IOSR Journal of Computer Engineering (IOSRJCE)
ISSN: 2278-0661 Volume 4, Issue 1 (Sep-Oct. 2012), PP 06-13
www.iosrjournals.org

An efficient and powerful advanced algorithm for solving real coded numerical optimization problem: artificial bee colony algorithm with crossover operators

Mrs. Shahana Gajala Qureshi[1], Mrs. Uzma Arshi Ansari[2]
[1,2]*(Computer Science & Engg., RIT/ CSVTU, Bhilai, Chhattisgarh, India)*

Abstract:*Artificial Bee Colony (ABC) algorithm is inhabitants -based flock intelligence algorithm. There are many algorithms Present for solving numeric optimization problem.ABC is based on the intellectual behaviour of honey bee flock. In this work, ABC algorithm is used with Genetic crossover machinist and tested on standard benchmark functions, and also result compared with the X-ABC algorithm with the plus of fewer organize parameters. Obtained results show that the concert of the ABC with crossover gives better results than the X-ABC algorithm.*

Keywords: *Artificial bee colony, Crossover operator, Evolutionary algorithm, Genetic algorithm, Numerical function optimization, Swarm Intelligence.*

I. INTRODUCTION

Several optimization algorithms have been proposed to choose the best component from some set of available alternatives. Based on intelligent behaviour of honey bee flock there is only one numerical optimization algorithm in the literature [1]. Flock develops collective intelligence. In 2005 D. Karaboga has described an artificial bee colony (ABC) algorithm [2] for optimizing multivariable functions.

Several inhabitants based algorithm has been proposed to find near-optimal solutions to the difficult optimization problems like: scheduling and routing problems. A inhabitants -based algorithm refers the inhabitants consisting of possible solutions to the problem are modified by applying some operators on the solutions depending on the information of their fitness values. Inhabitants based algorithm is classified into two groups: evolutionary algorithms and swarm intelligence-based algorithms [3]. Evolutionary algorithm [4] is one of the most accepted genetic algorithm (GA) based on the natural evolution. In the basic GA, a selection operation is applied to the solutions evaluated by the evaluation component.

Linear, Blend, Unfair average, Laplace real coded crossover operators is applied to ABC to check whether the algorithm works better on the Benchmark function or not as compare to X-ABC algorithm. In this paper we performed an experiment on advance model of ABC with crossover operations. In this work crossover steps are added to the standard ABC. The crossover operations are performed between each individual food source best positions. After the crossover operation, the fitness of the individual food source best position is compared with that of the two off-springs, and the best one is taken as the new individual food source best position. Two food sources are selected as parents through selection process and calculate their fitness values. After selecting a crossover points randomly, new fitness values are generated using crossovers probabilities. Both the above techniques perform a crossover by swapping the food source around the crossover points.

The rest of the paper is organized as follows: Genetic algorithm (GA) in second section. Crossover operator in section third. In section fourth praposed work. Algorithm and Flowchart .Benchmark functions are described in section fifth. Parameter setup in section sixth followed by Experimental Results in seventh, conclusion in section eight and future scope in Section ninth.

II. GENETIC ALGORITHM

In the computer science field of artificial intelligence, a genetic algorithm (GA**)** is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Basically it consists of five components: a random number generator, a fitness evaluation unit and genetic operators for reproduction; crossover and mutation operations. The basic algorithm of GA is as below:

```
1: Initialize Population
2: repeat
3: Evaluation
4: Reproduction
```

5: Crossover
6: Mutation
7: until requirements are met

The initial population required at the start of the algorithm, is a set of food source generated by the random generator. Each position of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. A fitness value is a measure of the goodness of the solution that it represents. Essentially the aim of the genetic operators is to transform this set of food source into sets with superior fitness values. The reproduction operator performs a natural selection function known as seeded selection. Individual food source are copied from one set (representing a generation of solutions) to the next according to their fitness values, the superior the fitness value, the greater the probability of a food source being selected for the next iteration. The crossover operator chooses pairs of food sources at random and produces new pairs. The simplest crossover operation is to cut the original food sources nectar amount at a randomly selected point and to exchange. The number of crossover operations is governed by a crossover rate. The mutation operator randomly mutates or reverses the values of food source. The number of mutation operations is determined by a mutation rate. A phase of the algorithm consists of applying the evaluation, reproduction, crossover and mutation operations. A new generation of solutions is produced with each phase of the algorithm [5].

