Some Modifications to Enhance the Performance of Artificial Bee Colony

Tarun Kumar Sharma
Indian Institute of Technology
Roorkee, India
taruniitr1@gmail.com

Millie Pant
Indian Institute of Technology
Roorkee, India
millifpt@iitr.ernet.in

Jagdish Chand Bansal
ABV-Indian Institute of Information Technology & Management
Gwalior, India
jcbansal@gmail.com

Abstract—**The Artificial Bee Colony (ABC) algorithm, proposed by Karaboga in 2005 for real-parameter optimization, is a recently introduced optimization algorithm which simulates the foraging behaviour of a bee colony. The proposed variant employs colony size (population size) reduction mechanism during the evolutionary process. Then modification is done to enhance the perturbation scheme. Further, in order to improve the population diversity and avoid the premature convergence, rank selection strategy is applied and analyzed through simulation. The results show that the modified algorithm outperforms the basic ABC algorithm.**

Keywords- Artificial Bee Colony, Colony Size Reduction, Diversity, Rank Selection

I. INTRODUCTION

Bonabeau defined swarm intelligence as ‘‘any attempt to design algorithms or distributed problem solving devices inspired by the collective behaviour of the social insect colonies and other animal societies” [1]. Swarm intelligence has some advantages such as scalability, fault tolerance, adaptation, speed, modularity, autonomy, and inherent parallelism [2].

The key components of swarm intelligence are self-organization and division of labor. Forage selection depends on recruitment for and abandonment of food sources. Swarm intelligence has been popularly used for solving the optimization problems; common algorithms being Particle Swarm optimization (PSO), Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC) algorithm. Out of these, ABC is the most recently proposed algorithm based on the systematic foraging behavior of honey bees

The ABC algorithm was first applied to numerical optimization [3]. Performance of the ABC algorithm was compared to Genetic Algorithm (GA), Particle Swarm Inspired Evolutionary Algorithm (PS-EA) [4-5]; and to Differential Evolution (DE), PSO and Evolutionary Algorithm (EA) on a limited number of basic test problems [6-7]. The effect of region scaling on algorithms including ABC, DE and PSO algorithms was studied in [8]. It was observed that ABC is as competitive as and sometimes even better than its counterpart algorithms.

However, it was also observed that despite having some positive features, the performance of ABC is not always up to the mark. Like other population based stochastic algorithms, it has some structural drawbacks which effects the convergence rate and sometimes also the quality of solution. These drawbacks become more prominent in case multimodal problems having several optima.

In this work, some modifications to the standard ABC algorithm are proposed to further improve its performance. These modifications are based on reducing the colony size; maintaining the perturbation scheme; and using a rank selection strategy for maintaining diversity.

The rest of the paper is organized as follows. In Section 2, the ABC Algorithm is described. The modifications to the basic ABC algorithm are introduced in Section 3. In Section 4, parameter settings & test bed are given and the simulations results are discussed in Section 5. Finally, the paper concludes with Section 6.

II. ARTIFICIAL BEE COLONY

The Artificial Bee Colony (ABC) algorithm, proposed by Karaboga in 2005 for real-parameter optimization, is a recently introduced optimization algorithm which simulates the foraging behaviour of a bee colony [3].

ABC algorithm simulates consists of three kinds of bees: employed bees, onlooker bees and scout bees. Half of the colony consists of employed bees, and the other half includes onlooker bees. Employed bees are responsible for exploiting the nectar sources explored before and giving information to the waiting bees (onlooker bees) in the hive about the quality of the food source sites which they are exploiting. Onlooker bees wait in the hive and decide on a food source to exploit based on the information shared by the employed bees. Scouts either randomly search the environment in order to find a new food source depending on an internal motivation or based on possible external clues [9].

This emergent intelligent behaviour in foraging bees can be summarized as follows [12]:

1) At the initial phase of the foraging process, the bees start to explore the environment randomly in order to find a food source.

2) After finding a food source, the bee becomes an employed forager and starts to exploit the discovered source. The employed bee returns to the hive with the nectar and unloads the nectar. After unloading the nectar, she can go back to her discovered source site directly or she can share information about her source site by performing a dance on the dance area. If her source is exhausted, she becomes a scout and starts to randomly search for a new source.

3) Onlooker bees waiting in the hive watch the dances advertising the profitable sources and choose a source site depending on the frequency of a dance proportional to the quality of the source.

