Dichotomous Search in ABC and its Application in Parameter Estimation of Software Reliability Growth Models

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

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

Ajith Abraham
Machine Intelligence Research Labs (MIR Labs), WA, USA
VSB-Technical University of Ostrava, Czech Republic
ajith.abraham@ieee.org

Abstract— **ABC (Artificial Bee Colony) is one of the most recent nature inspired algorithm (NIA) based on swarming metaphor. Proposed by Karaboga in 2005, ABC has proven to be a robust and efficient algorithm for solving global optimization problems over continuous space. In this paper, we propose a modified version of the ABC to improve its performance, in terms of converging to individual optimal point and to compensate the limited amount of search moves of original ABC. In modified version called Dichotomous ABC (DABC), the idea is to move dichotomously in both directions to generate a new trial point. The performance of the proposed algorithm is analyzed on five standard benchmark problems and also we explored the applicability of the proposed algorithm to estimate the parameters of software reliability growth models (SRGM). The proposed algorithm presents significant advantages in handling variety of modeling problems such as the exponential model, power model and Delayed S Shaped model.**

Keywords- Artificial Bee Colony, Bidirectional optimization, Software Engineering, Software Reliability

I. INTRODUCTION

In the past few decades several nature inspired algorithms (NIA) have emerged for solving global optimization problems. NIA algorithms may be classified as the ones based on natural evolution, commonly known as Evolutionary Algorithms (EA) and the ones that are based on behavioral pattern displayed by various species, particularly the ones that live in groups (or swarms). Evolutionary Algorithms (EAs) are optimization techniques based on the concept of a population of individuals that evolve and improve their fitness through probabilistic operators like recombination and mutation. These individuals are evaluated and those that perform better are selected to compose the population in the next generation. After several generations these individuals improve their fitness as they explore the solution space for optimal value. Some popular EA are Genetic Algorithms (GA) [1, 2], Evolutionary Strategies (ES) [3], Evolutionary Programming (EP) [4 – 7], Differential Evolution (DE) [8, 9] etc.

In the present study, the focus is on ABC, which is one of the recently proposed NIA. ABC follows the analogy of the socio-cooperative behavior demonstrated by honey bees in their search for nectar. A brief overview of the working of ABC algorithm is given in Section II.

ABC has been successfully applied for solving a variety of real life and benchmark problems. Its comparison with the contemporary algorithms has shown its competence in dealing with different types of problems [10 - 17].

The remaining of paper is organized as follow. In section III, a brief overview of software reliability growth models is discussed. Section IV, introduces the proposed DABC algorithm. Parameter settings for the algorithm, the considered benchmark problems and the criteria for the comparison of the algorithms are given in section V. The simulation results obtained are presented and discussed in section VI. Finally, the paper concludes with section VII.

II. OVERVIEW OF ARTIFICIAL BEE COLONY

ABC, as pointed out in the previous section, is one of the most recently defined NIA algorithms. It was proposed by Dervis Karaboga, Erciyes University of Turkey in 2005 [18]. ABC is motivated by the intricate and disciplined behavior displayed by honey bees [19 - 22]. In ABC system, artificial bees fly around in the search space, and some (employed and onlooker bees) choose food sources depending on the experience of themselves and their nest mates and adjust their positions. Some (scouts) fly and choose the food sources randomly without using experience. If the nectar amount of a new source is higher than that of the previous one in their memory, they memorize the new position and forget the previous one. Thus, ABC system combines local search methods, carried out by employed and onlooker bees, with global search methods, managed by onlookers and scouts, attempting to balance exploration and exploitation process.

In order to introduce the model of forage selection that leads to the emergence of collective intelligence of honey bee swarms, the three essential components: food sources, unemployed foragers and employed foragers are described as [20]:

978-1-4577-1123-7/11/$26.00 ©2011 IEEE

Food Sources (A and B in *Figure 2*): For the sake of simplicity, the "profitability" of a food source can be represented with a single quantity. In our function optimization problem, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution.

Unemployed foragers: If it is assumed that a bee have no knowledge about the food sources in the search field, bee initializes its search as an unemployed forager. There are two types of them, scouts and onlookers. Their main task is exploring and exploiting food source. At the beginning, there are two choices for the unemployed foragers: (i). it becomes a scout or (ii). it becomes a onlooker.

