results in Tables 3, 4, 5, and 10 are presented in bold.

Only GSO_PSO could find the zero for F1, F6, F18 and F21 as seen in Table 4. GSO_PSO especially outperformed on unimodal functions. The best results for the first four functions and the second best results for the other two functions belong to GSO_ABC. While only GSO_ABC could found optimum values for F11, F12 and F15, GSO_ABC is one of the two methods that can find optimum values for F8, F13 and F20. That is how GSO_ABC dominates the other methods. It can be said that GSO_AAA is the best method for F13–F26 function in Table 4.

Table 4 Test results for D = 60

F. no	GSO_ABC		AAA		DE		GA		GSO_PSO	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	3.65E−138	1.75E−137	2.08E−29	1.67E−29	0.09366157	0.41822378	1.48E−08	4.13E−09	**0**	**0**
F2	**1.43E−109**	**7.67E−109**	1.46E−26	8.33E−27	704.90995	2502.95713	1.72E−05	8.79E−06	1548346.043	8329321.272
F3	**7.35E−142**	**3.70E−141**	5.50E−30	4.82E−30	0.08437817	0.42019865	3.78E−09	1.13E−09	100.0010462	538.5162865
F4	**1.04E−219**	**0**	1.43E−104	6.73E−104	9.97E+18	5.37E+19	1.76E−17	8.85E−17	3.33337E+11	1.79507E+12
F5	**9.42E−47**	**2.20E−46**	1.62E−18	6.78E−19	0.04528367	0.1771188	6.26E−05	1.10E−05	4.01E−05	0.000216101
F6	1.15E−115	6.51E−116	3.40143433	0.40622077	72.1759267	5.3419605	3.93733	0.89113829	**0**	**0**
F7	**0**	**0**	**0**	**0**	62	146.322247	**0**	**0**	0.166666667	0.897527468
F8	**0**	**0**	2.17E−56	2.78E−56	3.93E−05	0.00011275	1.50E−22	6.84E−23	**0**	**0**
F9	**5.79E−06**	**4.81E−06**	0.03907985	0.00978251	5.40682783	5.10128727	0.00493698	0.00143009	0.000112277	0.000139461
F10	**2.8191198**	**1.90305033**	38.72484	35.7856329	129.137367	72.9409869	99.7561233	41.2292321	56.6416347	10.51892222
F11	**0**	**0**	0.0663306	0.24818638	190.960667	31.910439	0.56399904	0.75754874	7.852255333	23.35869743
F12	**0**	**0**	0.03333333	0.17950549	228.1397	46.0953439	0.23353709	0.42288472	1.667425316	8.975134424
F13	**0**	**0**	**0**	**0**	0.48159402	1.72896969	0.00541829	0.0068284	0.009116667	0.034657936
F14	1058.15547	302.514959	**3.94793333**	**21.2602716**	3741.47333	689.099825	205.31013	178.123518	8610.031667	1920.920738
F15	**0**	**0**	3.29E−14	4.20E−15	11.509725	4.12339164	2.13E−05	4.28E−06	1.219528603	3.865817431
F16	2.84E−13	2.17E−13	**1.36E−31**	**9.96E−32**	29.6050044	142.642915	2.00E−11	9.70E−12	0.116282397	0.061864301
F17	7.19E−13	7.70E−13	**2.39E−30**	**1.47E−30**	1676.72716	8924.77225	5.73E−10	1.68E−10	4.51904	0.553204884
F18	1.52E−05	1.48E−05	4.69E−07	5.48E−07	0.1216937	0.60180411	1.16E−05	2.74E−06	**0**	**0**
F19	6.14E−11	1.08E−10	**5.94E−31**	**4.33E−31**	4.95111905	7.12353122	4.23E−10	1.25E−10	46.86571667	22.90829372
F20	**0**	**0**	**0**	**0**	18.011125	4.72526561	0.0115933	0.00142259	0.405300857	1.589903342
F21	0.0025912	0.00429636	0.41184797	0.02489054	0.49891473	0.00095841	0.4747278	0.00922183	**0**	**0**
F22	1.41E−12	2.02E−12	**1.81E−29**	**1.08E−29**	0.021147385	0.043375837	1.99E−08	5.15E−09	45645.8739	57586.25565
F23	8.382707333	5.097683407	**0.0331653**	**0.178600606**	154.7482	36.86916362	0.796114484	0.974824709	531.7764	138.8786533
F24	0.000372579	0.002006368	**0**	**0**	1.13921626	3.31007835	0.006064041	0.010004649	302.6103433	484.9395333
F25	1.54E−05	8.21E−05	**3.30E−14**	**4.12E−15**	10.65931533	5.237443473	2.57E−05	4.14E−06	17.33265	3.652841549
F26	6.43E−05	8.41E−05	**6.39E−07**	**8.92E−07**	2.911194427	2.845954131	5.43E−05	1.14E−05	55.56321	11.92464607