In the ABC algorithm proposed by Karaboga, the position of a food source represents a possible solution of the optimization problem, and the nectar amount of a food source corresponds to the fitness of the associated solution. Each food source is exploited by only one employed bee. In other words, the number of employed bees is equal to the number of food sources existing around the hive (number of solutions in the population). The employed bee whose food source has been abandoned becomes a scout. Using the analogy between emergent intelligence in foraging of bees and the ABC algorithm, the units of the basic ABC algorithm can be explained as follows:

2.1. Producing initial food source sites

If the search space is considered to be the environment of the hive that contains the food source sites, the algorithm starts with randomly producing food source sites that correspond to the solutions in the search space. Initial food sources are produced randomly within the range of the boundaries of the parameters.

$$x_{ij} = x^{j}_{\min} + rand(0,1)(x^{j}_{\max} - x^{j}_{\min}) \tag{1}$$

where i = 1. . .*SN*, j = 1. . .*D*. *SN* is the number of food sources and *D* is the number of optimization parameters. In addition, counters which store the numbers of trials of solutions are reset to 0 in this phase. After initialization, the population of the food sources (solutions) is subjected to repeat cycles of the search processes of the employed bees, the onlooker bees and the scout bees. Termination criteria for the ABC algorithm might be reaching a maximum cycle number (MCN) or meeting an error tolerance (ε).

2.2. Sending employed bees to the food source sites

As mentioned earlier, each employed bee is associated with only one food source site. Hence, the number of food source sites is equal to the number of employed bees. An employed bee produces a modification on the position of the food source (solution) in her memory depending on local information (visual information) and finds a neighboring food source, and then evaluates its quality. In ABC, finding a neighboring food source is defined by (2):

$$v_{ij} = x_{ij} + r_{ij}(x_{ij} - x_{kj}) \tag{2}$$

Within the neighborhood of every food source site represented by x_i, a food source v_i is determined by changing one parameter of x_i. In Eq. (2), j is a random integer in the range [1, D] and k $\in$ {1, 2 . . . SN} is a randomly chosen index that has to be different from i. r_{ij} is a uniformly distributed real random number in the range [-1, 1].

After producing v_i within the boundaries, a fitness value for a minimization problem can be assigned to the solution v_i by (3).

$$fitness_i = \begin{cases} \dfrac{1}{(1+f_i)} & if\ f_i \geq 0 \\ 1 + abs(f_i) & if\ f_i < 0 \end{cases} \tag{3}$$

where f_i is the cost value of the solution v_i. For maximization problems, the cost function can be directly used as a fitness function. A greedy selection is applied between x_i and v_i; then the better one is selected depending on fitness values representing the nectar amount of the food sources at x_i and v_i. If the source at v_i is superior to that of x_i in terms of profitability, the employed bee memorizes the new position and forgets the old one. Otherwise the previous position is kept in memory. If x_i cannot be improved, its counter holding the number of trials is incremented by 1, otherwise, the counter is reset to 0.

2.3. Calculating probability values involved in probabilistic selection

After all employed bees complete their searches, they share their information related to the nectar amounts and the positions of their sources with the onlooker bees on the dance area. This is the multiple interaction features of the artificial bees of ABC. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source site with a probability related to its nectar amount. This probabilistic selection depends on the fitness values of the solutions in the population. A fitness-based selection scheme might be a roulette wheel, ranking based, stochastic universal sampling, tournament selection or another selection scheme. In basic ABC, roulette wheel selection scheme in which each slice is proportional in size to the fitness value is employed (4):

$$P_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \tag{4}$$

In this probabilistic selection scheme, as the nectar amount of food sources (the fitness of solutions) increases, the number of onlookers visiting them increases, too.

2.4. Food source site selection by onlookers based on the information provided by employed bees

In the ABC algorithm, a random real number within the range [0, 1] is generated for each source. If the probability value (P_i in Eq. (4)) associated with that source is greater than this random number then the onlooker bee produces a modification on the position of this food source site by using Eq. (2) as in the case of the employed bee. After the source is evaluated, greedy selection is applied and the onlooker bee either memorizes the new position by forgetting the old one or keeps the old one. If solution x_i cannot be improved, its counter holding trials is incremented by 1; otherwise, the counter is reset to 0. This process is repeated until all onlookers are distributed onto food source sites.

