An efficient algorithm for function optimization: Modified stem cells algorithm

Article *in* Open Engineering · March 2013

DOI: 10.2478/s13531-012-0047-8

4 authors, including:

Mohammad Taherdangkoo

19 PUBLICATIONS **229** CITATIONS

SEE PROFILE

Mehran Yazdi

Shiraz University

126 PUBLICATIONS **1,373** CITATIONS

SEE PROFILE

Mohammad Hadi Bagheri

Shiraz University of Medical Sciences

14 PUBLICATIONS **170** CITATIONS

SEE PROFILE

Cent. Eur. J. Eng. • 1-15
Proofreading version

Central European Journal of **Engineering**

An efficient algorithm for function optimization: modified stem cells algorithm

Research Article

Mohammad Taherdangkoo[1]*, Mahsa Paziresh[2], Mehran Yazdi[1], Mohammad Hadi Bagheri[3]

1 Department of Communications and Electronics, Faculty of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran

2 RSISE, Australian National University, Canberra, ACT 0200, Australia

2 Center for Evidence-Based Imaging, Department of Radiology, Brigham & Women's Hospital, Harvard Medical School, Brookline, MA, USA

Received 11 March 2012; accepted 05 August 2012

Abstract: In this paper, we propose an optimization algorithm based on the intelligent behavior of stem cell swarms in reproduction and self-organization. Optimization algorithms, such as the Genetic Algorithm (GA), Particle Swarm Optimization (PSO) algorithm, Ant Colony Optimization (ACO) algorithm and Artificial Bee Colony (ABC) algorithm, can give solutions to linear and non-linear problems near to the optimum for many applications; however, in some case, they can suffer from becoming trapped in local optima. The Stem Cells Algorithm (SCA) is an optimization algorithm inspired by the natural behavior of stem cells in evolving themselves into new and improved cells. The SCA avoids the local optima problem successfully. In this paper, we have made small changes in the implementation of this algorithm to obtain improved performance over previous versions. Using a series of benchmark functions, we assess the performance of the proposed algorithm and compare it with that of the other aforementioned optimization algorithms. The obtained results prove the superiority of the Modified Stem Cells Algorithm (MSCA).

Keywords: Optimization algorithm • Modified stem cells algorithm • Particle swarm optimization • Ant colony optimization • Artificial bee colony algorithm • Genetic algorithm

1. Introduction

The meta-heuristic algorithms are mostly derived from the behavior of biological systems (such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Stem Cells Algorithm (SCA)), or physical systems (such as Simulated Annealing (SA)), among others. Some of the main reasons for selection and development of these algorithms are their simplicity in formulation and the good understanding of the development of these processes.

Optimization algorithms have been mainly inspired by natural evolution. The most famous one is the genetic algorithm [1], which is based on the idea of evolution in the nature and looks into a given problem on a fully random basis. This method is based on biological techniques such as genetics and mutation; the search is carried out to find better responses in each generation compared to the previous one. Among the key features of the genetic algorithm are its ability to run in parallel and its capability for searching very large and complex spaces [2].

*E-mail: mtaherdangkoo@yahoo.com

Another well-known optimization algorithm is Particle Swarm Optimization (PSO), first proposed by Kennedy and Eberhart [3], and proven useful for continuous and discrete functions [4]. In this algorithm each particle as a member of the society uses the experiences of its previous particle as well as those of others in order to reach the ultimate goal. This algorithm is able to find the global optimum of the function in question in consequent iterations [5].

The third optimization algorithm that has been considered by many researchers is Ant Colony Optimization (ACO) algorithm [6], used to solve the salesman problem. This algorithm was inspired by studies and observations on the behavior of ants in their colony. Its implementation is based on the behavior of ants when searching food for the survival of their colony, by considering the behavior whereby ants scatter a chemical material called pheromone as they move, which evaporates and impresses the others on their move, so that eventually the ants can reach the food source (optimal response) [7]. One of the disadvantages of this algorithm is its poor criterion for iteration ending, because the process will continue until reaching the maximum number of ants available to search, lacking improvement. In addition it is very dependent on the move transfer function and the pheromone evaporation rate. Although these problems have been solved for many applications, this has been the reason for it not being widely used in the medical engineering field, especially in the medical image segment where generally a unique measure function that is consistent with the terms of this algorithm cannot be designed.

Lastly, the Artificial Bee Colony (ABC) algorithm [8] was inspired by the social behavior of honey bees in finding food. In this algorithm, three types of bees (employer, onlooker and scout) are used, but only the employer bee is capable of becoming a scout bee. Employer bees are responsible for searching a designated space, and they return to the hive and share the gathered information with onlooker bees after storing the location of food sources and its nectar content in its memory. Onlooker bees select an employer bee that contains the highest amount of nectar according to classification of food sources. The employer bee then becomes a scout bee and begins to recruit other bees. It then moves with recruited bees toward the food source in order to drain the nectar. This process will continue until the entire space is covered and the optimal response is achieved. Having low level of computation and fairly good flexibility are considered as the advantages of this algorithm.

In this paper, we have extended the performance of the Stem Cells Algorithm (SCA) for real parameter optimization on unimodal and multimodal functions, and have compared its performance to that of the aforementioned algorithms.

2. Stem cells algorithm

The stem cells algorithm has been introduced as one of the newest optimization algorithms for optimizing numerical functions and data clustering [9, 10]. This algorithm was introduced and implemented on the basis of stem cells behavior in the human body (at the time of entering into the body). In general, finding injured and weak organs is the main goal of this algorithm. Each of these organs is an optimum solution for solving all kinds of optimization problems. In order to implement this algorithm, multidimensional and self-renewal properties of stem cells are used. A brief introduction of the algorithm is as follows. An initial matrix which is composed of the problem variables, namely stem cells properties that are inherent, is formed. For example, we can point to the multidimensional property as the ability of a stem cell transforming into the marrow cells, blood cells, *etc.* The initial matrix is defined as:

$$SC_i = [SC_{i1}, SC_{i2}, \ldots, SC_{iD}] \quad i = 1, 2, \ldots, S \tag{1}$$

where S represents the total number of cells participating in the implementation process of the algorithm and D is the dimension of the problem space.

In this algorithm, an initial population is selected from the members, and then in subsequent iterations new members are added to the old population by a specific percentage fraction. This specific percentage of increment is determined before the implementation of the algorithm. Hence the algorithm starts by using only a part of the total members; if a large population was selected early during the implementation of the optimization algorithm, it would cause numerous iterations, high time consumption and occasionally trapping in local minimums.

The initial population is selected so that its distribution is uniformly and randomly extended in the problem space. Then a cost function for each cell is defined as:

$$Cost(SC_i) = \begin{cases} \dfrac{1}{a + f_i} & f_i \geq 0 \\ 1 + |f_i| & f_i > 0 \end{cases} \tag{2}$$

where a is a positive random number in interval [0,1], but for more cases with normal complex (for example Dimension $(D) < 50$), a is a constant with value 1, and f_i is the cost value of the solution SC_i. For maximization problems, the cost function can be directly used as a fitness function. Then the cost of each cell is normalized by:

$$Cost_N(SC)_n) = Max\big[Cost(SC_i)\big] - Cost(SC_n) \tag{3}$$

where $Cost(SC_n)$ is the cost of the n^{th} stem cell, $Max\left[Cost(SC_i)\right]$ is the maximum cost among stem cells and $Cost_N(SC)_n)$ is the normalized cost of the n^{th} stem cell.