Table 5 Test Results for D = 100

F. no	GSO_ABC		AAA		DE		GA		GSO_PSO	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	**4.70E−137**	**9.08E−137**	5.88E−29	3.24E−29	842.881703	1797.19593	8.15E−05	1.99E−05	333.334661	1795.05469
F2	**1.16E−110**	**6.11E−110**	4.04E−26	1.92E−26	2024878.97	4816661.55	0.0825649	0.01592521	1900543.66	8911887.29
F3	**7.36E−140**	**1.87E−139**	3.53E−29	2.96E−29	215.928198	399.468403	2.98E−05	7.30E−06	53.1251655	219.971431
F4	**6.97E−225**	**0**	1.26E−97	4.85E−97	1.91E+41	1.03E+42	1.93E−49	5.60E−49	3.33E+30	1.80E+31
F5	**3.29E−45**	**7.37E−45**	4.94E−18	1.87E−18	3.26288082	5.13793702	0.00625498	0.00081714	0.33353821	1.79501723
F6	**7.05E−119**	**3.14E−119**	9.23564467	1.14335471	87.93138	3.21555842	1.48077133	0.27841139	0.00053468	0.00200148
F7	**0**	**0**	**0**	**0**	708.1	847.160644	**0**	**0**	**0**	**0**
F8	**0**	**0**	7.53E−56	5.39E−56	1.98594627	5.15778252	1.10E−15	3.77E−16	1.11E−11	5.94E−11
F9	**3.00E−06**	**2.94E−06**	0.07821918	0.01489377	18.7095807	14.2702777	0.00955288	0.00206717	5.28E−05	4.89E−05
F10	**0.9901797**	1.08728887	113.541796	42.3743489	7846.5755	14578.0761	188.922547	52.1010966	98.52812	**0.18336732**
F11	5.39E−07	2.80E−06	**0**	**0**	416.875433	53.7499604	0.1369776	0.33866517	16.7389496	43.6449168
F12	**8.17E−07**	**4.39E−06**	0.00486927	0.0262218	508.588733	52.1031386	0.00493208	0.02116891	5.00002236	13.5400558
F13	**0**	**0**	**0**	**0**	9.8279067	17.5186808	0.00489168	0.00865781	0.00218469	0.01176174
F14	659.6197	216.046766	**3.94793333**	**21.2602716**	7673.553	1033.75561	32.126374	60.4260897	16711.2533	3882.78261
F15	**3.53E−14**	1.90E−13	5.41E−14	**5.92E−15**	16.1714733	1.12013807	0.00125528	0.0001726	1.30891543	3.86779899
F16	1.34E−13	9.84E−14	**2.17E−31**	**8.91E−32**	2626803.64	7735803.17	5.39E−08	1.16E−08	0.18363499	0.06837255
F17	1.71E−13	1.70E−13	**7.18E−30**	**4.79E−30**	6069278.42	12884674.2	2.73E−06	7.36E−07	8.654163	0.80273473
F18	0.00015397	6.41E−05	**7.48E−07**	**1.01E−06**	1.30267688	1.44870694	0.00119186	0.00022019	0.3629267	1.38130796
F19	4.12E−11	8.53E−11	**1.04E−30**	**8.28E−31**	24.2899877	32.6300949	1.82E−06	3.67E−07	90.2826333	29.6840155
F20	**0**	**0**	**0**	**0**	50.3968067	6.47276967	0.2671566	0.02906221	1.16403966	3.78174129
F21	**0.00065566**	0.00242182	0.49639647	0.0012169	0.4998637	**0.00011881**	0.489315	0.00423004	0.0168554	0.08898196
F22	2.55E−13	2.76E−13	**6.83E−29**	**3.00E−29**	155.0666968	286.9261177	8.89E−05	2.25E−05	79353.94167	104311.3947
F23	1.212478751	0.792010839	**0.265322467**	**0.509495232**	368.3857333	40.4523639	0.300853969	0.522903199	1139.7414	313.9259849
F24	7.53E−09	2.77E−08	**0**	**0**	2.958601863	7.268622644	0.004976788	0.006929552	829.3964367	1012.969077
F25	5.58E−10	1.05E−09	**5.25E−14**	**6.33E−15**	17.56024	2.075263251	0.001312284	0.000133934	20.02447333	0.60735925
F26	0.000403004	0.000748308	**4.89E−07**	**4.89E−07**	9.269966446	5.687930387	0.012664808	0.03431052	130.3738367	31.62676268

Table 6 The results of two-sided Wilcoxon signed-rank test for 30 dimensions (α = 0.05) (p value: estimated probability of rejecting the null hypothesis (H0); T: the smallest of the sum of the signed ranks; W: Winner)

F. no.	GSO-ABC vs AAA			GSO-ABC vs DE			GSO-ABC vs GA			GSO-ABC vs GSO_PSO		
	p value	*T*	*W*	*p* value	*T*	*W*	*p* value	*T*	*W*	*p* value	*T*	*W*
F1	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	465	−
F2	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.11E−05	435	+
F3	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	465	−
F4	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	2.77E−03	378	+
F5	1.73E−06	465	+	1.73E−06	0	+	1.73E−06	0	+	3.59E−04	406	+
F6	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	465	−
F7	1.00E+00	0	=	6.25E−02	0	=	1.00E+00	0	=	1.00E+00	0	=
F8	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	465	−
F9	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	2.37E−05	27	+
F10	1.36E−04	418	−	3.60E−01	188	=	2.11E−03	83	+	4.95E−02	137	+
F11	1.86E−05	391	−	1.73E−06	0	+	3.93E−01	191	=	1.04E−03	347	−
F12	1.00E+00	1	=	1.72E−06	0	+	1.47E−06	0	+	1.00E+00	1	=
F13	1.00E+00	0	=	1.73E−06	0	+	1.72E−06	0	+	1.00E+00	0	=
F14	1.73E−06	465	−	1.92E−06	464	−	1.73E−06	465	−	8.94E−01	239	=
F15	3.40E−04	59	+	1.73E−06	0	+	1.73E−06	0	+	1.00E+00	28	=
F16	1.73E−06	465	−	1.73E−06	0	+	3.11E−05	435	+	1.73E−06	0	+
F17	1.73E−06	465	−	1.73E−06	0	+	2.05E−04	413	−	1.73E−06	0	+
F18	2.56E−02	341	−	2.54E−01	288	=	4.91E−01	266	=	2.76E−03	378	+
F19	1.73E−06	465	−	5.79E−05	37	+	4.20E−04	404	−	1.73E−06	0	+
F20	1.00E+00	0	=	1.73E−06	0	+	1.73E−06	0	+	6.25E−02	0	=
F21	1.14E−06	0	−	1.71E−06	0	+	1.66E−06	0	+	1.90E−07	406	−
F22	1.73E−06	465	−	6.32E−05	427	−	2.13E−06	463	−	1.73E−06	0	+
F23	1.73E−06	465	−	7.51E−05	40	+	1.73E−06	465	−	1.73E−06	0	+
F24	1.73E−06	465	−	2.18E−02	121	+	3.61E−03	374	−	1.73E−06	0	+
F25	1.73E−06	465	−	2.43E−02	123	+	1.73E−06	465	−	1.92E−06	1	+
F26	1.73E−06	465	−	6.04E−03	99	+	1.73E−06	465	−	1.73E−06	0	+
+/−/=	9/13/4			21/2/3			15/8/3			14/6/6		