III. CROSSOVER OPERATOR

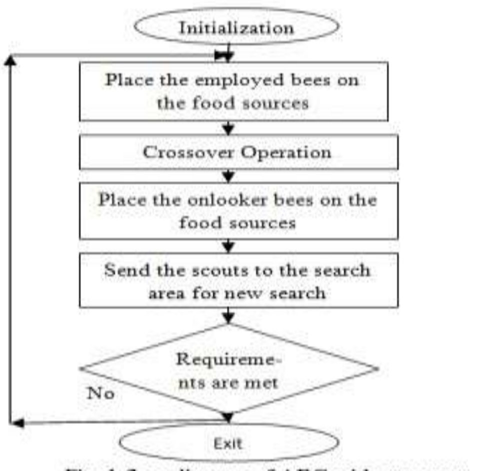
In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Crossover is a process of taking more than one parent solutions and producing a child solution from them, with the possibility that high-quality chromosomes may generate better ones. The crossover operator is not generally applied to all pairs of chromosomes in the intermediate inhabitants. A random option is made, where the probability of crossover being applied depends on probability defined by a crossover rate, the crossover probability. The crossover operator plays a vital responsibility in GAs. It combines parts of fine solution to form new probable result. Information contained in one solution combine with information contained in another solution and the resulting solution will either have excellent fitness or survive to exchange this information again. Crossover operators exist for both real coded and binary coded GA. Since this paper explores the application of crossover operator to ABC for numerical optimization problem therefore for this paper we discuss only real coded crossover operator. A number of real coded crossovers have been introduced for GA and other heuristic technique.

IV. PRAPOSED ARTIFICIAL BEE COLONY (ABC) ALGORITHM WITH CROSSOVER OPERATORS

ABC generally suffers from premature convergence, tending to get stuck in local optima, low solution precision and so on. In order to defeat these shortcomings and acquire better results, various improvements to ABC have been proposed. One of the approaches ABC with crossover operator is proposed by adding a crossover step to the standard ABC to acquire improved outcome.

In this paper ABC with linear, Blend, Unfair average and Laplace crossovers are applied to the 3 benchmark functions, which gives the enhanced optimum result at enhanced probability as compare to X-ABC algorithm. Food sources generated by ABC are randomly selected for crossover operation and two new offspring's are formed. The finest offspring (in terms of fitness value) selected from the new offspring's. This new finest offspring replaces the worst fitness value which is selected for crossover. The replacement is done if the new finest offspring has the excellent fitness value than the parent food source. Algorithm of proposed method is as follows:

• INITIALIZE.
• REPEAT.
(a) Place the employed bees on the food sources in the memory;
(b) Crossover Operator;
(c) Place the onlooker bees on the food sources in the memory;
(d) Send the scouts to the search area for discovering new food sources.
• UNTIL (requirements are met).


Initialization
Place the employed bees on the food sources
Crossover Operation
Place the onlooker bees on the food sources
Send the scouts to the search area for new search
Requirements are met
No
Exit
Fig. 1 flow diagram of ABC with crossover

V. BENCHMARK FUNCTION

In this paper 3 standard set of Benchmarks functions [6] are chosen to test and compare the performance of both ABC with crossovers and X-ABC. All the benchmarks chosen have multivariable functions and have different extent of complexity. This set includes greatly unusual kind of problems like: multimodal, local minima in order to approximate, global optimum, and dimensionality.