Each cell that has a higher cost is a weaker cell and thus its normalized cost is lower. The ultimate parameter in determining the best stem cell (optimum solution) is the relative potency (*i.e.* the potential of each stem cell in differentiating between different cell types, for example bone marrow cells and blood cells). It is derived as follow:

$$P_n = \frac{Cost_N(SC_n)}{\sum_{i=1}^{S} Cost_N(SC_i)} \tag{4}$$

$\sum_{i=1}^{S} Cost_N(SC_n)$ is the total cost of stem cells. From another point of view, P_n is the comparative power of the n^{th} cell. After calculating comparative power of each cell, its value is saved in the memory of each cell. Then each of these cells shares information saved in its memory and finally this information is classified in a table from highest to the lowest order.

A part of the cells (*e.g.* one third of them) located at the upper range of the table are permitted to participate in self-renewal process and they form a portion of the attended population in a next iteration (*e.g.* 60% of the attended population in a next iteration). The rest of the population is randomly selected from the cells that have no information from the problem space. The self-renewal process is done by the equation below:

$$SC_{Optimum}(t+1) = \zeta \times SC_{Optimum}(t) \tag{5}$$

where t represents each cycle (iteration) and ζ is a random number in the interval [0,1].

For the sake of simplicity in implementing the self-renewal process, which occurs similarly and reciprocally, $\zeta = 0.96$ is set for similar self-renewal and $\zeta = 0.01$ for mutual self-renewal. Also in some problems where problem space dimension is high or the problem is highly complex, one could utilize both self-renewal processes simultaneously in order to prevent uniformity in reaching the optimum solution. When using this algorithm in single variable problems, the goal is the formation of an organ and the algorithm would continue until reaching a complete organ. But when the objective is obtaining optimum values of two or more variables, the goal is to reach to two or more complete organs that justify the multidimensional property of stem cells.

3. Modified stem cells algorithm

As mentioned in the previous section regarding the main algorithm of stem cells, self-renewal process is done either similarly or mutually. Although this process has some advantages in numerical functions, this method can lead to difficulties in applying it to multi-objective functions or data classification. In addition, convergence process slows down and the time to reach the optimum solution or ideal classification can increase. In this work, we significantly improved the algorithm's performance by using Rechenberg's 1/5 Mutation Rule. Here, we improved the self-renewal process by modifying the Equation (5):

$$\begin{aligned} &SC_{Optimum}(t+1) = \zeta(t+1) \times SC_{Optimum}(t) \\ &\zeta(t+1) = \mu(t+1) \times \zeta(t) \\ &\mu(t+1) = \begin{cases} \mu(t) \times 0.85 & \aleph < 1/5 \\ \mu(t)/0.85 & \aleph > 1/5 \\ \mu(t & \aleph = 1/5 \end{cases} \end{aligned} \tag{6}$$

where the initial ζ is determined before the process of algorithm implementation and is a random number in the interval [0,1], and $\aleph$ is also a random number in the interval $[-\mu, \mu]$ and is specified before implementing the algorithm. If the algorithm cannot improve the solution with respect to Rechenberg's 1/5 rule (*i.e.* the ratio of successful mutations to all mutations $\aleph$ is less than 1/5), $\mu(t)$ is decreased. If $\aleph$ is greater than 1/5, then $\mu(t)$ is increased in order to speed up the search.

In the original SCA, after each self-renewal process and in the next iteration, the distribution of cells was considered uniform and random in the whole problem space, but in this new model, the location of each cell self-renewal is in the table in which highly informative cells have been located on the top. Then between each two cells (available space between two cells in the problem space), a Beta distribution is used instead of uniform distribution, in order to create self-renewal process more randomly. Beta distribution process of self-renewal cells is defined as:

$$\begin{aligned} Beta\ Distribution &= \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1} \\ &\quad a < x < b \\ B(\alpha,\beta) &= \int_0^1 x^{\alpha-1}(1-x)^{\beta-1} dx \\ &= 2\int_0^{2\pi} (\sin\theta)^{2\alpha-1}(\cos\theta)^{2\beta-1} d\theta \end{aligned} \tag{7}$$

where a and b are the first and second cells participating in the distribution process and whose positions are specified in the table. α and β are two positive symmetric random numbers ($\alpha, \beta > 0$).

The remaining cells participating in the process of algorithm implementation have no information about the sample space and their distribution happens uniformly and

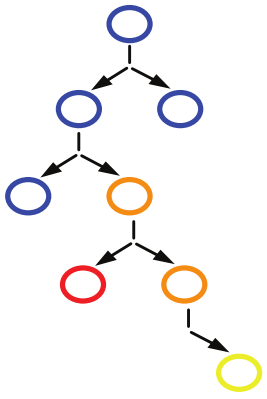


Figure 1. Stem cell self-renewal process, where the color circles shows the stem cell and the yellow circle is the best stem cell in the final iteration when the algorithm convergences to optima.

randomly. Figure 1 shows an example of self-renewal characteristics of stem cells until reaching to the optimum cell [10].

The pseudo code of the modified stem cells optimization algorithm is defined as follows:

Begin

Objective function, $f(x)$, $X=[x_1,x_2,...,x_N]^T$
Initialize population, $X_i,\{i=1,\cdots,N\}$
Define $\alpha,\beta,\zeta_{\min},\zeta_{\max},\mu,\aleph$.
Sort the initial population based on their objective function value by equation (2).
Normalize the cos *t of each cell by equation* (3).
Evaluate the comparative power of each cell by equation (4).
while ($t <$ *Maximum number of iterations*)
Find best stem cell according to fitness value.
Allow best cells to self $-$ *renew by equation* (6).
Beta Distribution of renewal cells in problem space by equation (7).
In each iteration : Find best stem cell, replace with respective stem cell and camparewith stem cells in previous iteration.
If $f(x_i^t) < f(x_{Optimum}^t)$
Accept and save the new solutions in the memory of stem cell
end if
Search for the new solutions in current iteration and compare with the solutions in previous step
end while
end (*obtain the optimum response of the MSCA*)

4. Experimental results

4.1. Benchmark functions

In order to prove our claim related to the better performance of our proposed algorithm than other introduced optimization algorithms, we have compared the efficiency and the accuracy of the MSCA using Benchmark functions. For all Benchmark functions, 50 runs were applied with different random seeds for generating the random variables, each run contains 5000 iterations, and the population size was set to 100 for all optimization algorithms aforementioned except the MSCA. Note that when the space size goes up (for example, Dimension (D) > 30) the performance of the optimization algorithms drop dramatically because the algorithms encounter several optimal solutions. In such cases, selected answer in a considered space randomly adds the complexity to the subject. Table 1 lists the eight Benchmark functions.

4.2. Settings for algorithms

The number of maximum generations and population size are common control parameters of the algorithms. In the experiments conducted, maximum number of generations for the dimensions (D) of 10, 20, 30 and 40 are considered to be 500, 750 and 1000, respectively. Table 2 shows other control parameters of the algorithms and the schemes, along with the values of these control parameters applied for GA, PSO, ACO, ABC and SCA and MSCA.

4.3. Comparison

Fifty runs were applied with different random seeds for generating the random variables, each containing 5000 iterations. Computations were performed in Matlab on a computer running Macintosh OS, two 2.93 GHz 6-Core Intel processors, 64 GB Ram, and ATI Radeon HD 5870 graphics card. Figures 2 to 8 show the experimental results of applying all the algorithms to seven Benchmark functions with different values of parameters. As can be seen, the MSCA for most parameters has better convergence to the minimum of the Benchmark functions than the other optimization algorithms. This proves that MSCA is able to avoid becoming trapped in local optima in the problem space and to achieve the global minimum. In the MSCA, the cells having higher comparative power than the other cells are very good for global optimization, and normal self-renewal process for best selected cells is very efficient for local optimization. Therefore we obtain better performance using MSCA in optimizing unimodal and multimodal functions. For different dimensions (10, 20, 30 and 40) the mean and standard derivation (SD) function values of the best solution found by the algorithms (GA, PSO, ACO, ABC, SCA and MSCA) are shown in Tables A.1 to A.7 in the Appendix.