Performance difference of GSO_ABC can be seen clearly in the first 15 function (except F11 and F14) in Table 5. While GSO_AAA has the second best performance, GA has the third best performance for these functions. Both GSO_ABC and GSO_AAA find the optimum value for F7, F13 and F20. Best solution found by GSO_AAA for functions F16–F19 and F22–F26 and F14.

In this study, to analyze the problem-solving success of methods the Wilcoxon signed-rank test was used. In Tables 6, 7 and 8 statistical comparisons by Wilcoxon signed-rank test are demonstrated between GSO_ABC and other methods. The last rows of Tables 6, 7 and 8 show the total counts in the (+/−/=) format for the three statistical significance cases in the pairwise comparison.

The successful results of the GSO_ABC method shown in Tables 3, 4 and 5 are also significant according to the statistical analysis results in Tables 6, 7 and 8. The GSO_ABC method yielded the most significant best result in studies of 30, 60 and 100 dimensions.

Table 9 summarizes the Tables 3, 4 and 5. It can be seen in the table GSO_ABC has the best mean value and AAA has the second best mean value with a little difference and the other algorithm can't make us feel the presence. Surprisingly GSO PSO is the leader in the best result found in 30 runs but it loses in the worst results found. Anyway standard deviation results show that it is not a stable algorithm. GSO_ABC and AAA have a significant performance supremacy in the worst solution and standard deviation. The table says that GSO_GA and GSO_DE are the worst algorithms among the others. In Figs. 6 and 7, convergence curves have been presented for 30, 60 and 100 dimensions of some of functions which have different characteristic.

Function F3 (sum squares) has no local minimum namely it is a unimodal function and its minimum is zero. When performance of the algorithms are analyzed for D = 30, GSO_ABC moves faster to better results after 1000 cycles on contrary of the other algorithms. However AAA seems getting better at the end of cycles GSO_ABC is the most successful one among the all algorithms. Ladder like shape

Table 7 The results of two-sided Wilcoxon signed-rank test for 60 dimensions ($\alpha = 0.05$) (p Value: estimated probability of rejecting the null hypothesis (H0); T: the smallest of the sum of the signed ranks; W: Winner)

F. no.	GSO-ABC vs AAA			GSO-ABC vs DE			GSO-ABC vs GA			GSO-ABC vs GSO_PSO		
	p value	*T*	*W*	*p* value	*T*	*W*	*p* value	*T*	*W*	*p* value	*T*	*W*
F1	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.11E−05	435	+
F2	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.71E−01	276	=
F3	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.59E−04	406	+
F4	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.71E−01	276	=
F5	5.71E−02	325	=	1.73E−06	0	+	1.73E−06	0	+	3.59E−04	406	+
F6	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	2.76E−03	378	+
F7	1.00E+00	0	=	2.54E−06	0	+	1.00E+00	0	=	1.00E+00	0	=
F8	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.59E−04	406	+
F9	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	2.13E−06	2	+
F10	1.24E−05	20	+	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F11	2.60E−06	461	+	1.73E−06	0	+	2.84E−05	436	−	2.77E−03	378	+
F12	1.73E−06	465	+	1.73E−06	0	+	6.58E−01	254	=	1.25E−02	354	+
F13	1.00E+00	0	=	1.73E−06	0	+	1.73E−06	0	+	2.50E−01	0	=
F14	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	465	−	1.73E−06	0	+
F15	3.91E−01	274	+	1.73E−06	0	+	1.73E−06	0	+	8.11E−01	91	=
F16	1.73E−06	465	−	1.73E−06	0	+	2.35E−06	3	+	1.73E−06	0	+
F17	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F18	1.73E−06	465	−	4.73E−06	10	+	3.06E−04	408	−	1.48E−02	351	+
F19	1.73E−06	465	−	1.73E−06	0	+	1.11E−02	109	+	1.73E−06	0	+
F20	5.00E−01	3	=	1.73E−06	0	+	1.73E−06	0	+	3.13E−02	1	+
F21	1.48E−06	0	+	1.71E−06	0	+	1.67E−06	0	+	8.97E−04	365	+
F22	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F23	1.73E−06	465	−	1.73E−06	0	+	1.92E−06	464	−	1.73E−06	0	+
F24	1.73E−06	465	−	1.73E−06	0	+	1.48E−04	48	+	1.73E−06	0	+
F25	1.73E−06	465	−	1.73E−06	0	+	3.11E−05	30	+	1.73E−06	0	+
F26	3.11E−05	435	−	2.35E−06	3	+	5.72E−01	205	=	1.73E−06	0	+
+/−/=	12/10/4			26/0/0			19/4/3			21/0/5		

of the convergence graphic of GSO_ABC and GSO_PSO in graphics for F3 for D = 30, 60 and 100 is its two phase algorithm. In the first phase it explores and in the second step it exploits. So in exploration step it can not find better results and graphic seems flat but in the exploitation step it quickly finds better results.