1.1 Griewank Function

The Griewank function comprehensively used to experiment the convergence of optimization functions. The Griewank function of order n is distinct by:

$$f_1(\overline{x}) = 1/400\left(\sum_{i=1}^{D} (x^2_i)\right) - \left(\prod_{i=1}^{D} \cos(x_i/\sqrt{i})\right) + 1 \qquad (1)$$

Its value is 0 at its global minimum (0,0,…,0) and local minima at $\underline{+}\, x$. Initialization range for the function is [−600,600]. It has a global minimum of 0 at the point x=0.

1.2 Rosenbrock Function

Rosenbrock function is frequently used as a test problem for optimization algorithms.

$$f_2(\overline{x}) = \sum_{i=1}^{D} 100(x_i^2 - x_{i+1})^2 + (1-x_i)^2 \qquad (2)$$

It has a global minimum at (x, y) = (1, 1) where (x, y) = 0. Initialization range for the function is [−15, 15].

1.3 Sphere Function

It is the first function of De Jong's test set. Function is continuous, convex and unimodal. It has the following general definition:

$$f_3(x) = \sum_{i=1}^{n} x^2_i \qquad (3)$$

Initialization range for the function is $-5.12 <= x_i <= 5.12$, i = 1,…, n. Global minimum f(x) = 0 is for $x_i = 0$, i = 1,…, n.

VI. PARAMETER SETUP FOR EXPERIMENT

In this paper the parameter setting are as colony size (NP) = 20, 40, 80, Maximum number of cycles (MCN) = 1000, 2000, 3000 respectively for the dimension 10. Food sources equal the half of the colony size. The percentage of onlooker bees was 50% of the colony, the employed bees were 50% of the colony and the number of scout bees was selected as one. X_{max} and X_{min} are the upper and lower bounds of the decision variables. X_{min} (lb) and X_{max} (ub) are taken in the range -100 to 100. This paper presents an experiment on X-ABC and Advanced ABC with Crossover operators. For Crossover, Linear, Blend, Unfair average and Laplace crossovers are chosen. For experiment, crossover's probabilities are taken in the range of 0.1 to 0.9.

Table 1: X-ABC vs. ABC with crossover operator indicate the results that obtained after 1000 cycles with a population 20 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with linear crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < . 8	pr < .9
Griewank	1.332268	1.221245	**1.554312**	1.221245	1.44329	**9.991007**	1.332268	1.332268	**9.992007**	**9.992007**
Griewank	8.881784	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	8.881784
Griewank	9.992007	7.771561	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007

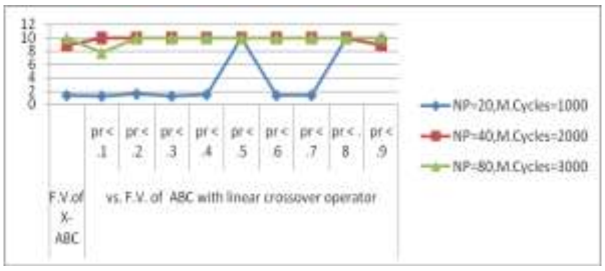


Fig .2 Fitness value of X-ABC vs. ABC with linear crossover operator with Griewank Function

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with linear crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < . 8	pr < .9
Rosenbrock	8.881784	**9.992476**	**9.502416**	**9.123774**	**9.409717**	**9.004217**	**9.715304**	**9.428769**	**9.480702**	**9.0259**
Rosenbrock	9.109312	**9.676996**	**9.899817**	**9.944221**	**9.620697**	**9.580132**	9.105082	8.607993	**9.918184**	**9.264489**
Rosenbrock	4.737206	**4.976909**	**4.80848**	**5.359311**	**5.41167**	**5.337368**	**4.799651**	**5.387644**	**5.846728**	**5.574773**

Table 2: X-ABC vs. ABC with crossover operator indicate the results that obtained after 2000 cycles with a population 40 for the dimension 10