For comparison between the SCA and the MSCA, different parameters values of these algorithms were used, which are given in Table 3. Figure 9 shows the results when these algorithms are run on the Quartic function. The effect of scalability on the computational complexity of SCA and MSCA for Rosenbrock function as described in [11] has also been computed. Code execution time (T_0), execution time of Rosenbrock function for 200,000 evaluations and for five runs (T_1), and mean of the MSCA execution time on Rosenbrock function for 200,000 evaluations ($\hat{T}_2$) were

Table 1. Benchmark functions

Function	Global Min	Search range	Initial range	Formulae
Sphere	0	$[-100, 100]$	$[-100, 50]$	$\sum_{i=1}^{D} x_i^2$
Rosenbrock	0	$[-2.04, 2.04]$	$[-2.04, 0]$	$\sum_{i=1}^{D-1}\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$
Ackley	0	$[-32.7, 32.7]$	$[-32.7, 16]$	$20+e-20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)}-e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}$
Griewank	0	$[-600, 600]$	$[-600, 200]$	$\frac{1}{4000}\left(\sum_{i=1}^{D}(x_i-100)^2\right)-\left(\prod_{i=1}^{D}\cos\left(\frac{x_i-100}{\sqrt{i}}\right)\right)+1$
Weierstrass	0	$[-0.5, 0.5]$	$[-0.5, 0.2]$	$\sum_{i=1}^{D}\left(\sum_{k=0}^{k_{max}}\left[a^2\cos\left(2\pi b^k(x_i+0.5)\right)\right]\right)-D\sum_{k=0}^{k_{max}}\left[a^k\cos(2\pi b^k 0.5)\right]$, $a=0.5,\ b=3,\ k_{max}=20$
Non-continuous Rastrigin	0	$[-5.12, 5.12]$	$[-5.12, 2]$	$\sum_{i=1}^{D}\left(y_i^2-10\cos(2\pi y_i)+10\right)$, $y_i=\begin{cases}x_i & \lvert x_i\rvert<\frac{1}{2}\\ \frac{round(2x_i)}{2} & \lvert x_i\rvert\geq\frac{1}{2}\end{cases}$
Schwefel	0	$[-500, 500]$	$[-500, 200]$	$418.9829\,D-\sum_{i=1}^{D}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$
Quartic	0	$[-1.28, 1.28]$	$[-1.28, 0]$	$\sum_{i=1}^{D} ix_i^4+rand[0,1]$

Table 2. Values of parameters of each of six algorithms.

Algorithm	Parameters	Value
GA	Population	100
	Crossover	0.95
	Mutation rate	0.001
	Number of Iterations	1000
PSO	Number of Swarm	100
	$\varphi_1=\varphi_2$	2
	W_{min}	0.7
	W_{max}	0.9
	Number of Iterations	500
ACO	Population	100
	Probability threshold for maximum trail	0.95
	Local search probability	0.01
	Evaporation rate	0.01
	Number of Iterations	1000
ABC	Population	100
	Number of Sites Selected for Neighborhood Search	10
	Number of Bees Recruited for Best Sites	5
	Number of Iterations	500
SCA	Population	50
	ζ_{min}	0.01 or 0.98
	ζ_{max}	0.98
	Number of Iterations	500
MSCA	Population	50
	ζ_{min}	0.01 or 0.98
	ζ_{max}	0.98
	$\alpha=\beta$	0.1
	Number of Iterations	500

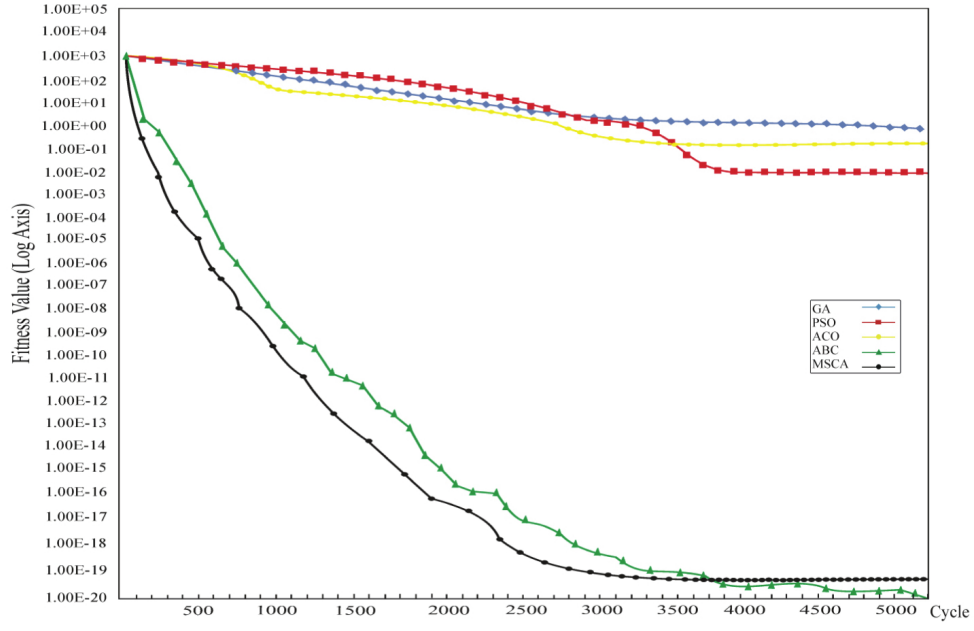


Figure 2. The results of applying different algorithms on Sphere function.

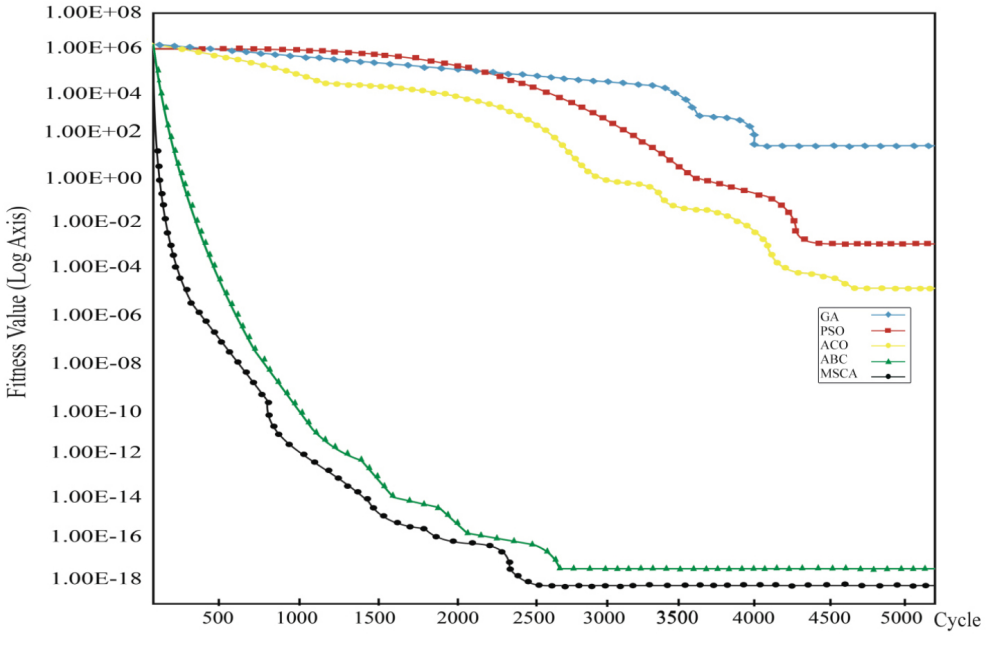


Figure 3. The results of applying different algorithms on Rosenbrock function.