When convergence characteristic of F3 investigated for D = 60, it is almost same with D = 30. Similarly after first 2000 cycles it convergences better than other methods. Also AAA tries to convergence better results starting from 2000 cycles GSO_ABC is better in both convergence speed and final results. Results are the same for F3, D = 100. The only difference slower convergence rate because of the increasing dimensionality. For AAA it can be totally said that its convergence speed increases in the middle of the maximum iteration and at the end it finds better results than DE, GA and GSO_PSO.

F4 is another unimodal function. Test results show that GSO_ABC draws the best convergence graphic from beginning to end when D = 30. Especially in the first 1500 cycles its performance is wonderful after that point it draws ladder like performance graphic due two phases working nature. AAA also convergence to optimum in regular way but it is not so as stable as GSO_ABC. However GSO_PSO and GA has a fast convergence for the first 100 cycles, the rest of the graphics is flat.

F9 is a unimodal separable function. As observed in the graphics algorithms can be divided to two groups according to their performance. So DE, AAA and GA are a group and GSO_PSO and GSO_ABC are the other group. The second group shows superior performance from beginning to 2000 cycles. While in the first 10 cycles, second group algorithm differs a lot, after 2000 cycles difference remain the same. As the result GSO_ABC is the best algorithm in convergence speed and final result and GSO_PSO is the second best algorithm.

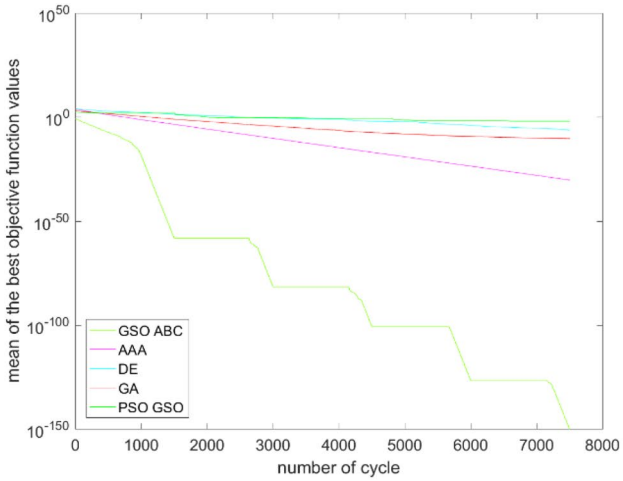
F9 graphic for D = 60 says that all methods perform the same except GSO_ABC in the beginning. GSO_ABC performs very well according to other algorithms from beginning to end. GSO_PSO which perform average, boost up

Table 8 The results of two-sided Wilcoxon signed-rank test for 100 dimensions ($\alpha = 0.05$) (p value: estimated probability of rejecting the null hypothesis (H0); T: the smallest of the sum of the signed ranks; W: Winner)

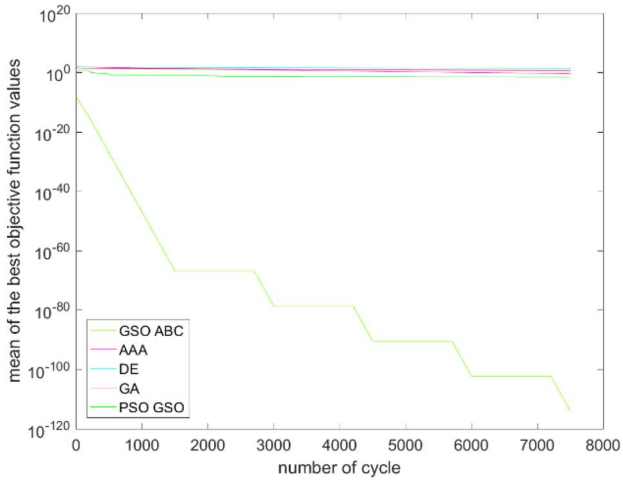
F. no.	GSO-ABC vs AAA			GSO-ABC vs DE			GSO-ABC vs GA			GSO-ABC vs GSO_PSO		
	p value	*T*	*W*	*p* value	*T*	*W*	*p* value	*T*	*W*	*p* value	*T*	*W*
F1	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	2.77E−03	378	+
F2	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	6.44E−01	210	=
F3	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	1.48E−02	351	+
F4	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	5.71E−02	325	=
F5	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	5.71E−02	325	=
F6	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.11E−05	435	+
F7	1.00E+00	0	=	1.73E−06	0	+	1.00E+00	0	=	1.00E+00	0	=
F8	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.59E−04	406	+
F9	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	3.18E−06	6	+
F10	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F11	9.17E−03	338	−	1.73E−06	0	+	1.96E−02	346	−	1.48E−02	310	+
F12	1.73E−06	465	+	1.73E−06	0	+	1.73E−06	465	−	3.60E−01	277	=
F13	1.00E+00	0	=	1.73E−06	0	+	1.73E−06	0	+	1.25E−01	0	=
F14	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	465	−	1.73E−06	0	+
F15	6.87E−01	213	+	1.73E−06	0	+	1.73E−06	0	+	1.00E+00	45	=
F16	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F17	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F18	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	5.71E−02	325	=
F19	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F20	1.00E+00	0	=	1.73E−06	0	+	1.73E−06	0	+	3.13E−02	0	+
F21	1.71E−06	0	+	1.73E−06	0	+	1.69E−06	0	+	1.34E−04	367	−
F22	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F23	1.73E−06	465	−	1.73E−06	0	+	8.92E−05	423	−	1.73E−06	0	+
F24	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F25	1.73E−06	465	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
F26	3.11E−05	435	−	1.73E−06	0	+	1.73E−06	0	+	1.73E−06	0	+
+/−/=	12/11/3			26/0/0			21/4/1			17/1/8		

