Journal of Modern Computing and Engineering Research

JMCER

Volume 2022, p. 1-8

https://jmcer.org

The Integration of Flower Pollination and Real Coded Genetic Algorithms as a Tool for SRGMs Parameters Estimation

Marwah M.A. Dabdawb[1] and Jamal Salahaldeen Alneamy[2]

[1,2]*Software Engineering Department, College of Computer Science and Mathematics, University of Mosul, Mosul 41002, Iraq*

[1]marwa_marwan21@uomosul.edu.iq, [2]jamal_alneamy@uomosul.edu.iq

Received: April 13, 2022 Revised: May 30, 2022 Accepted: June 3, 2022

Abstract: *Software quality contains many characteristics that can be measured and estimated to reach the required quality, and it can be said that software reliability is one of the most important characteristics that can be estimated through software reliability growth models (SRGMs). These models contain parameters whose values affect the accuracy of the measured reliability, and for that, the values of these parameters are estimated in several ways, such as swarm intelligence. In this work, a suggested binding between the flower pollination algorithm and the real coded genetic algorithm (named overlapping FPA (OFGA)) was used to estimate the parameters of SRGMs, and the results showed the superiority of (OFGA) over the past binding that compared with (namely, HFPA) in parameters estimating accuracy and performance using the same dataset.*

Keywords: *Swarm intelligence; Software Reliability Growth Models; Real Coded Genetic Algorithm; Flower Pollination Algorithm.*

1. Introduction

Software reliability is defined as "how the software achieves the desired requirements" and also defines the probability that the software will run without failure for a certain period of time and in a specific environment" (Sheakh & Singh, 2012). Trying to find out whether the program that will be delivered to the customer is reliable or not is a difficult thing because software providers need to know the reliability of the software before providing the customer with it. Software reliability models provide us with such information prior to delivery (Haque & Ahmed, 2022). During the last decades, many software reliability growth models (SRGMs) have been proposed and analysed to measure software reliability growth. The equations for these models contain parameters that are estimated based on failure data.

Since most SRGMs are non-linear, it becomes difficult to use traditional methods for this purpose. Based on this principle, it became necessary to find other methods for estimating the reliability of non-linear equations, so soft computing techniques were used such as neural networks, fuzzy logic, support vector machine (SVM), evolutionary computing, Bayesian network and chaos theory. In this work, the focus will be on Evolutionary Computing System which is further divided into evolutionary algorithms (Genetic Algorithm) and swarm intelligence techniques (Soltanali et al., 2021).

There are many previous studies that used the algorithms of these two branches in estimating the parameters of SRGMs. (Sheta & Al-Salt, 2007) estimated models' parameters by using Particle Swarm Optimization (PSO) which outperformed the traditional method like least squares estimation (LSE). In the same context, (Shanmugam & Florence, 2012) use an estimation method based on Ant Colony Algorithm ACO and compared it with PSO by using the same datasets, after all, ACO shows more accuracy than PSO in this estimation. (AL-Saati & Abd-AlKareem, 2013) employed the Cuckoo Search (CS) in parameters estimation which was more closely to the solution than PSO and ACO.

(Sharma et. al., 2014) proposed a Modified Artificial Bee Colony ABC called Dichotomous ABC (DABC), where the modified version outperforms the original algorithm in estimation accuracy. (Alneamy & Dabdoob, 2019) used a Flower Pollination Algorithm (FPA) in parameters estimation and the results were compared with some previous swarm algorithms, then the algorithm was linked with the genetic algorithm to be Hybrid Flower Pollination

Corresponding author: Marwah M.A. Dabdawb, Marwa_marwan21@uomosul.edu.iq

Algorithm (HFPA). In this suggested binding, the number of execution iterations was divided into two halves, the first half executed by using GA then the output was considered as input for FPA to execute the second half. HFPA has proven to improve performance and speed in execution in comparison with FPA alone. A modified genetic algorithm MGA was proposed by (Jung & Huang, 2010) in the SRGMs parameters' estimation process. their results showed that the proposed (MGA) was faster in reaching the solution than traditional genetic algorithms.