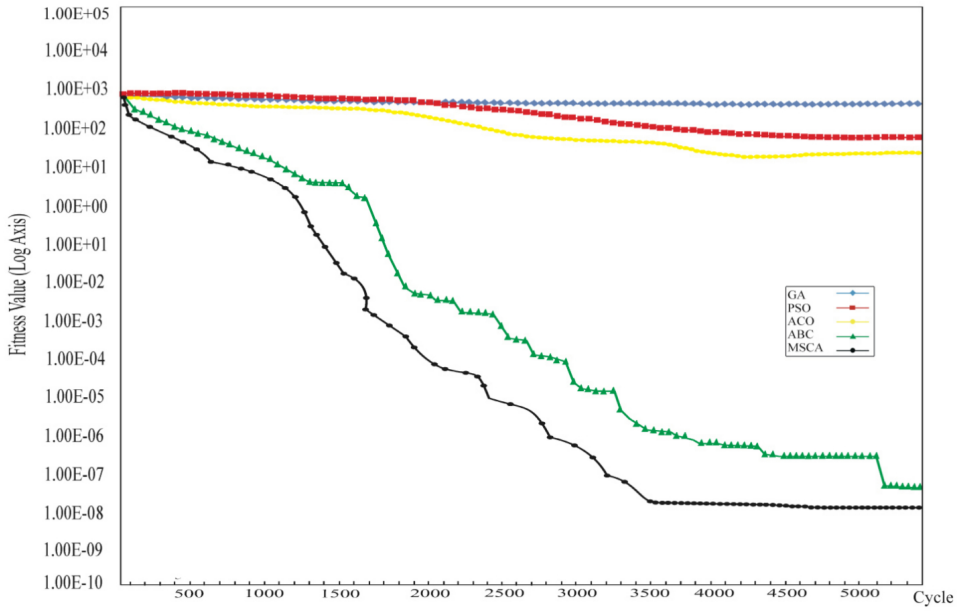


Figure 4. The results of applying different algorithms on Ackley function.

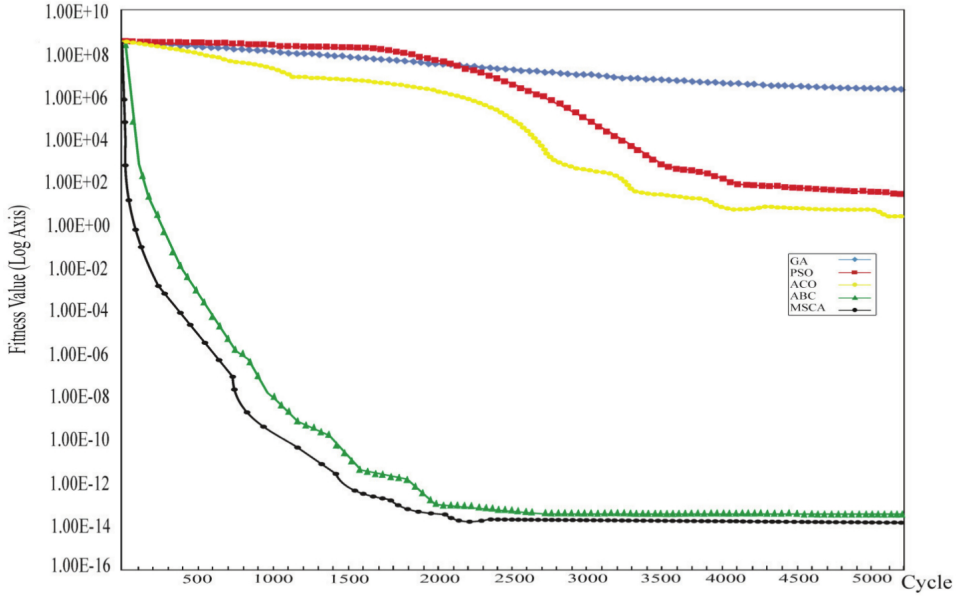


Figure 5. The results of applying different algorithms on Griewank function.

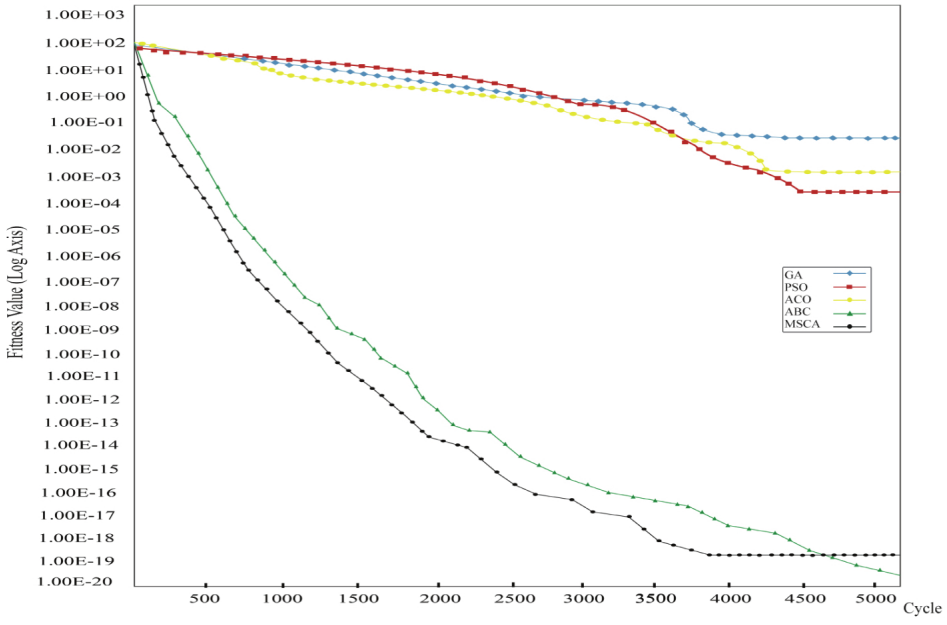


Figure 6. The results of applying different algorithms on Weierstrass function.

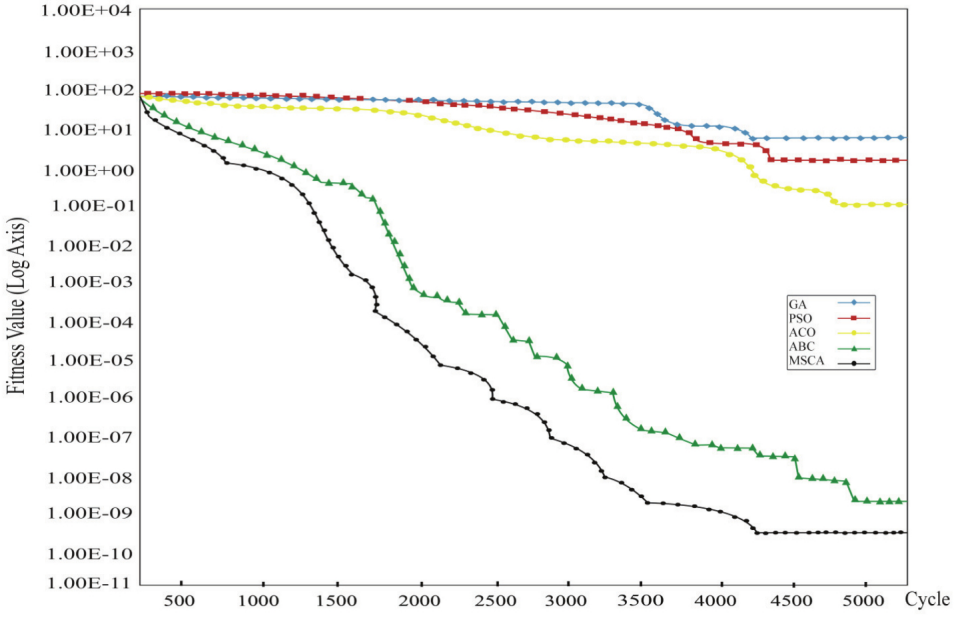


Figure 7. The results of applying different algorithms on Non-continuous Rastrigin function.