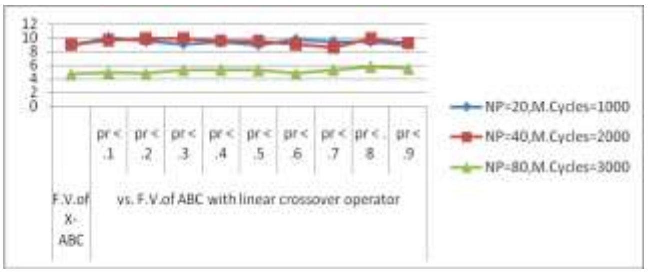


Fig .3 Fitness value of X-ABC vs. ABC with linear crossover operator with Rosenbrock Function

Table 3: X-ABC vs. ABC with crossover operator indicate the results that obtained after 3000 cycles with a population 80 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with linear crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < . 8	pr < .9
Sphere	1.724582	**1.975405**	**1.762315**	**1.868645**	**1.769938**	**1.831868**	**1.982908**	**1.909434**	**1.727958**	**1.761739**
Sphere	8.16916	**8.519625**	**8.777245**	**8.774533**	**8.561635**	**8.539031**	**8.274195**	**8.498855**	**8.191727**	**8.614061**
Sphere	6.350367	**6.769752**	**6.887722**	**6.818324**	**6.841564**	**6.728867**	**6.515446**	**6.90444**	**7.125173**	**7.24512**

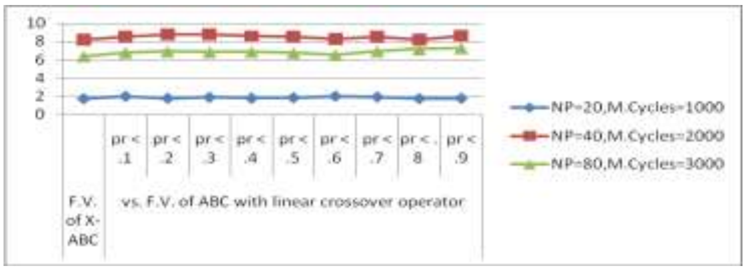


Fig.4 Fitness value of X-ABC vs. ABC with linear crossover operator with Sphere Function

Table 4: X-ABC vs. ABC with crossover operator indicate the results that obtained after 1000 cycles with a population 20 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with blend crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < .8	pr < .9
Griewank	1.332268	1.332268	1.332268	1.221245	**9.992007**	1.332268	**1.44329**	**1.554312**	**2.109424**	**1.776357**
Griewank	8.881784	**9.992007**	8.881784	8.881784	**9.992007**	8.881784	**9.992007**	8.881784	8.881784	**9.992007**
Griewank	9.992007	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	8.821784	**9.992007**	**9.992007**	**9.992007**

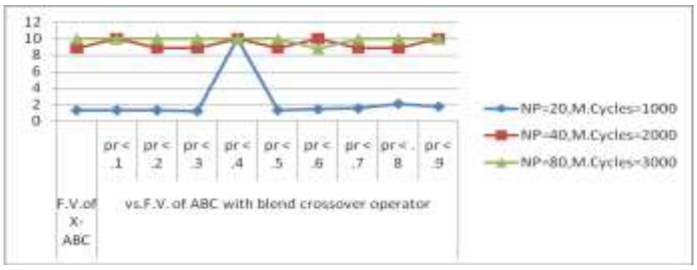


Fig.5 Fitness value of X-ABC vs. ABC with blend crossover operator with Griewank Function