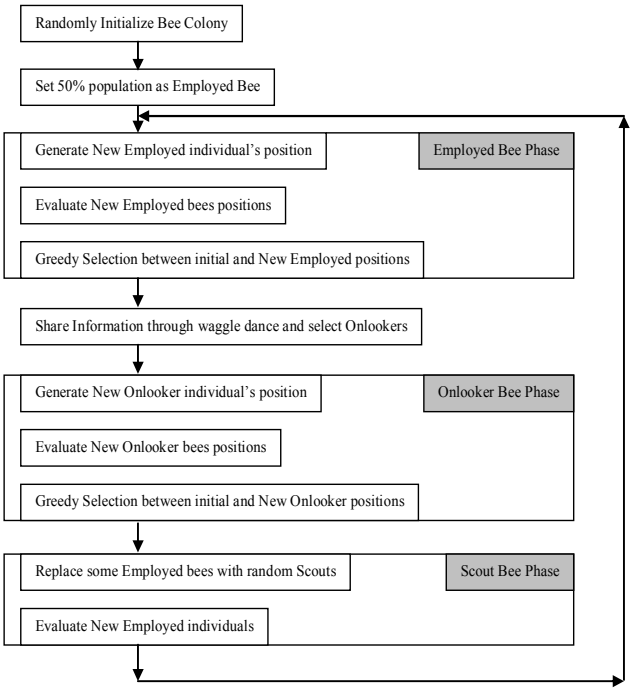


Figure 1. Pseudocode of ABC

2.5. Abandonment criteria: Limit and scout production

In a cycle, after all the employed and onlooker bees complete their searches, the algorithm checks to see if there is any exhausted source to be abandoned. In order to decide if a source is to be abandoned, the counters which have been updated during search are used. If the value of the counter is greater than the control parameter of the ABC algorithm, known as the ''*limit*", then the source associated with this counter is assumed to be exhausted and is abandoned. The food source abandoned by its bee is replaced with a new food source discovered by the scout. This is simulated by producing a site position randomly and replacing it with the abandoned one. Assume that the abandoned source is x_i, then the scout randomly discovers a new food source to be replaced with x_i. This operation can be defined as in (1). In basic ABC, it is assumed that only one source can be exhausted in each cycle, and only one employed bee can be a scout. If more than one counter exceeds the ''*limit*" value, one of the maximum ones might be chosen programmatically. The flowchart of ABC is shown in Figure 1

III. PROPOSED VARIANT: RABC

In the proposed variant of ABC we have applied three different modifications:

A. Colony Size Reduction

The task of selecting an appropriate population size for solving optimization problems remains still challenging that's also affects the convergence speed as well as the quality of the final solution. The initial colony (population) is selected uniform randomly between the lower $x^j_{\min}$ and upper $x^j_{\max}$ bounds defined for each variable x_j. Colony size *SN* seems to be an important ABC's parameter.

The population size reduction is calculated as follows [10-11]

$$x_i^{(G)} = \begin{cases} x^{(G)}_{\frac{SN}{2}+i} & if\ f(x^{(G)}_{\frac{SN}{2}+i} < f(x_i^{(G)}) \Lambda G = G_R \\ x_i^{(G)} & otherwise \end{cases} \quad (5)$$

$$SN^{(G+1)} = \begin{cases} \dfrac{SN^{(G)}}{2} & if\ G = G_R \\ SN^{(G)} & otherwise \end{cases} \quad (6)$$

$for\ \ i = 1,2,...,\dfrac{SN}{2}.$

Only a small part of the generations are exposed to reduction. Let G_R be the generation, whose population is to be reduced. The reduction is performed based on the sequence of comparisons between two individuals. One vector (individual) from the first half $x_i^{(G)}$ of the current population and a corresponding individual from the second half $x^{(G)}_{\frac{SN}{2}+i}$ are compared with regard to their fitness values and the better one is placed (as a survivor) at position *i* in the first half of the current population. The first part of the current population is the population of the next generation. The new population size is equal to half of the latest one. Equal numbers of evaluations for each colony size are performed.

B. Modifiying the perturbation scheme

The second modification in the ABC algorithm is done in the perturbation scheme (Eq. 2). The concept of controlling the frequency of perturbation was suggested in [12]. In this scheme a control parameter called modification rate (*MR*) is taken. For each parameter x_{ij}, a uniformly distributed random number, ($0 \le R_{ij} \le 1$), is produced and if the random number is less than *MR*, then the parameter x_{ij} is modified as in the Eq. (7).

$$v_{ij} = \begin{cases} x_{bestj} + r_{ij}(x_{ij} - x_{kj}), \ if \ R_{ij} < MR, \\ x_{ij}, \qquad\qquad\qquad otherwise \end{cases} \tag{7}$$

Where, $x_{best,j}$ is the individual having best fitness; $k \in \{1, 2, \ldots SN\}$ is randomly chosen index that has to be different from i ; MR is the modification rate, takes value between 0 and 1. A slight difference between our scheme and the one suggested in [12] is that we are also considering the individual having the best fitness. From the above equation we can easily see that a higher value of MR will result in making the selection process greedy in nature while a very small value will make it a bit too random in nature, slowing down the convergence. After doing a series of experiments we observed that the value of MR = 0.4 is reasonably good for the optimization problems considered in this paper.