Scouts (S in *Figure 2*): randomly search for new food sources without any knowledge around the nest. The percentage of scout bees varies from 5% to 30% according to the information into the nest. [23]

Onlookers(R in *Figure 2*): The onlookers wait in the nest and search the food source through sharing information of the employed foragers, and there is a greater probability of onlookers choosing more profitable sources.

Employed foragers: They are associated with a particular food source which they are currently exploiting. They carry with them information about this particular source, the profitability of the source and share this information with a certain probability. After the employed foraging bee loads a portion of nectar from the food source, it returns to the hive and unloads the nectar to the food area in the hive. There are three possible options related to residual amount of nectar for the foraging bee.

- If the nectar amount decreased to a low level or exhausted, foraging bee abandons the food source and become an unemployed bee.(UF in *Figure 2*)
- If there are still sufficient amount of nectar in the food source, it can continue to forage without sharing the food source information with the nest mates.(EF2 in *Figure 2*)
- Or it can go to the dance area to perform waggle dance for informing the nest mates about the food source. (EF1 in *Figure 2*), as is shown in *Figure 1*.

In this way, the bees finally can construct a relative good solution of the optimization problems.

Figure 1. Waggle dance of honey bees [9]

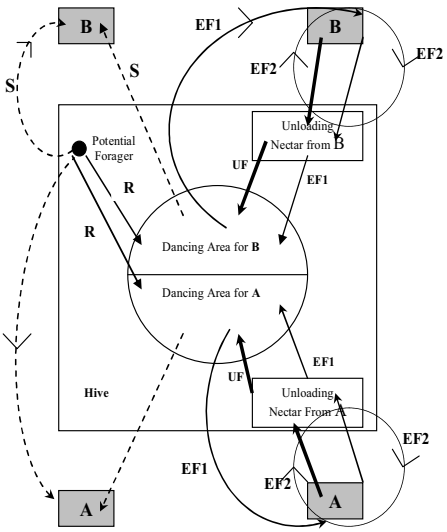


Figure 2. The behaviour of honey bee foraging for nectar [24]

Pseudocode of the ABC Algorithm

1. Initialize the population of solutions $X_{i,G}$.
2. Evaluate the population.
3. Cycle=1
4. repeat
5. Produce new solutions (food source positions) $V_{ij,G}$ in the neighborhood of $X_{ij,G}$ for the employed bees using the equation:

$$V_{ij,G} = X_{ij,G} + \Phi_{ij}(X_{ij,G} - X_{kj,G}) \tag{1}$$

   Where k ∈ {1,2,…,NP} and j ∈ {1,2,…D} are randomly chosen indexes; k has to be different from i; Φ_{ij} is a random number in the range [-1, 1].
6. Apply the greedy selection process between $X_{ij,G}$ and $V_{ij,G}$..
7. Calculate the probability values P_i for the solutions $X_{ij,G}$ by means of their fitness values using the equation:

$$P_i = \frac{fit_i}{\sum_{i=1}^{NP} fit_i} \tag{2}$$

   In order to calculate the fitness values of solutions the following equation is employed:

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

   Normalize P_i values into [0, 1]
8. Produce the new solutions (new positions) $V_{ij,G}$ for the onlookers from the solutions $X_{ij,G}$, selected depending on P_i, and evaluate them.

9. Apply the greedy selection process for the onlookers between $X_{ij,G}$ and $V_{ij,G}$..
10. Determine the abandoned solution (source), if exists, and replace it with a new randomly produced solution x_i for the scout using the equation:

$$X_{ij} = \min_j + rand(0,1) * (\max_j - \min_j) \quad (4)$$

where $\min_j$ and $\max_j$ are lower and upper bounds respectively.

11. Memorize the best food source position (solution) achieved so far.
12. Cycle = Cycle+1
13. until cycle= Maximum Cycle Number (MCN)

III. Software Reliability Growth Models

Software reliability is defined as the probability of failure free operation of a computer program in a specified environment for a specified period of time [25]. Failure process modeling represents a challenge because of the various nature of faults discovered and the methodologies to be used in order to isolate the faults [26, 27]. In the past three decades, hundreds of models were introduced to estimate the reliability of software systems [28 - 30].