Table 5: X-ABC vs. ABC with crossover operator indicate the results that obtained after 2000 cycles with a population 40 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with blend crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < .8	pr < .9
Rosenbrock	8.881784	**9.042882**	**9.806536**	**9.712594**	**9.581151**	**9.146515**	**9.34116**	**9.828222**	**9.750363**	**9.50469**
Rosenbrock	9.109312	**9.726826**	**9.159107**	**9.508287**	**9.414671**	**9.184223**	**9.824967**	**9.121683**	**9.70832**	**9.741905**
Rosenbrock	4.737206	**6.225532**	**6.085718**	**7.731131**	**5.919354**	**7.132709**	**6.137986**	**5.169167**	**5.668729**	**4.890934**

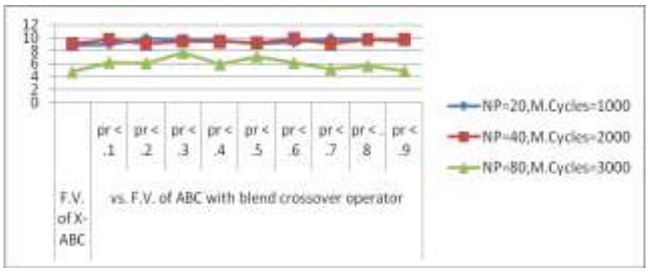


Fig.6 Fitness value of X-ABC vs. ABC with blend crossover operator with Rosenbrock Function

Table 6: X-ABC vs. ABC with crossover operator indicate the results that obtained after 3000 cycles with a population 80 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with blend crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < .8	pr < .9
Sphere	1.724582	**1.857113**	**1.990003**	**2.257062**	**2.478882**	**2.16562**	**1.753149**	**2.108819**	**2.057777**	**2.33518**
Sphere	8.16916	**8.493092**	**8.250463**	**9.560418**	**8.925584**	**8.228357**	7.632805	**8.577423**	7.811535	**8.151418**
Sphere	6.350367	**7.395156**	**7.18444**	**6.73418**	**7.324988**	**6.790247**	**6.989725**	**7.082038**	**7.229607**	**7.230075**

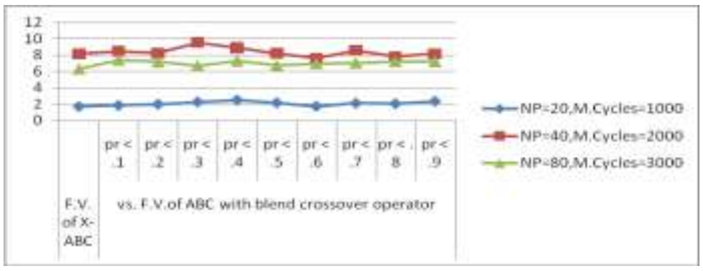


Fig.7 Fitness value of X-ABC vs. ABC with blend crossover operator with Sphere Function

Table 7: X-ABC vs. ABC with crossover operator indicate the results that obtained after 1000 cycles with a population 20 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with unfair average crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < .8	pr < .9
Griewank	1.332268	**1.553196**	**1.471672**	**1.579627**	**1.551399**	**1.698498**	**1.439733**	**1.483601**	**1.377626**	**1.547081**
Griewank	8.881784	**9.044839**	**9.00549**	**8.900546**	**9.059013**	**9.171036**	**9.520834**	**8.962556**	**9.096149**	**9.16431**
Griewank	9.992007	8.881784	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007

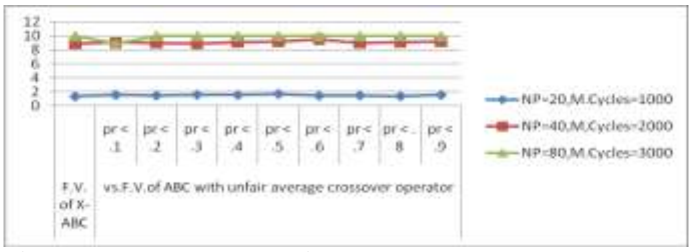


Fig.8 Fitness value of X-ABC vs. ABC with unfair average crossover operator with Griewank Function