C. Improvement in the population diversity

Selection Strategy

The basic structure of ABC follows a very simple selection strategy; a tournament is held and the individual with better fitness is selected for the next generation. This scheme though simple to apply makes the algorithm greedy in nature. The algorithm may soon lose its diversity, if only the elite individuals are considered for a next generation. In order to overcome these problems a rank selection strategy was suggested in [13-14], according to which:

$$P_k = \frac{1}{n} + a(t)\frac{n+1-2k}{n(n+1)} \qquad k = (1,2,\ldots,n)$$
$$a(t) = 0.2 + \frac{3t}{4N} \qquad t = (1,2,\ldots,N) \tag{8}$$

where $a(t)$ is a self-adaptive parameter, N is the maximum iterations. At the early stage of evolution, $a(t)$ should be lower value for keeping population diversity. At the later stage of evolution, $a(t)$ should be higher value for preventing the trend of searching stagnation due to weakened competition in the population.

The Main Steps of Modified ABC Algorithm

Step 1. Initializing each solutions of the population x_{ij}, i=1... SN, j=1…D. (using size reducing mechanism)

Step 2. Evaluate the population.

Step 3. Employed bees produce new solution v_{ij} according to Eq. (7) in the neighborhood of x_{ij}, and evaluate them.

Step 4. If fitness of v_i is better than x_i, v_i is replaced with x_i, otherwise x_i is maintained.

Step 5. Calculate the selection probability P_i for the solutions x_i according to corresponding rank selection strategy (Eq. (8)).

Step 6. Onlookers select food sources depending on P_i, and produce new solution v_i according to Eq. (7), and evaluate them.

Step 7. Go to step 4.

Step 8. Determining the abandoned solution, if exists, replacing it with a new randomly produced solution x_i for the scout according to Eq. (1).

Step 9. Memorizing the best solution achieved so far.

Step 10. If the stopping criterion is satisfied, output the best solution; otherwise, go to step 3.

IV. Parameter Settings And Test Bed

ABC has certain control parameters that have to be set by the user. Following are the parameter settings for ABC and RABC:

Parameter	Values
Colony Size	50
Dimension (D)	50, 100
Limit	200
Max. Cycle Numbers (MCN)	10,000
Max. Func. Evaluations (Max_FEs)	5.0E+05
MR (Modification Rate) [12]	0.4
Runs	25
VTR (Value to Reach)	$\lvert f_{\max} - f_{\min} \rvert < 10^{-14}$

All the algorithms have been executed on dual core processor with 1GB RAM. The programming language used is DEV C++. The random numbers are generated using inbuilt *rand* () function with same seed for every algorithm.

We have tested the ABC algorithm and RABC in two groups of functions. The first group consists of basic functions, and the second one consists of shifted functions.

a) Basic Functions:

In order to assess the performance of the ABC algorithm, we considered basic functions used in [12, 15-16] (given in Table 1). There are two groups of functions in the Table. The first group consists of unimodal functions: Sphere and Rosenbrock. The second group consists of multimodal functions: Ackley, Griewank, Weierstrass, Rastrigin, Non-continuous Rastrigin and Schwefel.

b) Shifted Function

In the second part of the experiments, we tested the ABC algorithm and RABC on the real-parameter optimization problems defined in [12, 15-16]. The benchmark functions are scalable. The dimensions of functions were 50, and 100, respectively, and 25 runs of an algorithm were needed for each function. The optimal solution results, f(x*), were known for all benchmark functions.

V. Simulation Results

The obtained results for basic functions and shifted functions [error values f(x)-f(x*)] for dimensions D=50; and D=100 are presented in Table 3.and 4 respectively. 25 runs were performed for each function. The best obtained results at

the end of each optimization process is recorded during each run. The best, mean values and standard deviations are shown in Table 3 and 4. We approximate all values below 1E-14 to 0.0.

Statistical analyses introduced by Demšar [17], García and Herrera [18], García et al. [19], and García et al. [20] were used to analyze the obtained simulation results of used algorithms. Under the null hypothesis, the k algorithms are equivalent. If the null hypothesis is rejected, then at least one of the k algorithms performed significant differently from at least one other algorithm. However, this does not indicate which one. A post-hoc test is required to be done to obtain this information.

Performance comparisons of multiple algorithms can be arranged after ranking the algorithms according to their mean value of each benchmark function. The average ranking of the algorithms to solve basic benchmark and shifted functions are given in Table 7 and 10 respectively. The best average ranking was obtained by the RABC algorithm, which outperformed the ABC algorithm.