In this work, a proposed binding between FPA and Real Values Encoded Genetic Algorithm (RGA) to produce (overlapping FPA_ RGA (OFGA)) which will be used to estimate the parameters of SRGMs. Then the obtained results will compare with the previously proposed HFPA algorithm.

The rest of this paper is organized as follows: Section 2 surveys various types of SRGMs used in this work. Then in Section 3, the research methodology is explained. Finally, the experimental results are illustrated and discussed in Sections 4 and 5.

2. SRGMS

In recent decades, studies in software reliability growth models have made great progress, and many reliability models have already been used in this field. The goal of SRGMs is to predict future failure behaviour, which uses the time between failures or the number of observed failures in a specified time period as its data (Zhen et al., 2020). There are many models of software reliability growth, but the non-homogeneous Poisson process (NHPP) models are the most accurate and the most widely used and they are non-linear models.

These models help software engineers to decide whether software reliability has reached an acceptable level and determine when the system is ready for delivery (Alneamy & Dabdoob, 2019). NHPP models assume that the number of defects detected during time (t) follows NHPP in function of the mean value $\mu(t)$. The derivation of the mean value function leads to $\lambda(t)$, which is the failure intensity of the program that decreases as defects are discovered and eliminated (Haque & Ahmed, 2022). There are many NHPP models, and for comparison purposes, three of them will be used, as shown in Table 1 below:

Table 1: The SRGM models used in this research (Alneamy & Dabdoob, 2019).

Model	Description	Mean Value Function	Estimated Parameters
G-Om	This model was introduced by scientists Goel and Okumoto in 1978 and is also called Exponential NHPP Model	$\mu(t) = a(1 - e^{-bt})$	a and b
POWm	The power model is one of the oldest models and was suggested by the scientist Duane in 1964. It is a graphical method to perform the analysis of reliability growth data, and it is simple and straightforward to understand.	$\mu(t) = at^{b}$	a and b
DSSm	Delayed S-Shaped model suggested by Yamada in 1983	$\mu(t) = a(1 - (1 + bt)e^{-bt})$	a and b
where: a > 0: represents the initial estimate of the total number of failures that will be detected at the end of the test. b > 0: represents the failure density of the defect. t: represents the time of failure			

3. Research Methodology

In this study, the parameters of three models of SRGMs are estimated. To do so, OFGA is used which is overlapping between FPA and RGA algorithms. First, each algorithm will be illustrated separately then, the proposed overlapping will be explained.

3.1. FPA

The flower pollination algorithm is one of the algorithms inspired by nature and inspired by the process of pollinating flowers to reproduce them again. From the point of view of biological evolution, the goal of flower pollination is the survival of the fittest and the ideal reproduction of plants in terms of numbers, and this is in fact a process of optimizing the species of the plant (Singh et al., 2021).

Pollination can take two main forms: biotic and abiotic. About 90% of flowering plants belong to biological pollination, where pollen is spread by pollinators such as insects and animals, and about 10% of pollination is abiotic, that is, it does not need pollinators, but rather Wind and water aid in pollination, and pollination is of two types:

- Self-Pollination: It occurs when pollen from a flower pollinates the same flower or another flower on the same plant.
- Cross-Pollination: It occurs when pollen grains are transferred from one flower on one plant to another flower of another plant (Mergos & Yang, 2021).