Table 9 Summary of the test results

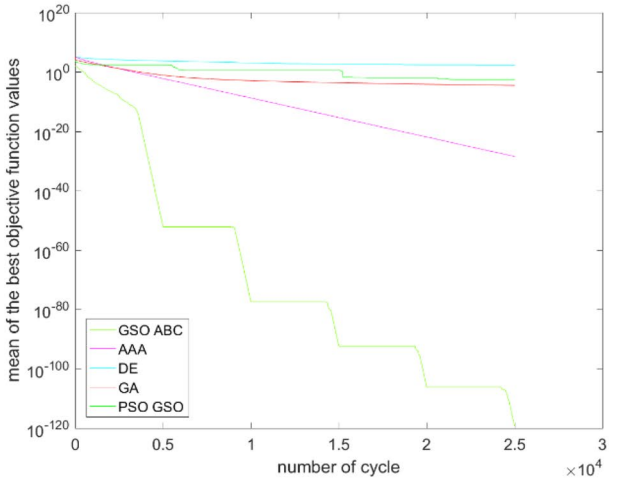
	GSO_ABC	AAA	DE	GA	GSO_PSO
Mean					
D = 30	14	15	0	1	3
D = 60	13	12	0	1	5
D = 100	15	14	0	1	1
Best					
D = 30	9	14	1	1	16
D = 60	9	14	1	1	16
D = 100	10	14	0	1	15
Worst					
D = 30	16	14	0	1	2
D = 60	13	12	0	1	5
D = 100	15	14	0	1	1
SD					
D = 30	16	14	0	1	2
D = 60	13	12	0	1	5
D = 100	13	14	1	1	2