The Convergence graphs for the basic Griekwank's and Rastrigin's benchmark functions are given in Fig. 3.

a) Comparisons with a control algorithm

The Bonferroni–Dunn test [21] is used for the post-hoc test to detect significant difference for the RABC algorithm. The performance of the RABC algorithm is significantly different if the corresponding ranks differ by at least the critical difference *CD* that is defined as follows:

$$CD = Q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

where Q_α is the critical value for a multiple non-parametric comparison with a control [22]. Four algorithms (k = 4) are used in statistical analysis, N = 8 denotes the number of functions. CD represents the threshold for the best performing algorithm, i.e., that with the highest ranking. Those rankings which do not exceed the threshold are associated with an algorithm with worse performance than the best performing algorithm.

Table 7 and 10 summarizes the ranking obtained by Friedman's test and the critical difference (CD) of Bonferroni-Dunn's procedure for basic and shifted functions respectively. Computing the p-value on each comparison. Table 5 and 8 shows the result of applying Friedman's tests in order to see whether there are global differences in the results. Given that the p-value of Friedman test is lower than the level of significance considered α = 0.05, there are significant differences among the observed results. We then compared RABC with the ABC pair wise using Wilcoxon test. The corresponding results are given in Table 6 and 9. It displays the statistics, p-value and number of +ve ranks (where control algorithm performed better than comparing algorithm), -ve ranks (where control algorithm performed worse than comparing algorithm) and tie (both algorithms performed equivalently).

Bonferroni–Dunn's graph is depicted in Fig. 2(a-d) for all problems. On each figure a horizontal cut line is drawn for two levels of significance, α = 0.05 and α = 0.10. The application of Bonferroni–Dunn's test informs us that RABC is significantly better than ABC.

VI. CONCLUSION

In the present paper a simple and easy variant of ABC called RABC for solving the numerical optimization problems having bound constraints is proposed. The proposed variant employs colony size (population size) reduction mechanism during the evolutionary process and then modification is done to control the frequency of perturbation. Further, in the variant we have used rank selection strategy to improve the population diversity as well as to control the premature convergence. The experimental results tested on eight benchmark functions and six shifted functions show that RABC algorithm with appropriate parameter outperforms ABC algorithm.

REFERENCES

[1] E. Bonabeau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, Inc., New York, NY, USA, 1999.

[2] I. Kassabalidis, M.A. El-Sharkawi, R.J. II Marks, P. Arabshahi, A.A. Gray, Swarm intelligence for routing in communication networks, Global Telecommunications Conference, GLOBECOM '01, 6, IEEE, 2001, pp. 3613–3617.

[3] D. Karaboga, An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[4] B. Basturk, D. Karaboga, An artificial bee colony (abc) algorithm for numeric function optimization, IEEE Swarm Intelligence Symposium 2006 (Indianapolis, Indiana, USA), May 2006.

[5] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, Journal of Global Optimization 39 (3) (2007) 459–471.

[6] D. Karaboga, B. Akay,, Solving large scale numerical problems using artificial bee colony algorithm, in: Sixth International Symposium on Intelligent and Manufacturing Systems Features, Strategies and Innovation (Sakarya, Turkiye), October 14–17, 2008.

[7] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Applied Soft Computing 8 (1) (2008) 687–697.

[8] D. Karaboga, B. Akay, Effect of region scaling on the initialization of particle swarm optimization differential evolution and artificial bee colony algorithms on multimodal high dimensional problems, in: International Conference on Multivariate Statistical Modelling and High Dimensional Data Mining (Kayseri, TURKEY), June 19–23, 2008.

[9] T.D. Seeley, The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies, Harvard University Press, 1995.

[10] Brest J, Mauc˘ec MS (2008) Population size reduction for the ifferential evolution algorithm. Appl Intell 29(3):228–247.

[11] Brest J, Zamuda A, Bos˘kovic´ B, Mauc˘ec MS, Z ˇ umer V (2008) Highdimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In: 2008 IEEE World Congress on computational intelligence, IEEE Press, pp 2032–2039.

[12] B. Akay, D. Karaboga, A modified Artificial Bee Colony algorithm for real-parameter optimization, Inform.Sci. (2010), doi:10.1016/j.ins.2010.07.015

[13] Li Bao and Jian-chao Zeng , "Comparison and Analysis of the Selection Mechanism in the Artificial Bee Colony Algorithm", In Proc. of Ninth International Conference on Hybrid Intelligent Systems, pp. 411-416, 2009.

[14] A.G. SONG, J.R. LUO, "A Ranking Based Adaptive Evolutionary Operator Genetic Algorithm", Acta Electronica Sinica, 1999, vol.27 no.1, pp.85-88.