The constancy of a flower and the behaviour of pollinators in the pollination process can be described by the following four rules:

1- Biotic and cross-pollination are considered as a global pollination process in which pollinators perform Lévy flight. The mathematical representation for this rule is as follows:

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)\left(x_i^t - x_{best}\right) \tag{1}$$

where:
x_i^t : Solution in iteration *t*.
x_{best}: Best solution found among all solutions at the current iteration.
γ: Scaling factor to control step size.
x_i^{t+1} : Solution in a new iteration.
$L(\lambda)$: Obeys a Lévy distribution and can be calculated as follow:

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{S^{1+\lambda}} \tag{2}$$

where:
λ: standard gamma function, it is recommended that *Γ(λ) = 1.5*.
S: is a step size that calculates as follows:

$$S = \frac{U}{|V|^{1/\lambda}} \tag{3}$$

where:
$U \sim N(0, \sigma^2)$: U follows a normal distribution with arithmetic mean equal to zero and a standard deviation equal to σ^2.
$V \sim N(0,1)$: V follows a normal distribution with arithmetic mean equal to zero and a standard deviation equal to one.

2- Abiotic and self-pollination are considered as local pollination. The mathematical representation for this rule is as follows:

$$x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right) \tag{4}$$

where:
x_i^t : solution in iteration *t*.
x_j^t, x_k^t: Pollen from different flowers on the same plant where *j, k* are randomly selected indices.
ϵ: is a random variable that follows a uniform distribution *U (0,1).*
x_i^{t+1} : solution in a new iteration.

3- The constancy of the flower can be considered as the probability of reproduction, which is proportional to the similarity of the two flowers concerned.

4- The switch or interaction between mass and local pollination can be controlled by the switch probability *p∈[0,1].*

In fact, it is clear that each plant has several flowers, and each flower releases many pollen grains, and for ease, it was assumed that each plant has one flower, and each flower has one pollen, so there is no need to distinguish between them (Mergos & Yang, 2021). Finally, the steps of the FPA are as follows (Nguyen & Dao, 2019):

Step1: randomly generating the initial population of *(n)* flowers by the following equation:

$$\vec{X} = \min + rand * (\max - \min) \tag{5}$$

where:
max, min: maximum and minimum domain of search.
rand: random number between [0,1].
Defining the objective function *f(x)*, *x*= (x_1, x_2 ,..,x_n) to find the best solution for, determining the number of cycles *(MCN)*, specifying the cycle counter = 0, and specifying the switching probability *p∈[0,1].*

Step 2: Calculate the objective function of all flowers and take the best solution let it be (x_{best}).

Step 3: Check the stop criterion. As long as the cycle counter is less than the total number of cycles *(MCN)*, go to step 4, or else stop.

Step 4: For each flower, a random number is generated, and let *r* follow the uniform distribution *U (0,1).*

Step 5: If *r<p*, global pollination is carried out, which follows a Lévy distribution, otherwise, local

pollination is carried out, which follows a uniform distribution.

Step 6: Calculate the objective function of the new solution, let it be f_j and compare it to the objective function of the previous solution f_i. If f_j is better than f_i, the previous solution is replaced by the new solution $x_i=x_j$, otherwise, the previous solution x_i remains the same.

Step 7: find the best new solution, let it be $(x_{new-best})$, and compare it with (x_{best}). If its value is better, then the best solution is updated $(x_{best}=x_{new-best})$ and if it is worse, (x_{best}) remains as it is.

Step 8: Update the cycle counter *cycle=cycle+1* and move to the third step.

Step 9: (x_{best}) is the best solution.

3.2. Real Coded Genetic Algorithm (RGA)

A genetic algorithm is a research technique studied by the scientist John Holland, that can solve optimization problems and it has been widely used in many scientific fields. The life cycle of a genetic algorithm consists of (Alneamy & Dabdoob, 2017):

1- **Generate Initial Population**

A population consists of some randomly generated individuals, each individual is considered a potential solution to the problem to be solved called a chromosome. The initial population is created as in equation (5).

2- **Evaluation**

The evaluation is done by defining the fitness function of each chromosome. This fitness function is an indicator of the chromosome that shows how close this chromosome is to the desired solution.

3- **Termination Condition**

The stop condition is checked after each generation to determine whether the algorithm should continue or stop.