Table 8: X-ABC vs. ABC with crossover operator indicate the results that obtained after 2000 cycles with a population 40 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with unfair average crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < .8	pr < .9
Rosenbrock	8.881784	**9.668708**	**9.369829**	**9.100729**	**9.364081**	**9.89934**	**9.427984**	**9.444507**	**8.993249**	**9.820548**
Rosenbrock	9.109312	**9.280388**	**9.757437**	**9.522515**	**9.774416**	**9.603424**	**9.159101**	9.021725	**9.909613**	**9.368554**
Rosenbrock	4.737206	**4.999622**	**5.988008**	**6.85281**	**5.176618**	**5.42835**	**5.71288**	**5.107928**	**6.342836**	**5.708231**

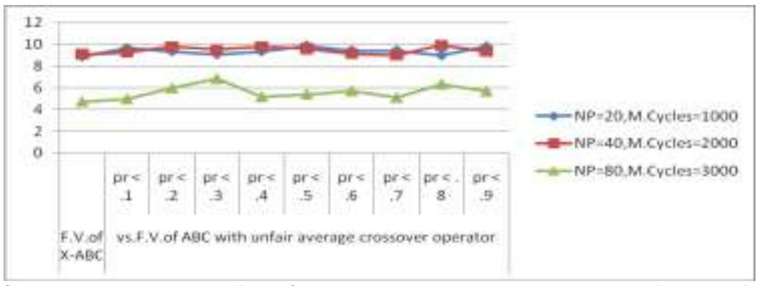


Fig.9 Fitness value of X-ABC vs. ABC with unfair average crossover operator with Rosenbrock Function

Table 9: X-ABC vs. ABC with cros sover operator indicate the results that obtained after 3000 cycles with a population 80 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with unfair average crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < .8	pr < .9
Sphere	1.724582	**1.82716**	**2.168752**	**1.818904**	**1.833731**	**1.767593**	**1.757495**	**1.794837**	**1.738484**	**1.73468**
Sphere	8.16916	**9.191849**	**8.267468**	**9.811755**	**8.989315**	**9.077142**	**8.952292**	**9.140383**	**9.055519**	**8.183684**
Sphere	6.350367	**6.979073**	**7.079201**	**7.557484**	**7.454903**	**7.263444**	**7.202288**	**6.734063**	**6.655374**	**7.59731**

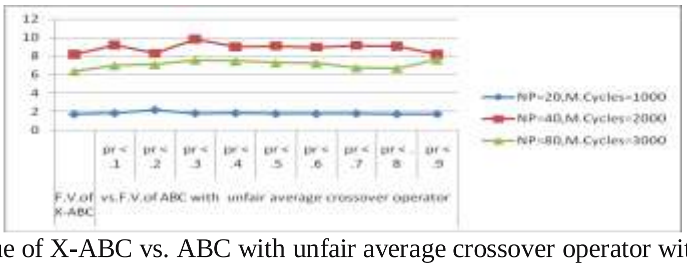


Fig.10 Fitness value of X-ABC vs. ABC with unfair average crossover operator with Sphere Function

Table 10: X-ABC vs. ABC with crossover operator indicate the results that obtained after 1000 cycles with a population 20 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with Laplace crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < . 8	pr < .9
Griewank	1.332268	**1.44329**	1.332268	1.332268	1.332268	1.332268	1.332268	1.332268	**1.44329**	**1.887379**
Griewank	8.881784	**9.992001**	8.881784	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**	**9.992007**
Griewank	9.992007	9.992007	8.881784	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007	9.992007

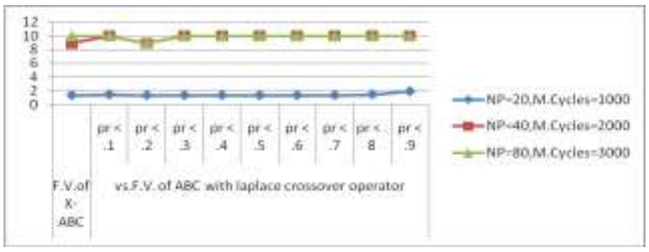


Fig.11 Fitness value of X-ABC vs. ABC with Laplace crossover operator with Griewank Function