[15] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, 2005.

[16] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281−295.

[17] Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30.

[18] García S, Herrera F (2008) An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J Mach Learn Res 9:2677–2694.

[19] García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 Special Session on Real Parameter Optimization. Journal of Heuristic 15(6):617–644.

[20] García S, Ferna´ndez A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Info Sci 180(10):2044–2064.

[21] Dunn OJ (1961). Multiple comparisons among means. J Am Stat Assoc 56(293):52–64.

[22] Zar JH (1999) Biostatistical analysis. Prentice-Hall, Englewood Cliffs.

TABLE I. BASIC UNIMODAL AND MULTIMODAL TEST FUNCTIONS

Function	Name	Definition	Range	$f(x^*)$
f_1	Sphere	$\sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
f_2	Rosenbrock's	$\sum_{I=1}^{D-1}(100(x_i^2 + x_{i+1})^2 + (x_i - 1)^2)$	$[-2.048, 2.048]^D$	0
f_3	Ackley's	$-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$	$[-32, 32]^D$	0
f_4	Griekwank's	$\sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	0
f_5	Weierstrass	$\sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos(2\pi b^k(x_i + 0.5))\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos(2\pi b^k 0.5)\right]$, $a = 0.5, b = 3, k\max = 20$	$[-0.5, 0.5]^D$	0
f_6	Rastrigin's	$\sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^D$	0
f_7	Noncontinuous Rastrigin	$\sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10),\quad y_i = \begin{cases} x_i & \lvert x_i\rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert \geq \frac{1}{2} \end{cases}$	$[-5.12, 5.12]^D$	0
f_8	Schwefel	$4.189829xD - \sum_{i=1}^{D} -x_i\sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]^D$	0

TABLE II. SHIFTED FUNCTIONS F_1 - F_6

Function	Name	Definition	Range	Fitness Optimum
F_1	Sphere	$\sum_{i=1}^{D} x_i^2 + f\_bias, x = z - o$	$[-100, 100]^D$	-450
F_2	Schwefel Problem 2.21	$\max_i\{\lvert x_i\rvert, 1 \le i \le D\} + f\_bias, x = z - o$	$[-100, 100]^D$	-450
F_3	Rosenbrock's	$\sum_{I=1}^{D-1}(100(x_i^2 + x_{i+1})^2 + (x_i - 1)^2) + f\_bias, x = z - o$	$[-100, 100]^D$	390
F_4	Rastrigin's	$\sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10) + f\_bias, x = z - o$	$[-5, 5]^D$	-330
F_5	Griekwank's	$\sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 + f\_bias, x = z - o$	$[-600, 600]^D$	-180
F_6	Ackley's	$-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e + f\_bias$	$[-32, 32]^D$	-140

TABLE III. EXPERIMENTAL RESULTS BASIC FUNCTIONS

Algorithm	Statistics	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
					D = 50 and SN = 50				
ABC	*Best*	6.43E-016	3.08E+001	8.22E-012	4.44E-012	6.86E-013	3.80E-013	1.42E-011	4.02E+001
	Mean	7.10E-17	4.12E+001	7.05E-011	3.19E-011	9.27E-012	4.28E-012	7.19E-011	1.01E+002
	Std. Dev	± 7.59E-017	± 1.15E+001	± 4.82E-012	± 2.22E-011	± 7.70E-013	± 1.19E-012	± 5.06E-011	± 7.17E+001
RABC	*Best*	1.72E-017	7.12E+000	9.07E-013	4.81E-013	2.15E-013	1.26E-013	7.18E-012	4.71E+000
	Mean	3..08E-016	2.35E+001	7.72E-012	4.19E-012	1.98E-012	2.47E-012	3.10E-011	8.01E+001
	Std. Dev	±7.83E-016	±1.01E+001	±5.68E-012	±2.72E-011	±1.65E-012	±1.10E-012	±2.71E-011	±3.72E+001
					D = 100 and SN = 50				
ABC	*Best*	6.03E-010	7.12E+001	2.19E-005	7.01E-005	4.31E-004	1.14E-006	0.91E-007	8.43E+002
	Mean	5.53E-008	1.45E+002	3.08E-004	1.36E-004	8.39E-003	6.26E-005	1.24E-005	2.00E+003
	Std. Dev	±4.56E-008	±3.01E+001	± 1.98E-004	±7.27E-004	±1.71E-003	±2.44E-005	±3.48E-005	±3.28E+002
RABC	*Best*	5.91E-011	0.00E+000	2.13E-006	1.19E-007	3.84E-006	1.60E-007	4.18E-008	3.97E+001
	Mean	4.97E-010	2.62E+001	4.79E-005	2.90E-005	9.37E-005	4.92E-007	5.63E-007	1.57E+002
	Std. Dev	± 2.31E-010	± 1.61E+000	± 6.59E-005	± 3.18E-005	± 3.11E-005	± 2.89E-007	± 2.45E-007	± 1.13E+002