4- **Generate New Population**

Generating a new generation consists of three steps as follows:

I. Selection

It is the process of selecting individuals with the best fitness function to mate and produce a new generation. There are many selection methods, including:

- *Top-Mate Selection:*

The first parent with the best fitness function is chosen and the second parent is chosen at random.

II. Crossover

The idea of a crossover is that the resulting offspring chromosome is better than the parent's if it takes good traits from both. Crossing frequency is controlled by crossover probability *(pc)*. There are several types of crossovers:

- *Heuristic Crossover:*

Used with real coding, only one offspring is generated, while the other offspring is the passing of the parent with the best fit to the new generation without any processing, and the following equations show this type of intersection:

$$fitness_{parent1} \quad is \quad better \quad than \quad fitness_{parent2}$$

$$offspring_1 = parent_1 \tag{6}$$

$$offspring_2 = parent_1 \pm r * \left(parent_1 - parent_2 \right)$$

where:

r: is a random number between [0,1].

III. Mutation

It is a change in the value of one or more genes for a chromosome depending on mutation probability *(pm)*. Types of mutations:

- *Non-Uniform Mutation:*

The value of the paternal chromosome changes within a limited range given the current generation number:

$$offspring_i = \begin{Bmatrix} parent_i + \left(upperbound_i - parent_i \right) * f\left(G\right) \\ or \\ parent_i - \left(parent_i - lowerbound_i \right) * f\left(G\right) \end{Bmatrix} \tag{7}$$

where:

f(G): is the range function considering the number of the current generation *(G)*. The function *f* (*G*) is as follows:

$$f\left(G\right) = \left(r * \left(1 - \frac{G}{G_{\max}} \right) \right)^b \tag{8}$$

where:

(G_{max}): Is the maximum number of generations

(b): Is a shape parameter.

(r): Is a uniform random number between 0 and 1.

The steps of the real coded genetic algorithm are as follows (Flores-Moran et al., 2018):

Step 1: randomly generate an initial population by using equation (5), and determine the objective function $f(x)$, $x=(x_1,x_2,..,x_n)$ to find the best solution for it, determine the maximum number of generations G_{max}, determine the current generation $G = 0$, and determine the probability of crossover *(pc)* and probability of mutation *(pm)*.

Step 2: calculate the target function for all chromosomes.

Step 3: Examining the criterion of stopping, as long as the condition for stopping is not met, move to step 4, or else stop.

Step 4: Parents are selected to produce the offspring using the Top-Mate Selection method.

Step 5: Generate a random number p_{cross}, if $p_{cross}<pc$ is crossed using the heuristic cross method.

Step 6: Generate a random number p_{mut}, if $p_{mut}<pm$ is mutant using non-uniform mutation.

Step 7: Placing the resulting chromosomes (children) into the new generation.

Step 8: As long as the size of the new generation is less than N, go to step 4, otherwise move to step 9.

Step 9: Replacing the previous generation with the current generation, updating the current generation $G=G+1$, and moving to step 2.

3.3. OFGA

Each algorithm has its advantages and disadvantages in the field in which it is used. In terms of optimization, the GA had many advantages like robustness and efficiency, while its drawbacks are that it may easily fall to a local minimum which is not the true solution, and sometimes have premature convergence or instability (Katoch et al., 2021). Regarding FPA, the advantages lay in its simple concept and accuracy but it consumes more iteration to reach the optimal solution (Couceiro et al., 2011).

For the purpose of collecting the advantages of the aforementioned algorithms, it was proposed to divide the population of one iteration and implement the two algorithms on it. In other words, in each iteration, the population will be divided into two sections, FPA will be implemented in the first section and the RGA will be implemented in the second section and the new population is generated. For the next iteration, the roles are exchanged, that means FPA will be executed in the second section, which is the output of the RGA in the previous iteration and the RGA implements in the first section, which is the output of the FPA in the previous iteration and so on until the condition for stopping is met. The FPA was deemed the main algorithm, in which the number of individuals was larger than those assigned to RGA. Figure 1 illustrates the proposed overlapping.