A. Exponential Model (EXMP)

This model is known as a finite failure model was first provided in [31].

$$\mu(t;\beta) = \beta_0(1 - e^{-\beta_1 t})$$
$$\lambda(t;\beta) = \beta_0\beta_1 e^{-\beta_1 t} \quad (5)$$

$\mu(t; \beta)$ and $\lambda(t; \beta)$ represent the mean failure function and the failure intensity function, respectively. The parameters β_0 is the initial estimate of the total failure recovered at the end of the testing process (i.e. v_0). β_1 represents the ratio between the initial failure intensity $\lambda 0$ and total failure $v0$. Thus, $\beta_1 = \lambda_0/v_0$. It is important to realize that:

$$\lambda(t;\beta) = \frac{\partial \mu(t;\beta)}{\partial t} \quad (6)$$

B. Power Model (POWM)

The model objective is to compute the reliability of a hardware system during testing process. The model is based on the non-homogeneous Poisson process model. The Power model was provided in [32]. The equations which govern the relationship between the time t and both $\mu(t; \beta)$ and $\lambda(t; \beta)$ are:

$$\mu(t;\beta) = \beta_0 t^{\beta_1}$$
$$\lambda(t;\beta) = \beta_0\beta_1 t e^{\beta_1 - 1} \quad (7)$$

C. Delayed S-Shaped Model (DSSM)

The model represents a learning process since some improvement was added to the exponential model based the growing experience of the project team. This model describes the software reliability process as a delayed S-shaped model [33]. This model is also a finite failure model. The system equation for $\mu(t; \beta)$ and $\lambda(t; \beta)$ are:

$$\mu(t;\beta) = \beta_0(1 - (1 + \beta_1 t)e^{-\beta_1 t})$$
$$\lambda(t;\beta) = \beta_0\beta_1^2 t^{-\beta_1 t} \quad (8)$$

IV. Proposed DABC Algorithm

Bidirectional ABC (DABC) algorithm uses bidirectional random optimization (BRO) concept. The pseudo code of BRO method is:

$$if \qquad [f(x+d) < f(x)] \rightarrow x = x + d$$
$$elseif \qquad [f(x-d) < f(x)] \rightarrow x = x - d$$
$$else \rightarrow do\,not\,change \quad x$$

The BRO method searches both in forward and in reverse direction. This method expresses that if movement in the forward direction does not improve value of fitness/cost function, with a high probability, reverse movement improves value of fitness/cost function. In the proposed DABC, concept of BRO is used while producing new food source positions. In the new stage, if forward movement, in ABC algorithm, does not improve value of fitness/cost function i.e. $f(V_i) > f(X_i)$, the reverse movement, will be checked as:

$$V_{ij,G} = X_{ij,G} - \Phi_{ij}(X_{ij,G} - X_{kj,G}) \quad (9)$$

On other words, the DABC presents an inversion of the search direction by changing the sign. So a new offspring by this new donor member generates and while its fitness was better than fitness of current member, it is replaced with new offspring.

V. Benchmark Problems and Parameter Settings

Basic Unimodal and Multimodal test functions employed to test the efficiency of the proposed algorithm:

Sphere

$$f_1(x) = \sum_{i=1}^{n} x_i^2$$

with $-5.12 \le x_i \le 5.12$, $\min f_1(0,\ldots,0) = 0$

Restrigin's

$$f_2 = 10\ n + \sum_{i=1}^{n} \left(x_i^2 - 10\ \cos\ (2\pi x_i)\right)$$

with $-5.12 \le x_i \le 5.12$, $\min f_2(0\ldots, 0) = 0$

Rosenbrock's

$$f_3(x) = \sum_{i=1}^{n-1} \left[100\ \left(x_{i+1}^2 - x_i^2\right) + \left(1 - x_i^2\right)\right]$$

with $-30 \le x_i \le 30$, min $f_3(1,\ldots,1) = 0$

Griekwank

$$f_4(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

with $-600 \le x_i \le 600$, min $f_4(0,\ldots,0) = 0$

Zakharov

$$f_5(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$$

with $-5 \le x_i \le 10$, min $f_5(0,\ldots,0) = 0$

Parameter Tunning for:

ABC & DABC	
Colony Size (Employed and Onlooker bees)	100
Food Numbers (Food Sources)	50 (50% of the colony Size)
Scout Bee	one
Software Reliability Growth Models	
Food Numbers	2
Domain of α (search space)	-1000:1000
Domain of β (search space)	-1:1

The *limit* is taken 100.Random numbers are generated using inbuilt function *rand()* in DEVC++. In order to minimize the effect of the stochastic nature of the algorithms on the metric, the reported NFE or MCN for each function is the average over 30 trials. The maximum number of function evaluations is set 10^6. In every case, a run was terminated when an VTR is equals to 10^{-6} was reached or when the maximum number of function evaluation was reached. All algorithms are executed on Pentium IV, using DEV C++.