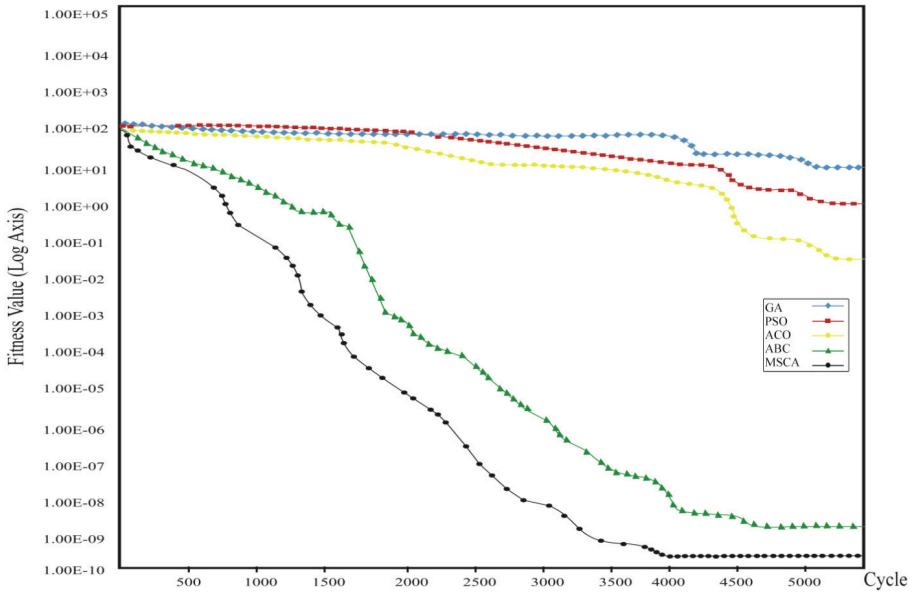


Figure 8. The results of applying different algorithms on Schwefel function.

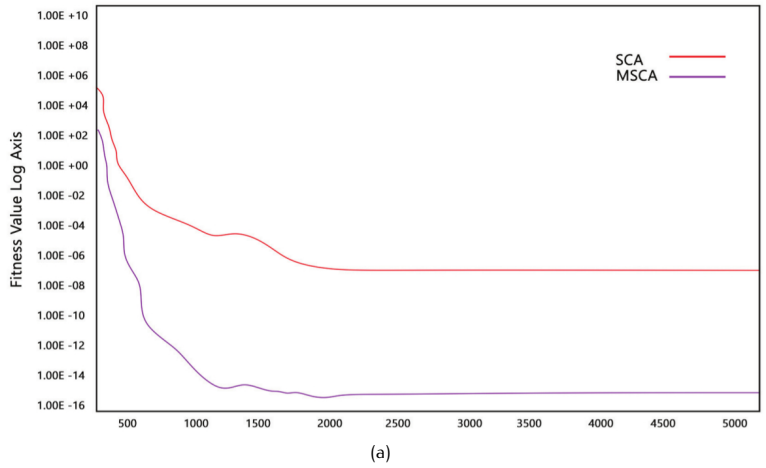
Table 3. Parameter values for comparison between SCA and MSCA.

Algorithm	Parameters and Values	Figure
SCA (1)	Population = 100 $\zeta_{min} = 0.01$ $\zeta_{max} = 0.98$ Number of Iterations = 1000	Fig. 9(a)
SCA (2)	Population = 50 $\zeta_{min} = 0.0236$ $\zeta_{max} = 0.765$ Number of Iterations = 1000	Fig. 9(b)
SCA (3)	Population = 50 $\zeta_{min} = 0.0683$ $\zeta_{max} = 0.587$ Number of Iterations = 1000	Fig. 9(c)
MSCA (1)	Population = 50 $\zeta_{min} = 0.01$ $\zeta_{max} = 0.98$ $\alpha = \beta = 0.1$ Number of Iterations = 500	Fig. 9(a)
MSCA (2)	Population = 100 $\zeta_{min} = 0.0126$ $\zeta_{max} = 0.628$ $\alpha = \beta = 0.17$ Number of Iterations = 1000	Fig. 9(b)
MSCA (3)	Population = 50 $\zeta_{min} = 0.0278$ $\zeta_{max} = 0.0179$ $\alpha = \beta = 1$ Number of Iterations = 500	Fig. 9(c)

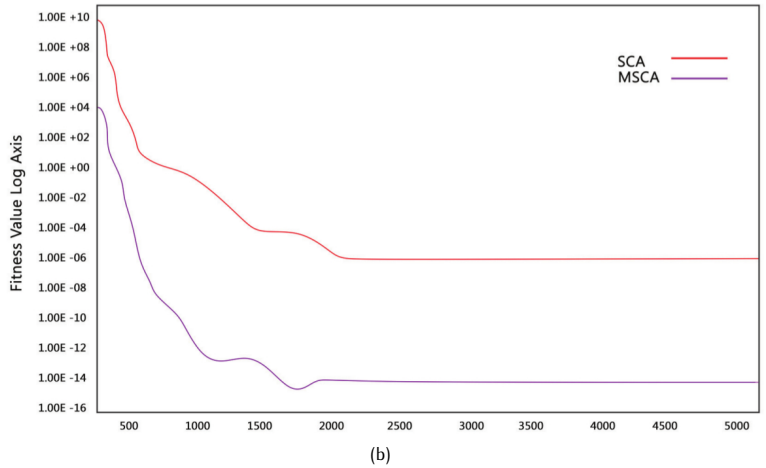
determined. The complexity of algorithm was computed by $\left\{(\hat{T}_2 - T_1)/T_0\right\}$. These results are presented in Tables 4 and 5.

In order to further evaluate the proposed algorithm (MSCA), we considered the performance of the MSCA using 25 functions from Suganthan *et al.* (2005). These 25 functions consist of two categories:

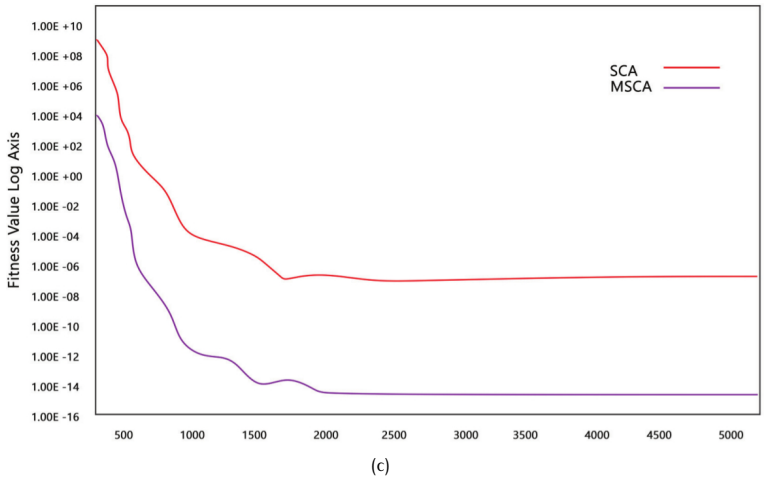
- The first one is Unimodal: F_1 (Shifted Sphere Function), F_2 (Shifted Schwefel's problem 1.2), F_3 (Shifted Rotated High Conditioned Elliptic Function), F_4 (Shifted Schwefel's Problem 1.2 with Noise in Fitness), F_5 (Schwefel's Problem 2.6 with global Optimum on Bounds).
- The second one is Multimodal: Basic Functions: F_6 (Shifted Rosenbrock's Function), F_7 (Shifted Rotated Griewank's Function without Bounds), F_8 (Shifted Rotated Ackley's Function with Global Optimum on Bounds), F_9 (Shifted Rastrigin's Function), F_{10} (Shifted Rotated Rastrigin's Function), F_{11} (Shifted Rotated Weierstrass Function), F_{12} (Shifted Schwefel's Problem 2.13); Expanded Functions: F_{13} (Expanded Extended Griewank's plus Rosenbrock's Function), F_{14} (Shifted Rotated Expanded Scaffer's); Hybrid Composition Functions: F_{15} (Shifted Rosenbrock's Function), F_{16} (Shifted Rotated Griewank's Function without Bounds), F_{17} (Shifted Rotated Ackley's Function with Global Optimum on Bounds), F_{18} (Shifted Rastrigin's Function), F_{19} (Shifted Rotated Rastrigin's Function), F_{20} (Shifted Rotated Weierstrass Function), F_{21} (Shifted Schwefel's Problem 2.13, F_{22} (Shifted Schwefel's Problem 2.13, F_{23} (Shifted Schwefel's Problem 2.13, F_{24} (Shifted Schwefel's Problem 2.13, F_{25} (Shifted Schwefel's Problem 2.13).