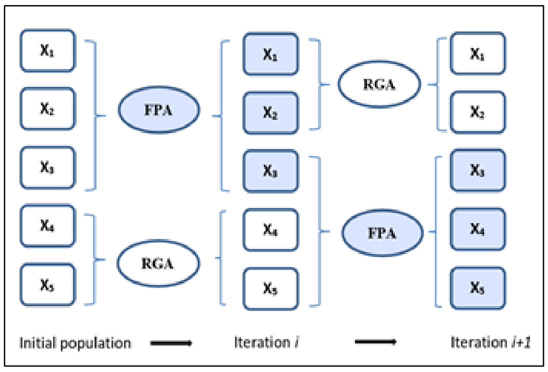

Figure 1: Proposed overlapping between FPA and RGA.

The steps of the overlapping FPA_RGA are as follows:

Step 1: Randomly generate an initial population using equation (5), set all tuning parameters as shown in table 2, and set *flag=0*.

Step 2: If *flag=0*, then set the *first* algorithm to be FPA, and set the *second* algorithm to be RGA. Else, set the *first* algorithm to be RGA, and set the *second* algorithm to be FPA.

Step 3: Divide the population into two parts: the first is an entry for the *first* algorithm, and the second part is an entry for the *second* algorithm.

Step 4: Examining the criterion of stopping, as long as the cycle counter is less than the total number of cycles, go to step 5, or else move to step 8.

Step 5: Implement the *First* and *second* algorithms as described previously.

Step 6: recombine the two parts to be the new generation.

Step 7: Updating the current generation $G=G+1$, set *flag=~flag*, and moving to step 2.

Step 8: Output the best solution.

4. Results

For the purpose of comparison with the results obtained from the previous study in (Alneamy & Dabdoob, 2019), the same data set was used in addition to the use of the same tuning parameters as shown in Table 2. Also, in the work of (Alneamy &

Dabdoob, 2019) HFPA was proposed which is binding between FPA and RGA. it assumed that the whole number of iterations assigned to the hybrid algorithm is split equally between FPA and RGA. Firstly, RGA is executed until it consumes its own half of the whole iterations to deliver a temporary output result represented in final population. Then, the output population from RGA consider as an initial population for FPA, which in turn, consumes the second half of the whole iterations until the stop criteria is met. The output from FPA is the final result for estimation.

Table 2: The tuning parameters for the OFGA.

Operator Value	Value
Domain search for a	[-1000,1000]
Domain search for b	[-1,1]
Search dimensions	2
Total no. of population	20
No. of search agents for FPA	14
No. of chromosomes for RGA	6
Maximum iterations	1000
Chromosome representation type	Value encoding
Selection type	Top-mate selection
Crossover type	Heuristic Crossover
Mutation type	Non-Uniform Mutation
Crossover rate	0.5
Mutation rate	0.1

Table 3 shows the results of the implementation of OFGA and compares them with the results obtained from the implementation of HFPA. Three models were used: G_O, POW, and DSS, and for each model, Root Mean Square Error (RMSE) was calculated which reflect how close the estimated value is to the real value, and whenever the value of RMSE is smaller that means the better was the estimation. Furthermore, the iteration needed to reach the best solution for each model was collected, and finally, the estimated values for the parameters a and b were shown. Figures 2 to 4 show plotting for observed and estimated data for the three models by using OFGA.

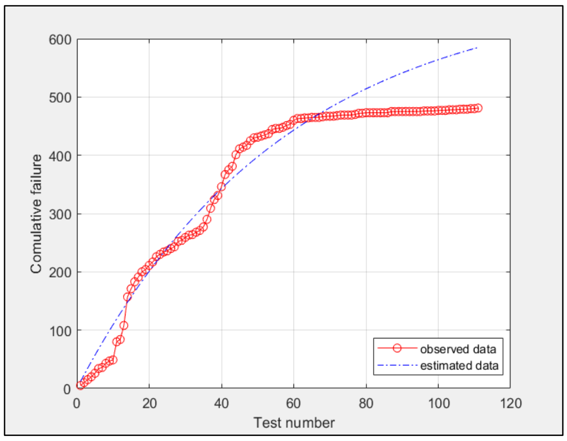

Figure 2: Observed and estimated data using G_O model.