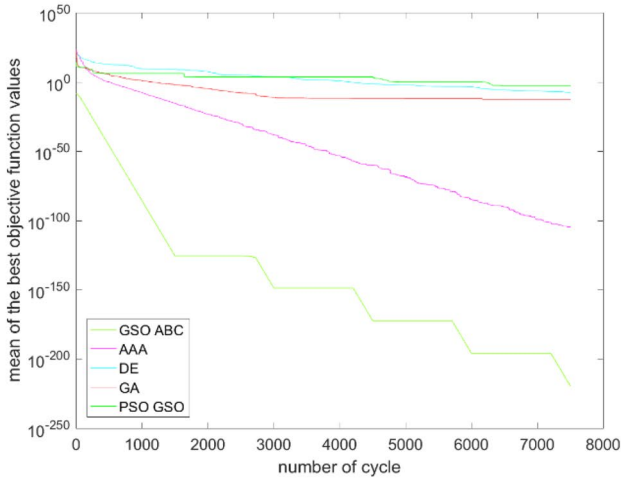
F3, D=30



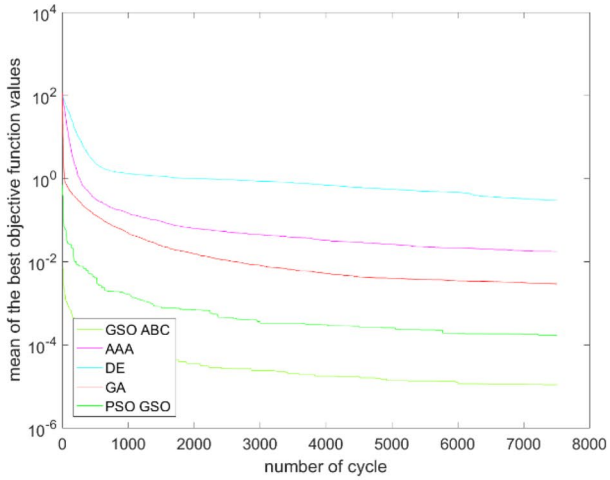
F6, D=30



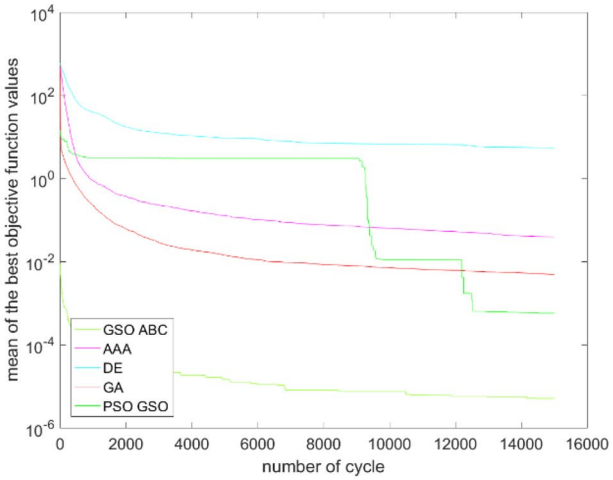
F3, D=100



F4, D=30

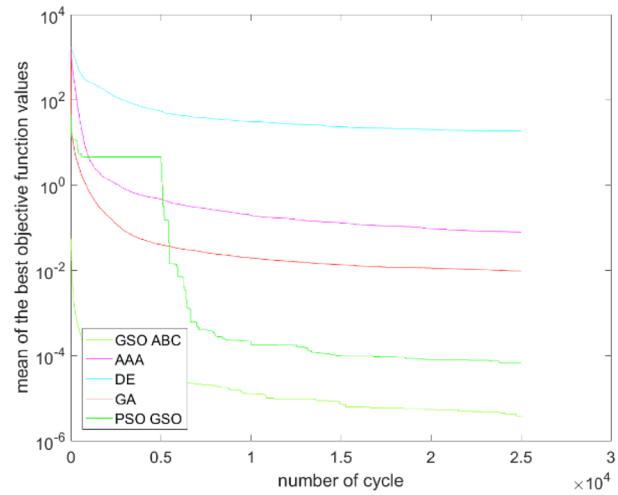


F9, D=30

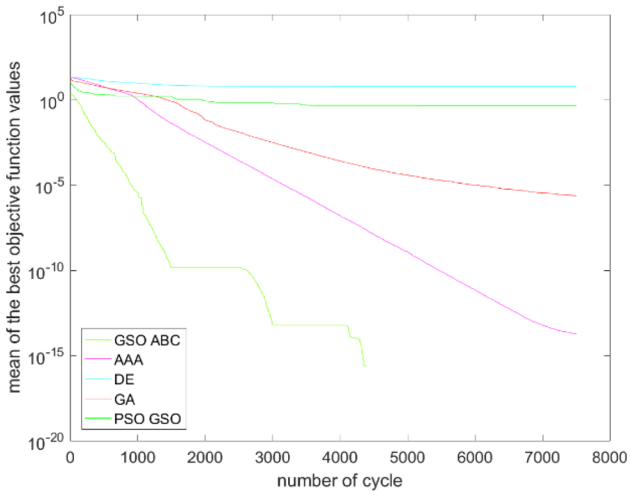


F9, D=60

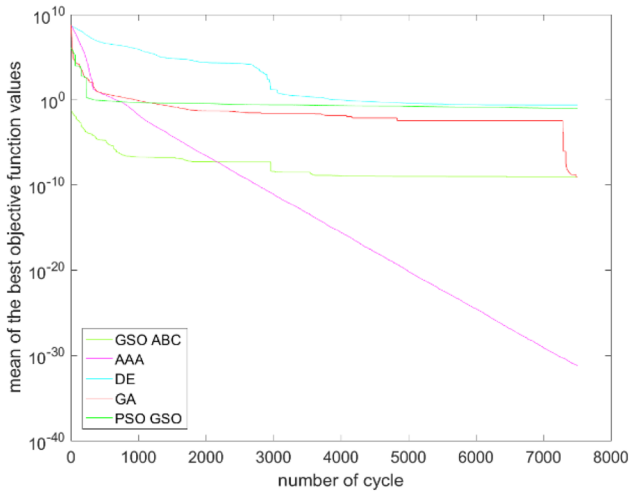
Fig. 6 Comparison of convergence curves



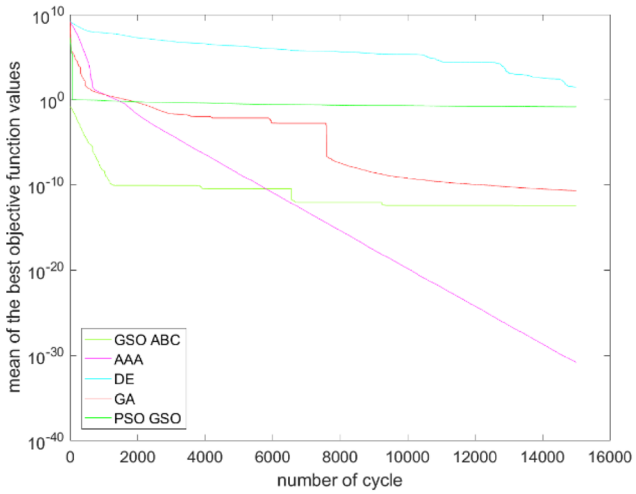
F9, D=100



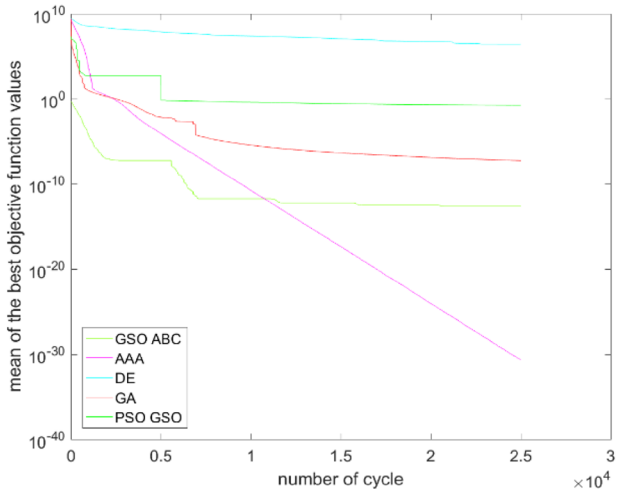
F15, D=30



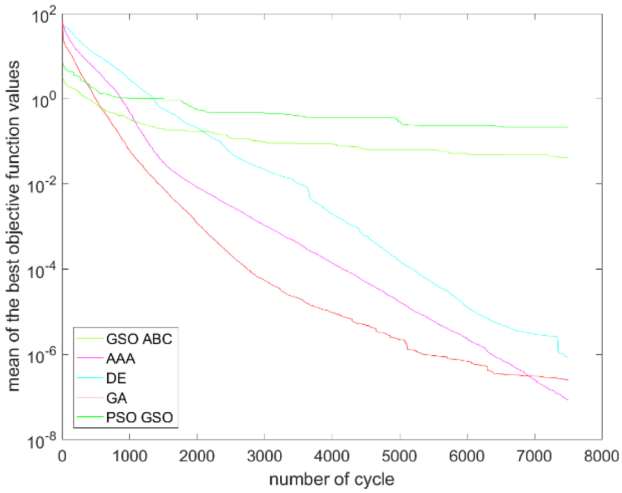
F16, D=30



F16, D=60



F16, D=100



F28, D=30

Fig. 7 Comparison of convergence curves

Table 10 Test results for GSO_ABC, GRABC, GRGABC and GRCABC

F. no.	GSO_ABC		GRABC		GRGABC		GRCABC		DFSABC_elite	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
F1	**1.21E−89**	**4.93E−89**	1.29E−19	9.63E−20	2.08E−37	2.24E−37	1.40E−53	1.51E−53	4.14E−82	8.76E−82
F2	**8.66E−80**	**3.18E−80**	1.50E−11	2.35E−11	4.93E−29	5.72E−29	3.62E−46	4.17E−46	5.37E−78	8.66E−78
F3	**1.42E−92**	**7.55E−92**	2.71E−21	3.75E−21	2.83E−39	2.14E−39	9.29E−55	1.88E−54	2.84E−83	4.66E−83
F4	**1.49E−171**	**0**	1.60E−31	7.07E−31	2.55E−51	9.33E−51	7.93E−59	3.43E−58	2.41E−110	1.19E−109
F5	3.36E−20	1.20E−19	3.99E−12	2.08E−12	2.67E−21	1.93E−21	9.29E−29	5.07E−29	**2.06E−42**	**2.08E−42**
F6	**6.49E−84**	**5.36E−84**	1.07E−01	2.32E−02	7.77E−01	6.99E−01	4.92E−03	2.10E−03	5.08E−07	3.69E−07
F7	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F9	**9.49E−06**	**1.13E−05**	3.32E−02	1.24E−02	1.87E−02	1.35E−03	1.24E−02	4.04E−03	1.20E−02	3.80E−03
F10	23.67066	4.60957	**8.83E−02**	**6.69E−02**	6.09E+00	1.96E+01	3.59E−01	5.99E−01	3.45E+00	1.45E+01
F11	6.91E−11	3.67E−10	1.25E−13	4.31E−13	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F12	**0.00E+00**	**0.00E+00**	1.02E−12	2.09E−12	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F13	**0.00E+00**	**0.00E+00**	8.14E−15	2.72E−14	2.96E−04	1.48E−03	2.59E−14	1.29E−13	**0.00E+00**	**0.00E+00**
F14	2.64E+03	1.26E+03	6.04E−11	2.16E−10	3.20E−12	1.08E−12	**0.00E+00**	**0.00E+00**	4.37E−13	1.09E−12
F15	**2.15E−16**	**7.50E−16**	7.04E−10	1.01E−09	6.50E−15	9.84E−16	2.66E−15	0.00E+00	3.80E−15	1.69E−15
F16	5.63E−11	1.88E−10	2.31E−19	6.28E−19	**1.57E−32**	**5.59E−48**	1.57E−31	5.59E−48	**1.57E−32**	**5.59E−48**
F17	1.29E−08	3.56E−08	1.39E−19	2.11E−19	**1.50E−33**	**0.00E+00**	**1.50E−33**	**0.00E+00**	**1.50E−33**	**0.00E+00**
F18	9.02E−07	1.63E−06	1.21E−07	3.55E−07	3.53E−07	1.20E−06	5.52E−07	2.76E−03	**3.10E−40**	**1.03E−39**
F19	3.13E−06	9.96E−06	1.17E−13	1.44E−13	**1.35E−31**	**2.23E−47**	**1.35E−31**	**2.23E−47**	**1.35E−31**	**2.23E−47**
F20	1.55E+03	1.59E+03	9.71E−03	1.49E−02	4.15E−05	5.26E−05	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**