(a)



(b)



(c)

Figure 9. Comparison between SCA and MSCA with different parameters values on Quartic function.

Table 4. Time Complexity of the SCA on Rosenbrock function.

Dimension	T_0	T_1	$\hat{T}_2 = Mean(T_2)$	Complexity $\left\{(\hat{T}_2 - T_1)/T_0\right\}$
10	0.39927817625823	0.20988767289342	0.57988724772683	0.92667
30	0.39927817625823	0.24738567228916	0.89625433462671	1.6251
50	0.39927817625823	0.27345764267899	0.90256784472897	1.6002
100	0.39927817625824	0.46986606725426	1.12873455627842	1.6501

Table 5. Time Complexity of the SCA on Rosenbrock function.

Dimension	T_0	T_1	$\hat{T}_2 = Mean(T_2)$	Complexity $\left\{(\hat{T}_2 - T_1)/T_0\right\}$
10	0.31672837928782	0.19256673894299	0.37909441607317	0.58892
30	0.31672837928782	0.19875367435536	0.44652395090463	0.78228
50	0.31672837928782	0.21894465785268	0.53537444385587	0.99891
100	0.31672837928782	0.26987782534622	0.70819822944263	1.38391

The results of applying the proposed algorithms on the 25 functions are shown in Figures 10 to 14. Moreover the results of applying other algorithms in comparison with the proposed algorithm for a particular function are shown in Figure 15. Figure 10 to 15 show the error, which is the mean difference between original functions and those estimated by an algorithm versus function evaluations. This performance evaluation is a standard assessment used in most reported studies.

Most of the mentioned optimization algorithms in this paper achieved good results but not as well as the presently proposed method (MSCA) since the other algorithms are intensively depended on the adjustment of their parameters. For instance, for the ABC algorithm, we need to change its parameters (for example, MR, SF, *etc.*) for each function to obtain a good result for that function. The obtained results shown in Figures 1010 to 15 confirm our argument.

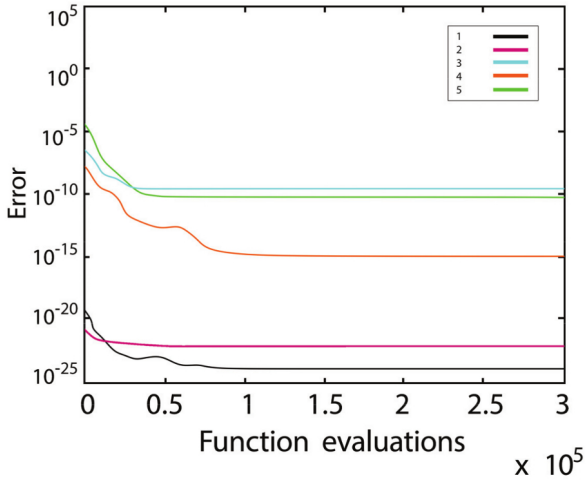


Figure 10. The results of applying the MSCA on F_1–F_5 (Convergence of Functions F_1–F_5).

5. Discussion

The genetic algorithm implements the law of survival of the fittest with a focus on the existing solutions in order to reach a better solution. The genetic algorithm suffers a lot from too much dependency on such functions as selection, mutation, crossover, *etc.*, and on the conditions of the problem and the initial conditions; a weak selection of these constraints and parameters has a remarkable influence on the function of the genetic algorithm (GA). Despite the existence of methods to improve the GA, it is still suboptimal (unresponsive, poorly responsive, or weak) for problems with continuous and discontinuous spaces, with high dimensions of early convergence, or repeated interruptions. For instance, when some weak members join the referenced set and the continuation of examinations in the sample space is stopped, a most appropriate chromosome is selected for reproduction. As a result, the offspring becomes similar to its parent and thus the chromosomes become very similar to each other. As a result, before reaching the optimal solution, early convergence is formed.

The particle swarm optimization (PSO) algorithm has a very simple implementation, but it also suffers from early convergence. Although the PSO algorithm has a more reasonable speed than other optimization algorithms [12], it cannot optimize the quality of solutions by increasing the number of iterations. This issue is more visible when

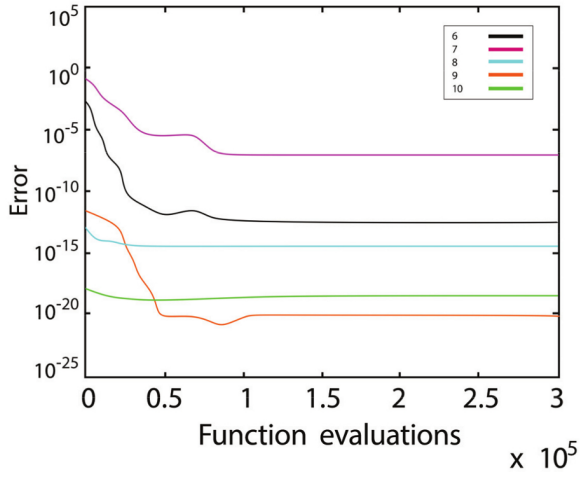


Figure 11. The results of applying the MSCA on F_6–F_{10} (Convergence of Functions F_6–F_{10}).

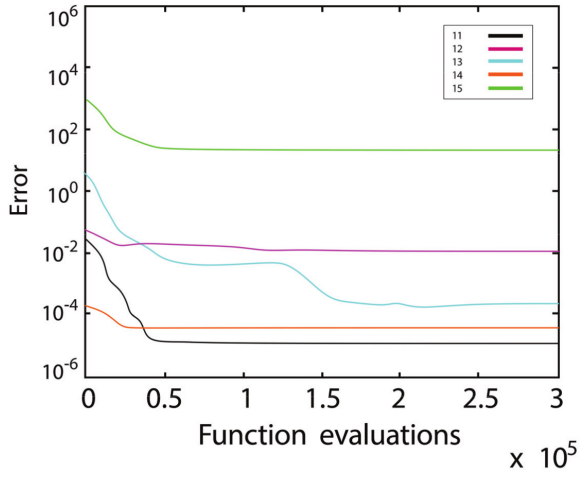


Figure 12. The results of applying the MSCA on F_{11}–F_{15} (Convergence of Functions F_{11}–F_{15}).