Table 11: X-ABC vs. ABC with crossover operator indicate the results that obtained after 2000 cycles with a population 40 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with Laplace crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < . 8	pr < .9
Rosenbrock	8.881784	**9.056589**	**8.952119**	**9.177026**	**9.362058**	**9.034302**	8.863433	**9.765404**	**9.082154**	**9.623144**
Rosenbrock	9.109312	**9.986449**	**9.761061**	**9.757753**	**9.539391**	**9.166396**	**9.754584**	8.841904	**9.518827**	9.018886
Rosenbrock	4.737206	**5.206836**	**6.207948**	**7.769313**	**5.518227**	**8.153793**	**9.10503**	**7.730469**	**9.064648**	**7.836019**

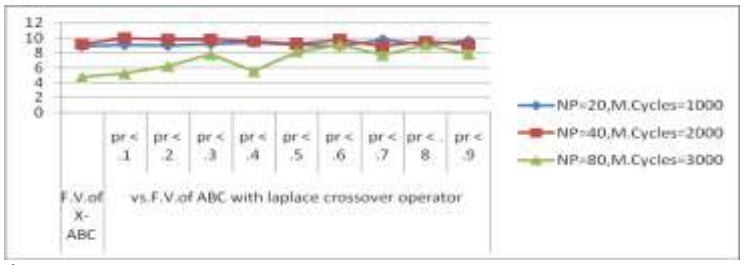


Fig.12 Fitness value of X-ABC vs. ABC with Laplace crossover operator with Rosenbrock Function

Table 12: X-ABC vs. ABC with crossover operator indicate the results that obtained after 3000 cycles with a population 80 for the dimension 10

Benchmark Function	Fitness value of X-ABC	Fitness value of ABC with Laplace crossover operator								
		pr < .1	pr < .2	pr < .3	pr < .4	pr < .5	pr < .6	pr < .7	pr < . 8	pr < .9
Sphere	1.724582	**2.099992**	**1.745922**	**2.182766**	**2.270631**	**2.743091**	**2.057093**	**2.688745**	**2.566057**	**2.375882**
Sphere	8.16916	**8.523161**	**8.81656**	**8.634238**	**9.45504**	**9.819334**	**8.893172**	**9.286061**	**9.258724**	**8.517896**
Sphere	6.350367	**6.960124**	**6.776648**	**6.764065**	**6.526025**	**7.127448**	**6.996966**	**7.201134**	**7.428873**	**6.760549**

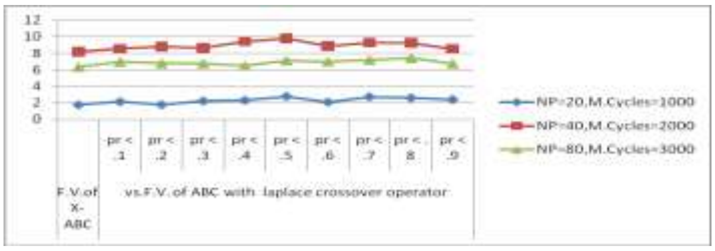


Fig.13 Fitness value of X-ABC vs. ABC with Laplace crossover operator with Sphere Function

VII. EXPERIMENTAL RESULT

In Experiments we compared the X-ABC and ABC with crossover operator algorithms on a set of 3 benchmark problems for each crossover operators given in above Tables. The X-ABC and ABC with crossovers are implemented in C++ language as a front end. Recorded simulated results are presented in above Tables for each benchmark problem. In this experiment mean fitness value is calculated for both the algorithms X-ABC and Advanced ABC with each 3 benchmark problems. In the above tables the bold readings are enhanced result over the ABC with crossovers. Obtained results indicated in tables that ABC with crossover provides better results than original ABC algorithm. Here we also observed that Sphere function provides the better improved result than the Griewank and Rosenbrock.