after 9000th cycle and after GSO_ABC it becomes the second best method. The third best algorithm is GA for that graphic. Same result are valid for F9 D = 100 namely GSO_ABC is the best and GSO_PSO is the second best algorithm after 6000th cycles.

F15 is a difficult multi modal non-separable function. It has multi local optima. All the methods show the same performance until 1700th cycles. After that point GSO_ABC convergence boost up and even after 4500th cycles and only it finds the optimum value. While DE and GSO_PSO draws a flat convergence, performance of the AAA and GA improves with iteration number. In this graphic the second best performance belongs to AAA and the third best performance belongs to GA.

In the graphic of F16 for all different D values, AAA has the best performance. However GSO_PSO finds good solution in beginning it falls behind the AAA. It can be seen that GSO_PSO can escape from local minima by its two phase search mechanism. GSO_ABC is the second best performed algorithm.

F28 graphic for D = 30 all the algorithms except GSO_PSO show the similar performance. At the end AAA found the best results. Although GSO_ABC finds good results at the beginning it can be the fourth best algorithm in the final results.

The proposed GSO_ABC method is compared with the GRABC, GRGABC, GRCABC and DFSABC_elite methods. The results are presented in Table 10. The comparison was made on 19 common benchmark functions.. The studies were carried out with 25 runs and a total of 150,000 maxFEs parameters. As a comparison, the GSO_ABC method yielded the best results in 7 functions. The state of art methods have achieved the best result in 10 functions. The GRABC, GRGABC, GRCABC and DFSABC_elite methods yielded the best results with 2, 6, 7 and 10 functions respectively. In this context, it can be said that the GSO_ABC method is also in a competitive position with state-of-the-art studies.

5 Conclusions

In this work a new optimization method, GSO_PSO algorithm is investigated. The power of the GSO comes from its two phases searching mechanism which makes it possible for both exploration and exploitation. By the two phase architecture every population based algorithm can be used inside it. So performance of other well-known optimization methods are examined both inside and outside GSO framework. Test results show that using ABC in GSO framework gives the best result among the other algorithms. It is planned as establishing knowledge transfer between epochs as future work.

If the subpopulations of the GSO algorithm convergences fast in the first phase, the ability of exploitation of the method is adversely affected. In this case, different subpopulations concentrate on the same area and decrease the population diversity. This is seen as a disadvantage of the GSO method. Future studies can be modified to control the population diversity and improve the exploration ability of the method.