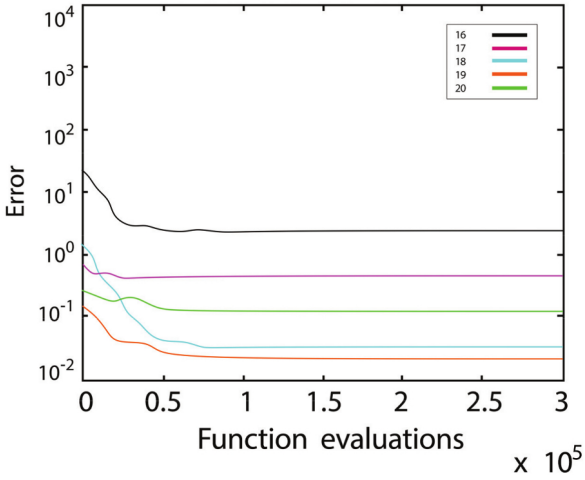


Figure 13. The results of applying the MSCA on F_{16}–F_{20} (Convergence of Functions F_{16}–F_{20}).

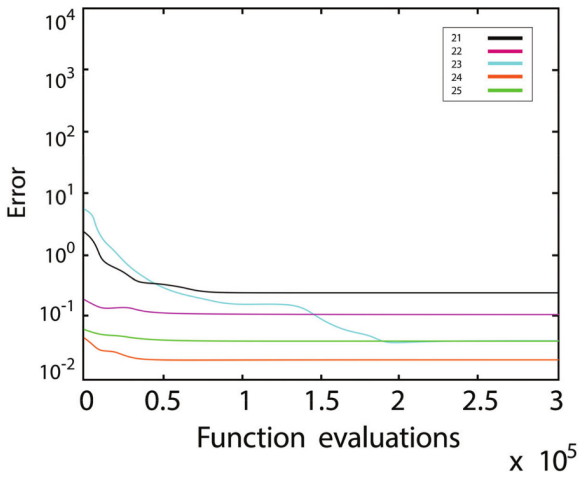


Figure 14. The results of applying the MSCA on F_{21}–F_{25} (Convergence of Functions F_{21}–F_{25}).

examining and optimizing multi-model problems. The reason for its occurrence in $_{gbest}$PSO is that the particles become convergent in one specific point, while this point is located on one line, between the best global position and the best individual position. Other problems arise from too much dependency on regulating the PSO algorithm parameters.

The ant colony optimization (ACO) algorithm has a high dependency on the pheromone evaporation rate and the transfer function, which impacts on the quality of the solution. In this algorithm, by implementing restrictions, efforts have been made to optimize its performance, but the problem of early convergence, when the dimensions of the problem are high, has not been solved, both in continuous and discrete spaces [7].

The artificial bee colony (ABC) algorithm has more accuracy and speed than other optimization algorithms, but the use of a roulette wheel in selecting the employed bees and the relation of employed bees and onlooker bees has not been implemented into a model appropriately. Therefore this limitation does distinguish this algorithm from other optimization algorithms. Research to modify the basics of the ABC algorithm has able to solve its main limitation by using control of phase and magnitude, and has achieved reasonable results [13]. It should be mentioned that in our study the same algorithm has been used for comparison of proposed methods, but they have been included within the context as ABC algorithm.

Other optimization algorithms exist such as the Bat algorithm [14], the Firefly algorithm [15], among others. Al-

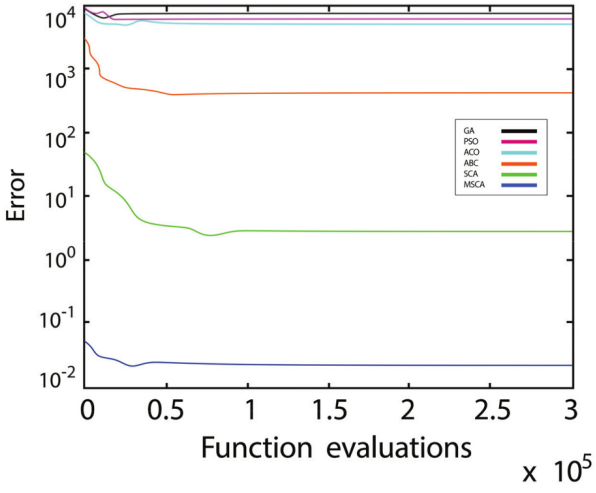


Figure 15. The results of applying different algorithms on F_{22} (Convergence of Function F_{22} for all algorithms).

though these recently developed algorithms have solved many of the problems arising from the past optimization algorithms, because of their use of relatively more constraints and conditions, they have more complexity in implementation. The Bat algorithm has been able to combine and use the advantages of other optimization algorithms with echolocation of the bat, but it has given much complexity to the algorithm and high dependency on the initial conditions and constraints of the problem.

In this paper, we attempted to cover all major problems extracted from other optimization algorithms and set forth a new method for optimization. For instance, with the idea of placing population in one area and using them, we have prevented in each implementation of the algorithm the monotony and consequently the early convergence from occurring in a single iteration. This approach causes a sort of mutational improvement of the algorithm, while no additional constraint has been used in implementing it and the simplicity and flexibility of the algorithm is maintained. The observation of the presented results is a good evidence of the abovementioned issues.

6. Conclusion

The optimization algorithms mostly inspired by the natural evolutions can produce very good solutions in many complex applications where the traditional algorithms normally fail. However, the complexity of their implementation and exhaustive time for their convergence has limited their wide applicability. In this paper, we have proposed a new optimization algorithm, the Modified Stem Cells Algorithm (MSCA), which can successfully overcome the existing limitations in the previously introduced optimization algorithms, including the genetic algorithm (GA), ant colony algorithm (ACO), particle swarm optimization (PSO) algorithm, and artificial bee colony (ABC) algorithm. We have evaluated the proposed algorithm on a series of benchmark functions and the obtained results that, when compared with other optimization algorithms, have demonstrated the superior performance of the MSCA.

Abbreviations

Genetic Algorithm	GA
Particle Swarm Optimization	PSO
Ant Colony Optimization	ACO
Artificial Bee Colony Algorithm	ABC
Stem Cells Algorithm	SCA
Modified Stem Cells Algorithm	MSCA

Appendix A

Table A.1. The results of GA, PSO, ACO, ABC, SCA and MSCA for Sphere function.

Dimensions		10	20	30	40
GA	Mean	0.0046	0.0088	0.0167	0.0796
	SD	0.004	0.0057	0.0097	0.0356
PSO	Mean	6.16E-16	7.33E-11	3.87E-07	9.22E-06
	SD	2.32E-14	2.76E-13	3.84E-09	3.78E-09
ACO	Mean	0.0083	0.0103	0.0534	0.0893
	SD	0.0061	0.0087	0.0102	0.0367
ABC	Mean	9.87E-18	1.26E-16	3.77E-16	4.65E-14
	SD	4.21E-16	7.78E-16	8.36E-16	3.81E-13
SCA	Mean	8.26E-18	8.91E-17	0.17E-16	2.13E-14
	SD	2.78E-16	9.07E-16	0.06E-15	6.01E-13
MSCA	Mean	7.17E-19	4.26E-18	6.57E-17	6.14E-16
	SD	4.78E-18	4.98E-17	7.86E-16	8.21E-14

Table A.2. The results of GA, PSO, ACO, ABC, SCA and MSCA for Rosenbrock function.

Dimensions		10	20	30	40
GA	Mean	2.563	7.865	13.757	23.456
	SD	1.763	3.678	9.457	17.642
PSO	Mean	0.876	1.342	9.453	17.453
	SD	1.124	3.751	7.563	15.674
ACO	Mean	5.753	11.456	19.453	27.567
	SD	4.436	8.456	16.456	31.456
ABC	Mean	1.25E-02	1.76E-00	7.14E+1	2.16E+3
	SD	2.36E-02	5.76E-00	1.17E+1	0.68E+2
SCA	Mean	1.76E-03	3.36E-02	2.88E-01	1.09E+1
	SD	9.25E-03	0.22E-01	2.27E-00	6.23E+1
MSCA	Mean	0.06E-03	1.14E-03	4.46E-01	7.66E-00
	SD	1.02E-04	7.26E-03	3.77E-03	7.66E-01

Table A.3. The results of GA, PSO, ACO, ABC, SCA and MSCA for Ackley function.

