An Efficient Method for Enhancing Reliability and Selection of Software Reliability Growth Model through Optimization Techniques

Mallikharjuna Rao K.[1]*, K. Anuradha[2]
Department of IT, GITAM University, Visakhapatnam, Andhra Pradesh, India.
Department of CSE, School of Computing, GRIET, Hyderabad, Telangana, India.

*Corresponding author. Tel.: +91-9652861997; email: rao.mkrao@gmail.com
doi: 10.17706/jsw.12.1.1-18

Abstract: Software reliability engineering has recently turned out to be an interesting research topic in the field of software engineering. For the purpose of reliability calculation of software, various software reliability models have been designed based on the application of software in particular fields. The ability of the particular model in estimating the failure rate, reliability, and cost of the software are the major requirement in reliability modeling since any sort of failure or fault in the software can make the entire system unreliable to perform the desired operation. This paper proposes an efficient software reliability growth model (SRGM) model selection for estimating the reliability of the software. The reliability model selection criteria are generally based on the improved computational time and better failure rates. The selection of the model is done by utilizing optimization techniques. Here we have used modified cuckoo search optimization and modified ABC in order to find the effectiveness of the reliability model. Using these optimization algorithms, we evaluate various measures of the reliability models and are compared with that of other models. Here two different optimization approaches are used since we can efficiently find the best model using these algorithms.

Key words: Modified cuckoo search algorithm, modified artificial bee colony optimization, genetic algorithm, software reliability modeling, software reliability growth model, optimization techniques.

1. Introduction

Computer systems have become a very important and essential part of our daily lives. Computer hardware and software together form a complete system. A software system automates the working of a system. In this century, we hardly ever see any industry or other firms functioning without the help of an entrenched software scheme. Such an addiction on software systems has ended it indispensable to construct more reliable software's. The reliability requirement is even higher for the safety critical real-time control system software's [1]. The advancements in the information technology have changed the human life and society as well as software development dynamically. It has added a variety of dimensions like e-learning, e-conferencing, e- commerce, e-meeting, e-governance and the list is now becoming endless [2].Building reliability growth models for predicting software reliability represents a challenge for software testing engineering. The forecast of the amount of defects in software aid in figuring out the software release day and supervise project assets. Most software reliability growth models, known in history, have

two or three parameters to be estimated. They include the expected number of failures in the software by the end of the testing process and the initial failure intensity [3].

Developing and maintaining the software reliability has become essential due to its escalating dependence and demands as well as the functionality has become decisive bearing in mind the factors like reliability, safety of human lives and security issues as well. On the developers and users point of view, the major issues are specification, assessment and verification of these quality descriptions [4]-[5]. Diverse Machine Learning (ML) and Soft Computing techniques, such as Genetic Programming (GP), Artificial Neural Networks (ANN), Genetic Algorithms (GAs), Fuzzy Logic (FL), Evolutionary Strategies (ESs), and Particle Swarm Optimization (PSO) are using in solving the problems existing in software engineering in the 21 century. The recent trend being the usage of Particle Swarm Optimization (PSO) algorithm to calculate approximately the SRGM parameters. The technique illustrated considerable advantages in managing a wide range of modeling problems such as the Exponential Model (EXPM), power model (POWM) and Delayed S-Shaped Model (DSSM). Similar to the above process, Parameter assessment of Hyper-Geometric Distribution Software Reliability Growth Model by utilizing Genetic Algorithms is an additional paradigm [6].

Execution time is one of the critical parameters in estimating the reliability of a system since it affects system usage to a great extent. Reliability is not time dependent. If a system logic path contains error, failures occur when it is executed. Reliability growth is enhanced when the detected errors are corrected [7]. Reliability of software defined as the probability that the software will work before it struck with an error in the given conditional environment. The behaviors of the software reliability in terms of different failure rates are being explained. Describing the complete software resting in terms of mathematical equations are called reliability growth model [8]. Measuring or prophesying reliability has always been an innate task. In the case of software's, researches have been done to make this work more scientific rather than intuitive. The present paper adds a bit in this series of researches. Software reliability has continuously been one of the prime concerns of the researchers for a very extended period of time. Optimization techniques inspired by SI have become increasingly popular during the last decade. SI can be applied to several aspects of computer science [9].