VI. Experimental Results and Discussion

A. Benchmark Problems

The proposed DABC algorithm is validated on a set of five benchmark problems having different dimensions and performances in terms of best and mean fitness function values, standard deviation *(SD),* shown in Table I, average number of function evaluations *(NFE)* and time taken are compared with the ABC, shown in Table II. These parts demonstrate that the proposed concept can improve the performance ABC. In order to compare convergence speeds, we use the *acceleration rate* (AR) which is defined as follows, based on the NFEs for the two algorithms ABC and DABC:

$$AR = \frac{NFE_{ABC}}{NFE_{BABC}} \tag{10}$$

Where $AR > 1$ means DABC is faster.

The number of times, for which the algorithm successfully reaches the VTR for each test function, is measured as the *success rate* (SR)

$$SR = \frac{No.of\ times\ reached\ VTR}{Total\ No.of\ trials} \tag{11}$$

Further, the *average acceleration rate* (AR_{ave}) and the *average success rate* (SR_{ave}) over test functions are calculated as follows:

$$AR_{ave} = \frac{1}{n}\sum_{i=1}^{n} AR_i \tag{12}$$

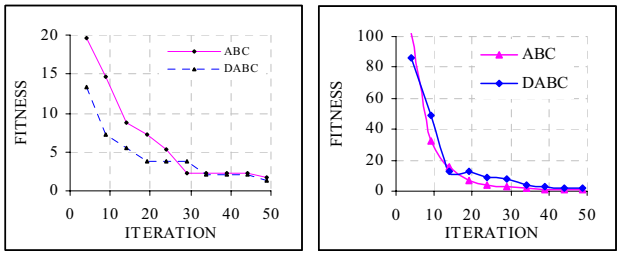
$$SR_{ave} = \frac{1}{n}\sum_{i=1}^{n} SR_i \tag{13}$$

TABLE I. Statistical Results in terms of Best, Mean and Standard Deviation (D* - dimension)

Function	D*	Algorithm	Best	Mean	SD
f_1	10	ABC	0.00057	0.00081	0.00015
		DABC	0.00059	0.00063	0.00004
f_2	4	ABC	0.00016	0.00060	0.00026
		DABC	0.00009	0.00052	0.00012
f_3	10	ABC	0.00059	0.00083	0.0018
		DABC	0.00041	0.00081	0.0010
f_4	10	ABC	0.00056	0.00080	0.00013
		DABC	0.00040	0.00063	0.00011
f_5	10	ABC	0.00042	0.00080	0.00017
		DABC	0.00034	0.00041	0.00018

TABLE II. Comparision of ABC and DABC: Average Number of Function Evaluations, Time, Success Rate, Avg. Success Rate, Acceleration Rate, and Avg. Acceleration Rate Taken by Benchmark Problems

Function	ABC		DABC		AR
	NFE (Time)	SR	NFE (Time)	SR	
f_1	9870 (0.3)	1	8580 (0.2)	1	1.15
f_2	11600 (0.1)	1	10120 (0.1)	0.9	1.15
f_3	83250 (2.9)	0.86	71901 (1.8)	1	1.16
f_4	90160 (2.8)	1	44080 (1.4)	1	2.05
f_5	12940 (1.0)	1	11440 (0.8)	1	1.13
Total	207820 (7.1)	4.86	146121 (4.3)	4.9	6.62 (1.65)
Average	41564 (1.42)	0.972	29224.2 (0.86)	0.98	1.42 (1.65)



(a) (b)

Figure 3. Convergence Graphs of (a) Restrigin's & (b) Griekwank problems

From the Table II it is analysed that the proposed algorithm ABC propforms 42% better in comparision of ABC. And DABC is more successful than ABC. The total time taken to execute the five benchmark problems by ABC is 7.1 where as DABC has taken only 4.86 sec, which clearly indicates the performance & efficiency of DABC. The convergence graph of Restigin and Griekwank are demonstrated in Figure 3.