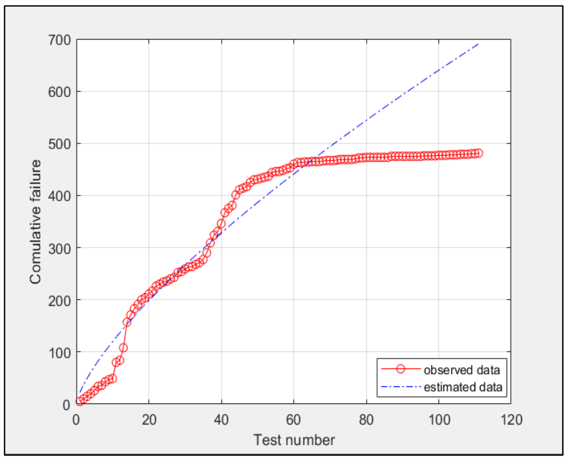

Figure 3: Observed and estimated data using POW model.

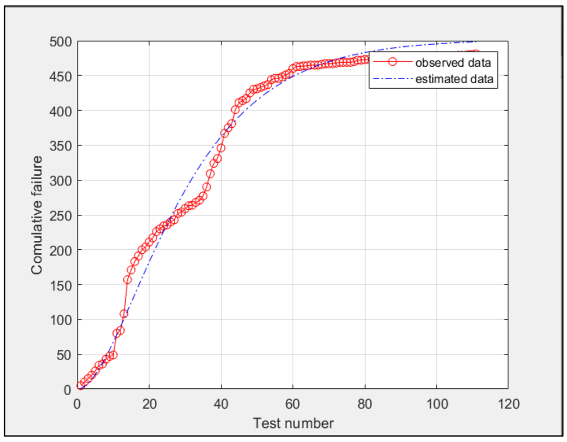

Figure 4: Observed and estimated data using DSS model.

Table 3: A comparison between HFPA and OFGA.

HFPA [9]						
Model	**RMSE Testing**	**No. of Iterations**	**Parameter *a***	**Parameter *b***		
G_O	77.859	366	684.3160	0.0174		
POW	146.785	571	22.3832	0.7281		
DSS	16.627	237	501.8481	0.0636		
OFGA						
Model	**RMSE Testing**	**No. of Iterations**	**Parameter *a***	**Parameter *b***	**RMSE Improvement**	**No. of Iterations Improvement**
G_O	4.786	27	496.1376	0.0360	93.852 %	92.622 %
POW	29.230	15	29.8988	0.6014	80.086 %	97.373 %
DSS	15.592	46	502.0888	0.0618	6.224 %	80.590 %

5. Conclusion

In this work, overlapping between FPA and RGA called OFGA was proposed to estimate two parameters of three models of software reliability and they are: G_O, POW, and DSS. Our results compared to another proposed binding between the aforesaid algorithms called HFPA. the results show that OFGA outperformed HFPA in estimation accuracy and speed of execution. In future works, other binding methods can be proposed between these two algorithms used in this paper. In addition, other suitable algorithms may be used for parameters estimation problem.

REFERENCES

Alneamy, J. S. M., & Dabdoob, M. M. A. (2017). The use of original and hybrid grey wolf optimizer in estimating the parameters of software reliability growth models. *International Journal of Computer Applications*, *167*(3), 12-21.

Alneamy, J. S., & Dabdoob, M. M. (2019). The use of original and hybrid flower pollination algorithm in estimating the parameters of software reliability growth models. *Journal of Education and Science*, *28*(2), 196-218.

AL-Saati, N. & Abd-AlKareem, M. (2013). The use of cuckoo search in estimating the parameters of software reliability growth models. *International Journal of Computer Science and Information Security, 11*(6).