VIII. CONCLUSION

In this paper, X-ABC is compared with advanced ABC algorithm. Advanced ABC is obtained by an supplementary step added to X-ABC that is a crossover method of GA. Code of X-ABC and Advanced ABC is developed in C++ language and results are shown in above Tables. The experiments are performed on a set of 3 benchmark problems presented in the literature. The ranges of crossovers probability are set to 0.1 to 0.9. For each benchmark problems results are tabulated for crossovers probability 0.1 to 0.9 in terms of fitness value which describes the efficiency and accuracy are depicted in bold. From the result obtained it is concluded that Advanced ABC has been found to have victorious performance on Griewank, Rosenbrock, and Sphere benchmark problems.

IX. FUTURE SCOPE

In this experiment we compared X-ABC and Advanced ABC with Linear, Blend, Unfair average and Laplace crossover operator's algorithms on a set of 3 benchmark problems and concluded that advanced algorithm gives the enhanced result than X-ABC algorithm. In the future work X-ABC is compared with Advanced ABC with 3 more crossover operators of GA naive, simulated and uniform on a set of 4 benchmark problems Griewank, Rosenbrock, Rastrigin and Sphere .

REFERENCES

[1] Yang, X.S.: Engineering optimizations via *nature- inspired virtual bee algorithms, pp. 317–323. Springer, GmbH (2005).*

[2] Karaboga, D.: *An idea based on honey bee swarm for numerical optimization*. Technical Report- TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[3] R.C. Eberhart, Y. Shi, J. Kennedy, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[4] A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.

[5] D.T. Pham, D. Karaboga, *Optimum design of fuzzy logic controllers using genetic algorithms*, Journal of Systems Engineering 1 (1991) *114-118.*

[6] Srinivasan, D., Seow, T.H.: *Evolutionary Computation*, CEC '03, 8–12 Dec. 2003, 4(2003), Canberra, Australia, *pp. 2292–2297.*

[7] D. Karaboga, B. Akay," *Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization*", Erciyes University, the Dept. of Computer Engineering, 38039, Melikgazi, Kayseri, Turkiye.

[8] Dervis KARABOGA,"*An idea based on honey bee swarm for numerical optimization*", Erciyes University, Engineering Faculty Computer Engineering Department, Kayseri/Türkiye, TECHNICAL REPORT-TR06, OCTOBER, 2005.

[9] Adil Baykaso lu, Lale Özbakır and Pınar Tapkan," *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*", University of Gaziantep, Department of Industrial Engineering 2Erciyes University, Department of Industrial Engineering Turkey.

[10] Adil Baykaso lu, Lale Özbakır and Pınar Tapkan," *Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem*", University of Gaziantep, Department of Industrial Engineering 2Erciyes University, Department of Industrial Engineering Turkey.

[11] Teodorovi´c, D.: Transport Modeling By Multi-Agent Systems: *A Swarm Intelligence Approach*, Transport. Plan. Technol. **26**(4), *289–312* (2003).

[12] Teodorovi´c, D., Dell'Orco, M. : *Bee colony optimization—a cooperative learning approach to complex transportation problems*. In: Proceedings of the 10th EWGT Meeting, Poznan, *13–16*, September 2005.

[13] Tereshko, V.: Reaction-*diffusion model of a honeybee colony's foraging behaviour*. In: Schoenauer M. (ed.) Parallel Problem Solving from Nature VI. Lecture Notes in Computer Science, *vol. 1917, pp. 807–816*, Springer, Berlin (2000).

[14] Holland, J.H.: *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI (1975).

[15] Lucic, P., Teodorovi´c, D. : *Transportation Modeling: An Artificial Life Approach*. ICTAI, *pp.216–223*.Washington D.C. (2002).