B. Test/Debug data for estimating the parameters of software reliablity growth models

A field report data was developed to measure system faults during testing in a real-time application [34]. The software system consists of 200 modules with each having one KLOC of FORTRAN. DABC is used to find the best parameters to tune the exponential model, power model and Delayed S-Shaped model. A Test/Debug data set of 111 measurements presented in [35] was used for the experiments.

RMSE criterion is used to measure the performance of the proposed DABC. RMSE is frequently used to measure differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated. It is just the square root of the mean square error as shown in equation given below:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \qquad (14)$$

Where y_i represents the i^{th} value of the effort, $\hat{y}$ is the estimated effort and N is the number of measurements used in parameter estimation of growth models.

The convergence graph of the three growth models are shown in Figure 4. The computed parameters and RMSE (training & testing) of all the three software reliability growth models using ABC and proposed DABC algorithms are given in the Table III & IV respectively. It can clearly be analysed that Delayed S-Shaped model provided the minimum RMSE in comparison of other models.

TABLE III. ESTIMATED RELIABLITY GROWTH MODEL PARAMETERS USING ABC AND DABC

EXMP	ABC	$\mu(t;\beta) = 681.096(1 - e^{-0.158177})$
	DABC	$\mu(t;\beta) = 318.807(1 - e^{-0.188818})$
POWM	ABC	$\mu(t;\beta) = 28.8965\, t^{0.0134281}$
	DABC	$\mu(t;\beta) = 23.2249 t^{0.234016}$
DSSM	ABC	$\mu(t;\beta) = 779.724(1 - (1 + 0.01144\ t)e^{-0.01144})$
	DABC	$\mu(t;\beta) = 661.004(1 - (1 + 0.05084\, t)\, e^{-0.05084})$

TABLE IV. COMPUTED RMSE FOR TEST/DEBUG DATA

Model	Algorithm	RMSE Training	RMSE Testing
EXMP	ABC	27.0912	119.6425
	DABC	19.7196	72.0187
POWM	ABC	41.0482	158.6753
	DABC	34.3901	81.9231
DSSM	ABC	21.0830	17.091
	DABC	18.9215	29.8051

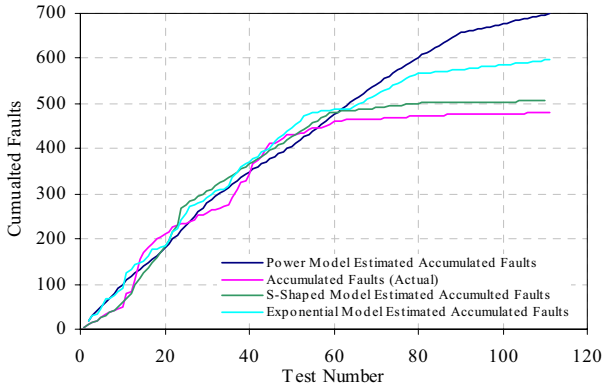


Figure 4. Actual and accumulated failures for the three growth models using test/debug data (111 Measurements)[30]

VII. CONCLUSION

In this paper the modified version, the DABC, to improve performance of algorithm, was used from bidirectional optimization and applied successfully on 5 benchmark functions. Non-parametric analysis of results demonstrated that the proposed methods have a better success rate than original ABC. Further we have applied the proposed algorithm to estimate the parameter of software reliability models (i.e. exponential model, power model and S-shaped model).The estimated model parameters were used to predict the faults in a software system during the testing process.

REFERENCES

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

[2] J.H Holland,: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor, 1975.

[3] I. Rechenberg: Evolution strategie 1994. Frommann-Holzboog, Stuttgart, 1994.

[4] T. Bäck and H.-P. Schwefel, "Evolutionary computation: An overview," in *Proc. IEE Int. Conf. Evolutionary Computation*, 1996, pp. 20–29.

[5] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, 1997.

[6] D. Fogel, *Evolutionary Computation*. Piscataway, NJ: IEEE Press,, 1995.

[7] L. J. Fogel, A. J. Owens, and M. J.Walsh, *Artificial Intelligence through Simulated Evolution*. New York: Wiley,, 1966.

[8] R.Storn and K. Price,"DE –a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Space", Technical Report TR-95-012, ICSI, March 1995. Available via the Internet: ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z,1995

[9] R. Storn and K. Price, "Differential Evolution – a simple and efficient Heuristic for global optimization over continuous spaces", Journal Global Optimization. 11, pp. 341 – 359,1997.

[10] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," Applied Soft Computing, vol. 9, pp. 625—631, 2009.