Couceiro, M. S., Ferreira, N. M., & Tenreiro Machado, J. A. (2011). Fractional order Darwinian particle swarm optimization. In *Symposium on Fractional Signals and Systems* (pp. 127-136).

Flores-Morán, E., Yánez-Pazmiño, W., & Barzola-Monteses, J. (2018, May). Genetic algorithm and fuzzy self-tuning PID for DC motor position controllers. In *2018 19th International Carpathian Control Conference (ICCC)* (pp. 162-168). IEEE.

Haque, M. A., & Ahmad, N. (2022). Key issues in software reliability growth models. *Recent Advances in Computer Science and Communications*, *15*(5), 741-747.

Haque, M. A., & Ahmad, N. (2022). Key issues in software reliability growth models. *Recent Advances in Computer Science and Communications*, *15*(5), 741-747.

Hsu, C. J., & Huang, C. Y. (2010, July). A study on the applicability of modified genetic algorithms for the parameter estimation of software reliability modeling. In *2010 IEEE 34th Annual Computer Software and Applications Conference* (pp. 531-540). IEEE.

Katoch, S., Chauhan, S. S., & Kumar, V. (2021). A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, *80*(5), 8091-8126.

Mergos, P. E., & Yang, X. S. (2021). Flower pollination algorithm parameters tuning. *Soft Computing*, *25*(22), 14429-14447.

Nguyen, T. T., Pan, J. S., & Dao, T. K. (2019). An improved flower pollination algorithm for optimizing layouts of nodes in wireless sensor network. *IEEE Access*, *7*, 75985-75998.

Shanmugam, L., & Florence, L. (2012). A comparison of parameter best estimation method for software reliability models. *International Journal of Software Engineering & Applications*, *3*(5), 91-97.

Sharma, T. K., Pant, M., & Abraham, A. (2011, October). Dichotomous search in ABC and its application in parameter estimation of software reliability growth models. In *2011 Third World Congress on Nature and Biologically Inspired Computing* (pp. 207-212). IEEE.

Sheakh, T. H., & Singh, V. (2012). Taxonomical study of software reliability growth models. *International Journal of Scientific and Research Publications*, *2*(5), 1-3.

Sheta, A., & Al-Salt, J. (2007). Parameter estimation of software reliability growth models by particle swarm optimization. *AIML Journal*, *7*(1), 14, 55-61.

Singh, P., & Mittal, N. (2021). An efficient localization approach to locate sensor nodes in 3D wireless sensor networks using adaptive flower pollination algorithm. *Wireless Networks*, *27*(3), 1999-2014.

Soltanali, H., Rohani, A., Abbaspour-Fard, M. H., & Farinha, J. T. (2021). A comparative study of statistical and soft computing techniques for reliability prediction of automotive manufacturing. *Applied Soft Computing*, *98*, 106738.

Tarun Kumar Sharma, Millie Pant and Ajith Abraham, "Dichotomous Search in ABC and its Application in Parameter Estimation of Software Reliability Growth Models", 978-1-4577-1124-4/11/$26.00_c IEEE, 2011.

Zhen, L., Liu, Y., Dongsheng, W., & Wei, Z. (2020). Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. *IEEE Access*, *8*, 29354-29369.

Marwah M.A. Dabdawb received her Bachelor degree in Software Engineering in 2008 from the University of Mosul/ Iraq. Her M.Sc. degree was from the same university in 2018. She currently works as a faculty member at Software Department, University of Mosul, Iraq. Her main research interests include Software Engineering and Artificial Intelligence.

Jamal Salahaldeen Alneamy received his Ph.D. degree in Computer Science from the University of Mosul in 2006. He is working as a faculty member at the Software Dept., University of Mosul, Iraq. His current research interests include Computing in Mathematics, Natural Science, Engineering and Medicine, Artificial Intelligence, and Artificial Neural Network.

.