References

1. Aimin F, Wang X, He Y, Wang L (2014) A study on residence error of training an extreme learning machine and its application to evolutionary algorithms. Neurocomputing 146(1):75–82
2. Booker LB, Goldberg DE, Holland JH (1989) Classifier systems and genetic algorithms. Artif Intell 40:235–282. https://doi.org/10.1016/0004-3702(89)90050-7
3. Chunru D, Ng WWY, Wang X et al (2014) An improved differential evolution and its application to determining feature weights in similarity-based clustering. Neurocomputing 146:95–103
4. Colorni A, Dorigo M, Maniezzo V (1992) Distributed optimization by ant colonies. In: From Anim Animat, pp 134–142
5. Cui L, Li GH, Wang XZ, Lin QZ, Chen JY, Lu N, Lu J (2017) A ranking-based adaptive artificial bee colony algorithm for global numerical optimization. Inf Sci 417:169–185. https://doi.org/10.1016/j.ins.2017.07.011
6. Cui LZ, Li GH, Lin QZ, Du ZH, Gao WF, Chen JY, Lu N (2016) A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. Inf Sci 367:1012–1044. https://doi.org/10.1016/j.ins.2016.07.022
7. Cui LZ et al (2017) A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. Inf Sci 414:53–67. https://doi.org/10.1016/j.ins.2017.05.044
8. Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1:3–18. https://doi.org/10.1016/j.swevo.2011.02.002
9. Gao WF, Liu SY (2012) A modified artificial bee colony algorithm. Comput Oper Res 39:687–697. https://doi.org/10.1016/j.cor.2011.06.007
10. Gao WF, Liu SY, Huang LL (2013) A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. IEEE Trans Cybern 43:1011–1024. https://doi.org/10.1109/Tsmcb.2012.2222373
11. Gunduz M, Kiran MS, Ozceylan E (2015) A hierarchic approach based on swarm intelligence to solve the traveling salesman problem. Turk J Electr Eng Computer Sci 23:103–117. https://doi.org/10.3906/elk-1210-147
12. Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
13. Holland JH (1992) Genetic algorithms. Sci Am 267:66–72
14. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471. https://doi.org/10.1007/s10898-007-9149-x
15. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8:687–697. https://doi.org/10.1016/j.asoc.2007.05.007
16. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: 1995 IEEE international conference on neural networks proceedings, vols 1–6, pp 1942–1948. https://doi.org/10.1109/Icnn.1995.488968
17. Kiran MS (2015) TSA: tree-seed algorithm for continuous optimization. Expert Syst Appl 42:6686–6698. https://doi.org/10.1016/j.eswa.2015.04.055
18. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680. https://doi.org/10.1126/science.220.4598.671
19. Li GH, Cui LZ, Fu XH, Wen ZK, Lu N, Lu J (2017) Artificial bee colony algorithm with gene recombination for numerical function optimization. Appl Soft Comput 52:146–159. https://doi.org/10.1016/j.asoc.2016.12.017
20. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evolut Comput 10:281–295. https://doi.org/10.1109/Tevc.2005.857610
21. Locatelli M, Maischberger M, Schoen F (2014) Differential evolution methods based on local searches. Comput Oper Res 43:169–180. https://doi.org/10.1016/j.cor.2013.09.010
22. Mallipeddi R, Suganthan PN, Pan QK, Tasgetiren MF (2011) Differential evolution algorithm with ensemble of parameters and mutation strategies. Appl Soft Comput 11:1679–1696. https://doi.org/10.1016/j.asoc.2010.04.024
23. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evolut Comput 8:204–210. https://doi.org/10.1109/tevc.2004.826074
24. Mernik M, Liu SH, Karaboga D, Crepinsek M (2015) On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation. Inf Sci 291:115–127. https://doi.org/10.1016/j.ins.2014.08.040
25. Moore PW, Venayagamoorthy GK (2006) Empirical study of an unconstrained modified particle swarm optimization. In: 2006 IEEE congress on evolutionary computation, vols 1–6, p 1462
26. Muthiah-Nakarajan V, Noel MM (2016) Galactic swarm optimization: a new global optimization metaheuristic inspired by galactic motion. Appl Soft Comput 38:771–787. https://doi.org/10.1016/j.asoc.2015.10.034
27. Nasir M, Das S, Maity D, Sengupta S, Halder U, Suganthan PN (2012) A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. Inf Sci 209:16–36. https://doi.org/10.1016/j.ins.2012.04.028
28. Parouha RP, Das KN (2016) A memory based differential evolution algorithm for unconstrained optimization. Appl Soft Comput 38:501–517. https://doi.org/10.1016/j.asoc.2015.10.022
29. Parsopoulos KE, Tasoulis DK, Vrahatis MN (2004) Multiobjective optimization using parallel vector evaluated particle swarm optimization. In: Proceedings of the iasted international conference on artificial intelligence and applications, vols 1 and 2, pp 823–828
30. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evolut Comput 13:398–417. https://doi.org/10.1109/Tevc.2008.927706
31. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248. https://doi.org/10.1016/j.ins.2009.03.004
32. Sharma H, Bansal JC, Arya KV (2012) Fitness based differential evolution. Memet Comput 4:303–316. https://doi.org/10.1007/s12293-012-0096-9
33. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359. https://doi.org/10.1023/A:1008202821328

34. Uymaz SA, Tezel G, Yel E (2015) Artificial algae algorithm (AAA) for nonlinear global optimization. Appl Soft Comput 31:153–171. https://doi.org/10.1016/j.asoc.2015.03.003
35. Uymaz SA, Tezel G, Yel E (2015) Artificial algae algorithm with multi-light source for numerical optimization and applications. Biosystems 138:25–38. https://doi.org/10.1016/j.biosystems.2015.11.004
36. Xizhao W, He Q, Chen D, Yeung D (2005) A genetic algorithm for solving the inverse problem of support vector machines. Neurocomputing 68:225–238
37. Yang XS (2010) A new metaheuristic bat-inspired algorithm Nicso 2010. In: Nature inspired cooperative strategies for optimization, vol 284, pp 65–74
38. Zhang JQ, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evolut Comput 13:945–958. https://doi.org/10.1109/Tevc.2009.2014613