[11] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters," Journal of The Franklin Institute, vol. 346, pp. 328—348, 2009.

[12] J. W. Ponton and J. Klemes, “Alternatives to neural networks for inferential measurement,” Computers and Chemical Engineering, vol. 17, pp. 42—47, 1993.

[13] R. S. Rao, S. Narasimham, and M. Ramalingaraju, “Optimization of distribution network conFigureuration for loss reduction using artificial bee colony algorithm,” International Journal of Electrical Power and Energy Systems Engineering (IJEPESE), vol. 1, pp. 116—122, 2008.

[14] D. Karaboga, B. Akay, and C. Ozturk, “Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks,” in LNCS: Modeling Decisions for Artificial Intelligence. Springer-Verlag, 2007, vol. 4617/2007, pp. 318—329.

[15] P. Pawar, R. Rao, and J. Davim, “Optimization of process parameters of milling process using particle swarm optimization and artificial bee colony algorithm,” in Int. Conf. Advances in Mechanical Engineering, 2008.

[16] D. Karaboga and B. Basturk, “Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems,” in LNCS: Advances in Soft Computing-Foundations of Fuzzy Logic and Soft Computing. Springer- Verlag, 2007, vol. 4529/2007, pp. 789—798.

[17] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, “A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem,” Information Sciences, In Press.

[18] D. Karaboga, B. Basturk, “A Powerful And Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm”, Journal of Global Optimization, Springer Netherlands, vol.39, no.3, pp.459-471, 2007.

[19] Karaboga, D., B. Basturk, "On the performance of artificial bee colony (ABC) algorithm", *Applied Soft Computing,* Vol. 8, pp. 687-697, 2008.

[20] D. Karaboga, “An Idea Based on Honey Bee Swarm for Numerical Optimization”, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[21] B. Basturk, D. Karaboga, “An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization”, IEEE Swarm Intelligence Symposium 2006, Indianapolis, Indiana, USA. May 12- 14, 2006.

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

[23] T.D. Seeley, *The Wisdom of the Hive*, Harvard University Press, Cambridge, MA, 1995.

[24] Haibin Duan, Zhihui Xing, and Chunfang Xu,”An Improved Quantum Evolutionary Algorithm Based on Artificial Bee Colony Optimization” W. Yu and E.N. Sanchez (Eds.): Advances in Computational Intell., AISC 61, pp. 269–278, 2009.

[25] John Musa, A. Iannino, and K. Okumoto. *Software Reliability: Measurement, Prediction, Applications*. McGraw Hill, 1987.

[26] H. Pham. *Software Reliability*. Springer-Verlag, 2000.

[27] P. G. Bishop and R. Bloomfield. Worst case reliability prediction on a prior estimate of residual defects. In *Proceedings of the 13 th IEEE International Symposium on Software Reliability Engineering (ISSRE-2002)*,pages295–303,2002.

[28] M. Xie. Software reliability models - past, present and future. *In N. Limnios and M. Nikulin (Eds) Recent Advances in Reliability Theory: Methodology,Practice and Inference*, pages 323–340, 2002.

[29] S. Yamada, M. Ohba, and Osaki S. S-Shaped software reliability growth models and their applications. *IEEE Trans. Reliability*, pages 289– 292, 1984.

[30] Shigeru Yamada. Software reliability models and their applications: A survey. In *International Seminar on Software Reliability of Man-Machine Systems - Theories Methods and Information Systems Applications - August 17-18, Kyoto University, Kyoto, Japan*, 2000.

[31] J. D. Musa, “A theory of software reliability and its application,” *IEEE Trans. Software Engineering*, vol. 1, pp. 312–327, 1975.

[32] P. B. Moranda, “Predictions of software reliability during debugging,” in *Proceedings of Annual Reliability and Maintainability Symposium*, pp. 327–332, 1975.

[33] S. Yamada, M. Ohba, and O. S., “S-Shaped relaibility growth modeling for software error detection,” *IEEE Trans. Reliability*, pp. 475–478, 1983.

[34] Y. Tohman, K. Tokunaga, S. Nagase, and Murata Y. Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution model. IEEE Trans. on Software Engineering, pages 345–355, 1989.

[35] Alaa Sheta. Reliability growth modeling for software fault detection using particle swarm optimization. In 2006 IEEE Congress on Evolutionary Computation, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21, 2006., pages 10428–10435, 2006.