Dimensions		10	20	30	40
GA	Mean	0.63	0.89	1.13	3.78
	SD	0.32	0.35	0.37	0.57
PSO	Mean	6.8E-12	4.6E-07	3.6E-06	3.2E-03
	SD	7.2E-13	6.2E-08	5.7E-06	6.1E-02
ACO	Mean	0.28	0.56	0.63	0.97
	SD	0.29	0.31	0.42	0.73
ABC	Mean	0.26E-24	1.72E-19	8.17E-11	6.21E-09
	SD	1.35E-22	6.01E-17	6.12E-12	1.65E-09
SCA	Mean	1.73E-23	3.47E-20	8.1E-14	0.01E-11
	SD	4.78E-22	8.11E-20	3.62E-13	2.02E-10
MSCA	Mean	0.71E-28	9.23E-21	1.01E-16	1.26E-13
	SD	1.01E-25	2.56E-19	1.07E-14	1.28E-14

Table A.4. The results of GA, PSO, ACO, ABC, SCA and MSCA for Griewank function.

Dimensions		10	20	30	40
GA	Mean	0.069	1.01	1.29	2.79
	SD	0.05	0.015	0.123	0.283
PSO	Mean	0.0065	0.0014	0.0017	0.024
	SD	0.0037	0.0026	0.0048	0.014
ACO	Mean	0.036	0.096	0.086	0.097
	SD	0.019	0.023	0.028	0.042
ABC	Mean	1.02E-03	1.27E-03	1.77E-02	1.34E-01
	SD	1.22E-04	1.38E-03	1.29E-02	2.88E-02
SCA	Mean	0.38E-05	0.28E-04	0.02E-02	0.14E-01
	SD	7.55E-04	2.77E-03	1.08E-02	0.08E-02
MSCA	Mean	1.17E-06	2.23E-06	6.11E-05	7.11E-03
	SD	2.56E-05	7.31E-05	1.17E-04	1.27E-04

Table A.5. The results of GA, PSO, ACO, ABC, SCA and MSCA for Weierstrass function.

Dimensions		10	20	30	40
GA	Mean	0.123	0.632	1.352	7.653
	SD	0.023	1.865	2.768	4.786
PSO	Mean	0.0002	0.0096	0.0167	1.677
	SD	0.0034	0.0082	0.0734	0.0962
ACO	Mean	0.008	0.028	0.787	1.256
	SD	0.001	0.006	0.154	1.234
ABC	Mean	0.22E-22	6.92E-19	1.24E-14	6.98E-11
	SD	1.06E-21	3.76E-17	7.27E-13	2.67E-09
SCA	Mean	3.66E-20	7.02E-18	4.16E-13	1.92E-10
	SD	0.03E-19	1.04E-16	8.27E-13	0.07E-08
MSCA	Mean	6.14E-29	1.18E-24	1.22E-18	2.66E-16
	SD	0.82E-27	1.87E-23	2.13E-17	4.19E-16

Table A.6. The results of GA, PSO, ACO, ABC, SCA and MSCA for Non-continuous Rastrigin function.

Dimensions		10	20	30	40
GA	Mean	0.165	0.945	3.768	11.435
	SD	0.734	1.674	8.546	14.35
PSO	Mean	0.000023	0.000578	0.00124	0.0235
	SD	0.000012	0.000342	0.00676	0.0134
ACO	Mean	0.000017	0.000427	0.00036	0.0016
	SD	0.000009	0.000127	0.00379	0.00231
ABC	Mean	3.67E-16	1.22E-11	2.84E-08	2.55E-04
	SD	1.88E-15	2.76E-13	4.56E-09	1.08E-04
SCA	Mean	1.28E-16	8.22E-14	9.82E-11	1.64E-07
	SD	0.26E-16	0.74E-14	0.17E-12	0.03E-07
MSCA	Mean	2.16E-18	4.76E-17	8.77E-14	2.67E-11
	SD	1.66E-17	8.56E-16	1.23E-11	2.83E-11

Table A.7. The results of GA, PSO, ACO, ABC, SCA and MSCA for Schwefel function.

Dimensions		10	20	30	40
GA	Mean	10.844	22.783	41.983	78.564
	SD	9.032	17.637	36.272	47.782
PSO	Mean	2.567	6.547	11.365	21.457
	SD	1.098	3.093	7.543	18.366
ACO	Mean	5.844	12.043	17.634	32.764
	SD	3.536	7.737	11.635	17.623
ABC	Mean	2.67E-03	3.75E-01	7.63E+01	8.71E+03
	SD	3.03E-03	2.11E-02	5.34E+1	6.34E+02
SCA	Mean	1.77E-04	8.02E-02	0.03E+00	1.54E+01
	SD	11.23E-03	6.18E-02	5.34E+00	4.04E+01
MSCA	Mean	1.67E-05	1.49E-03	3.88E-01	7.65E-01
	SD	4.64E-05	7.35E-04	3.65E-02	2.35E-02

References

[1] Holland J.H., Adaptive in Natural and Artificial Systems, University of Michigan, Ann Arbor, Michigan, USA, 1975

[2] Goldberg D., Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA, 1989

[3] Kennedy J., Eberhart R.C., Particle Swarm Optimization, In: Proc. IEEE International Conference of Neural Network, Australia, 1995, 1942–1948

[4] Clerc M., Kennedy J., The Particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput., 2002, 6(1), 58–73

[5] Chakraborty P., Das S., Roy G.G., Abraham A., On convergence of multi-objective particle swarm optimizer, Inform. Sci., 2010, 181(8), 1411–1425

[6] Dorigo M., Optimization, Learning and Natural Algorithms, PhD Thesis, Politecnico di Milano, Milan, Italy, 1992

[7] Dorigo M., Stutzle T., Ant Colony Optimization: Overview and Recent Advances, Handbook of Metaheuristics, 2010, 146, 227–263

[8] Karaboga D., Basturk B., A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm, J. Global Optim., 2007, 39(3), 459–471

[9] Taherdangkoo M., Yazdi M., Bagheri M.H., Stem Cells Optimization Algorithm, LNBI, 2011, 6840, 394–403

[10] Taherdangkoo M., Yazdi M., Bagheri M.H., A Powerful and Efficient Evolutionary Optimization Algorithm based on Stem Cells Algorithm for Data Clustering, Cent. Eur. J. Comput. Sci., 2012, 2(1), 47–59

[11] Suganthan P.N., Hansen N., Liang J.J., Deb K., Chen Y.P.A., Auger, Tiwari S., Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization, Technical Report, Nanyang Technological University, Singapore, 2005, available at http://www.ntu.edu.sg/home/EPNSugan

[12] Deep K., Bansal J.C., Mean particle swarm optimization for function optimization, Int. J. Comput. Intel. Stud., 2009, 1(1), 72–92

[13] Akay B., Karaboga D., A modified Artificial Bee Colony algorithm for real-parameter optimization, Inform. Sci., 2012, 192, 120–142

[14] Yang X.-S., A New Metaheuristic Bat-Inspired Algorithm. Nature Inspired Cooperative Strategies for Optimization, Stud. Comput. Intel., 2010, 284, 65–74

[15] Yang X.-S., Firefly Algorithms for Multimodal Optimization, Stochastic Algorithms: Foundations and Applications, LNCS, 2009, 5792, 169–178