TABLE IV. EXPERIMENTAL RESULTS SHIFTED FUNCTIONS

Algorithm	Statistics	F_1	F_2	F_3	F_4	F_5	F_6
				D=50 and SN=50			
ABC	*Best*	0.00E+00	4.91E-03	7.12E+01	0.00E+00	0.00E+00	5.12E-14
	Mean	0.00E+00	4.13E-02	2.36E+01	0.00E+00	0.00E+00	7.45E-14
	Std. Dev	±0.00E+00	±2.72E-02	±1.87E+01	±0.00E+00	±0.00E+00	±2.91E-14
RABC	*Best*	0.00E+00	1.87E-04	4.91E-01	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	2.76E-03	9.91E+01	0.00E+00	0.00E+00	0.00E+00
	Std. Dev	±0.00E+00	±1.72E-03	±2.12E+00	±0.00E+00	±0.00E+00	±0.00E+00
				D=100 and SN=50			
ABC	*Best*	0.00E+00	9.31E-02	5.33E+01	0.00E+00	0.00E+00	1.41E-13
	Mean	0.00E+00	1.21E-01	6.13E+01	0.00E+00	0.00E+00	2.03E-13
	Std. Dev	±0.00E+00	±3.80E-01	±1.65E+01	±0.00E+00	±0.00E+00	±3.18E-14
RABC	*Best*	0.00E+00	9.19E-03	1.21E+01	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	2.81E-02	3.21E+01	0.00E+00	0.00E+00	0.00E+00
	Std. Dev	±0.00E+00	±1.26E-02	±1.87E+01	±0.00E+00	±0.00E+00	±0.00E+00

TABLE V. RESULTS OF FRIEDMAN TEST BASED ON ERROR FOR BASIC FUNCTIONS

D & SN	N	df	*p*-value
50 & 50	8	1	<0.001
100 & 50	8	1	
df – Degrees of Freedom		N - Total No of functions	

TABLE VI. RESULTS OF PAIR WISE COMPARISON BASED ON ERROR

D & SN	RABC Vs.	Wilcoxon Test			
		+ve	-ve	tie	p-value
50 & 50	ABC	7	1	0	.034
100 & 50	ABC	8	0	0	.005

TABLE VII. RANKING OBTAINED THROUGH FRIEDMAN'S TEST AND CRITICAL DIFFERENCE (CD) CALCULATED THROUGH BONNFERRONI-DUNN'S PROCEDURE

Algorithm	Mean Rank (D=50 &SN=50)	Mean Rank (D=100 & SN=50)
RABC	1.12	1.06
ABC	2.96	3.23
CD for α = 0 .05	**1.265**	
CD for α = 0 .10	**1.062**	

TABLE VIII. RESULTS OF FRIEDMAN TEST BASED ON ERROR FOR SHIFTED FUNCTIONS

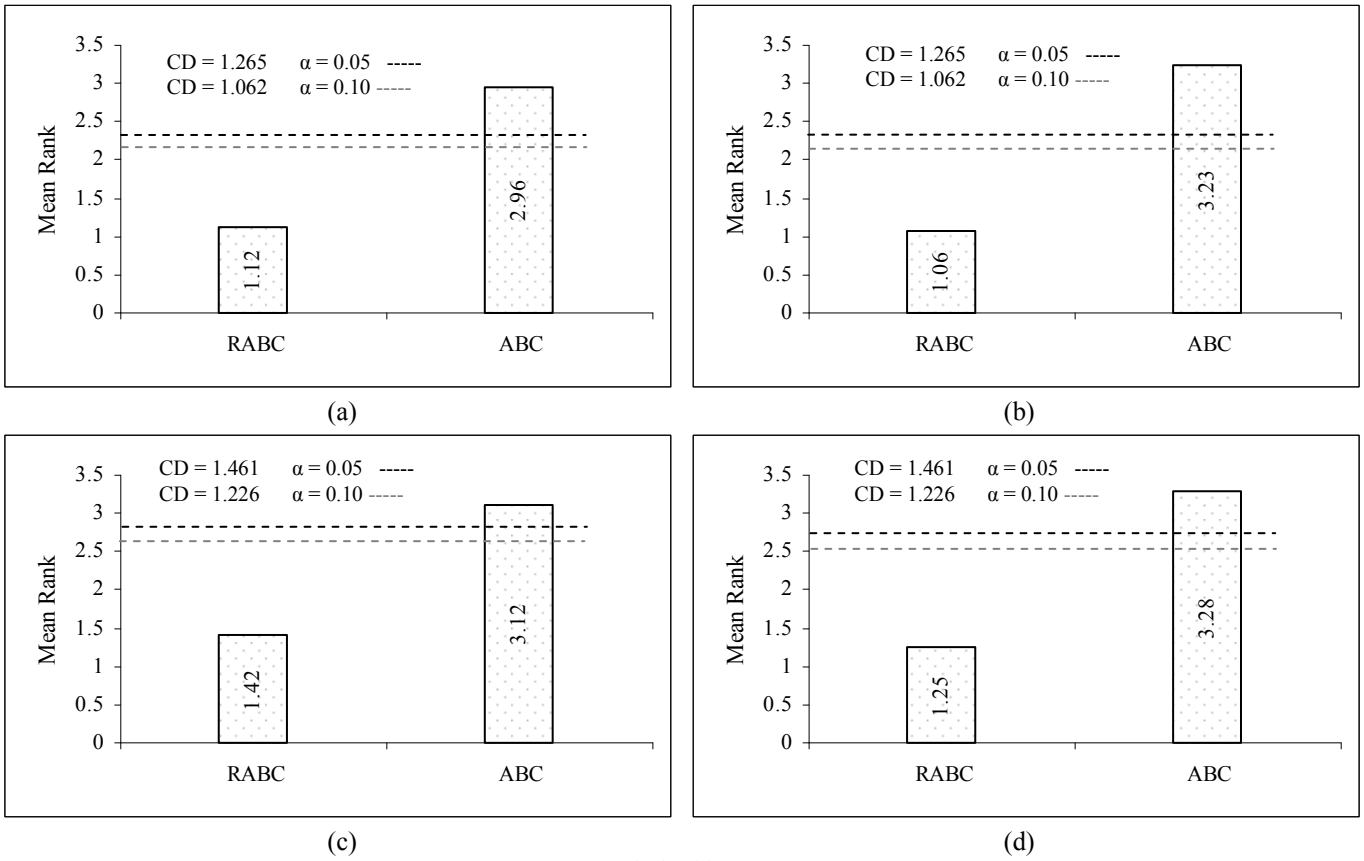
D & SN	N	df	*p*-value
50 & 50	6	1	<0.001
100 & 50	6	1	
df – Degrees of Freedom		N - Total No of functions	

TABLE IX. RESULTS OF PAIR WISE COMPARISON BASED ON ERROR

D & SN	RABC Vs.	Wilcoxon Test			
		+ve	-ve	tie	p-value
50 & 50	ABC	2	1	3	.047
100 & 50	ABC	3	0	3	.019

TABLE X. RANKING OBTAINED THROUGH FRIEDMAN'S TEST AND CRITICAL DIFFERENCE (CD) CALCULATED THROUGH BONNFERRONI-DUNN'S PROCEDURE

Algorithm	Mean Rank (D=50 &SN=50)	Mean Rank (D=100 & SN=50)
RBC	1.42	1.25
ABC	3.12	3.28
CD for α = 0 .05	**1.461**	
CD for α = 0 .10	**1.226**	



Control Algorithm : RABC

Figure 2. Bonferroni-Dunn's graphic corresponding to error for basic functions(a) D=50 &SN=50 (b) D=100 &SN=50 and for shifted functions (c) D=50 &SN=50 (d) D=100 &SN=50

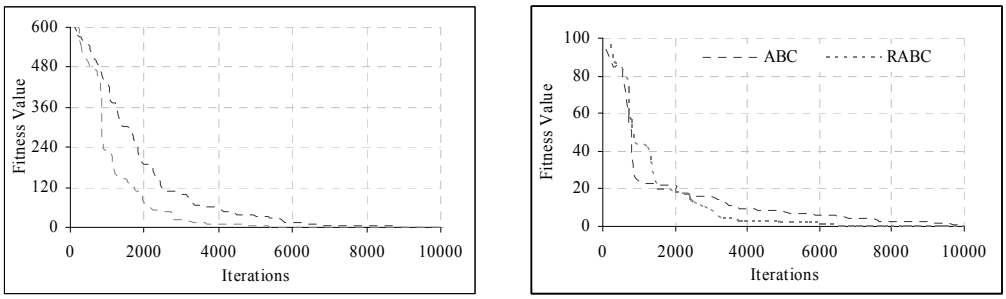


Figure 3. Convergence graphs of (a) f_4 and (b) f_6 test functions