The rest of the paper is organized as follows. Section II presents the various researches performed in relation to our suggested work. Section III elucidates the plan, approach, and the advanced technique. Section IV proves and details about the results of our suggested technique, and finally, section V closes our proposed method for selection of software reliability growth model.

2. Related Work

Numerous researches are done in the field of software reliability growth model selection for software reliability checks. Some of the recent researches are given below,

In order to estimate the reliability of software as SRGMs, many software reliability growth models have been suggested in recent past. The proper parameter assessment was tedious since the functions recommended were non-linear in nature. Shanmugam and Florence [10] have discussed an estimation method based on Ant Colony Algorithm in which parameters are assessed. Statistical illustration based on five sets of actual failure data were examined by utilizing existing methods feasible solutions for some of the models, and data sets could not be obtained, whereas; in the proposed method, at least one solution could be obtained.

Due to the growth in demand for software with high reliability and safety, software reliability prediction becomes more and more essential. Software reliability was an important part of software quality. Over the years, many software reliability models have been successfully utilized in practical software reliability

engineering; however, no single model could obtain an accurate prediction for all cases. So in order to improve the accuracy of software reliability prediction Rita G. Al Gargoor *et al.* [11] have proposed a model which combined the software reliability models with the neural networks (NN). To opt for the most excellent structural design of the neural network, Particle swarm optimization (PSO) algorithm have been preferred and applied for learning procedure. The applicability of the proposed model was demonstrated through three software failure data sets. The results exhibited that the proposed model has good prediction capability and more applicable for software reliability prediction.

Software reliability, defined as the likelihood of software to function devoid of malfunction for a precise phase of time in a specific environment. Diverse software reliability growth models have been proposed to envisage the reliability of software. These models help traders to predict the actions of the software before consignment. The reliability was forecasted by approximating the constraints of the software reliability growth models. Traditional techniques like Maximum Likelihood and least Square Estimation finds it difficult to estimate best possible parameters since the model parameters are generally in nonlinear relationships. Parameter estimation of NHPP based reliability models, using MLE and using an evolutionary search algorithm called Particle Swarm Optimization, has been explored by Bidhan and Awasthi [12]. Various stochastic search algorithms have been introduced which have made the task of parameter estimation, more reliable and computationally easier.

Software Reliability Model was categorized into two; one was the static model and the other one was the dynamic model. Dynamic models observed the temporary behavior of debugging process during the testing phase. In Static Models, modeling and analysis of program logic were done on the same code. A Model which describes error detection in software Reliability was called Software Reliability Growth Model. Aggarwal and Gupta [13] have reviewed various existing software reliability models and their failure intensity function and the mean value function. On the basis of this review, a model was proposed for the software reliability having different mean value function and failure intensity function.

For software testing engineers and project managers, it's been a challenge for developing reliability growth models to forecast software reliability to identify and remove errors. Predicting the fault in software helps radically in estimating the software release date and to manage scheme resources. The majority of growth models consider two or three parameters to assess the amassed defects in the testing course. The concern in utilizing evolutionary computation to resolve forecast and modeling problems has grown rapidly. Alweshah *et al.* [14] have explored the use of genetic programming (GP) as a tool to help in building growth models that could accurately predict the number of faults in software early on in the testing process. The proposed GP model was based on a recursive relation derived from the history of measured faults. The developed model was tested on real-time control, military, and operating system applications.

3. Proposed Methodology

Software reliability growth model plays a significant role in identifying the quality and reliability of software being designed. A software reliability growth model is the one in which various measures like failure rate, number failures decreased and other such measures needed for estimating the quality of software are measured. In our proposed system, an efficient software reliability growth model (SRGM) selection for determining the reliability of the software. The reliability model selection criteria are based on the improved computational time and better failure rates. The selection of the model is done by utilizing optimization techniques. Here we have used modified cuckoo search optimization and modified ABC in order to find the effectiveness of the reliability model. Various parameters of reliability models are estimated using the proposed algorithm and are compared with some existing techniques. Here two different optimization approaches are used since we can effectively find the best model using these

algorithms. The structural view of our proposed method is shown in Fig. 1 below,

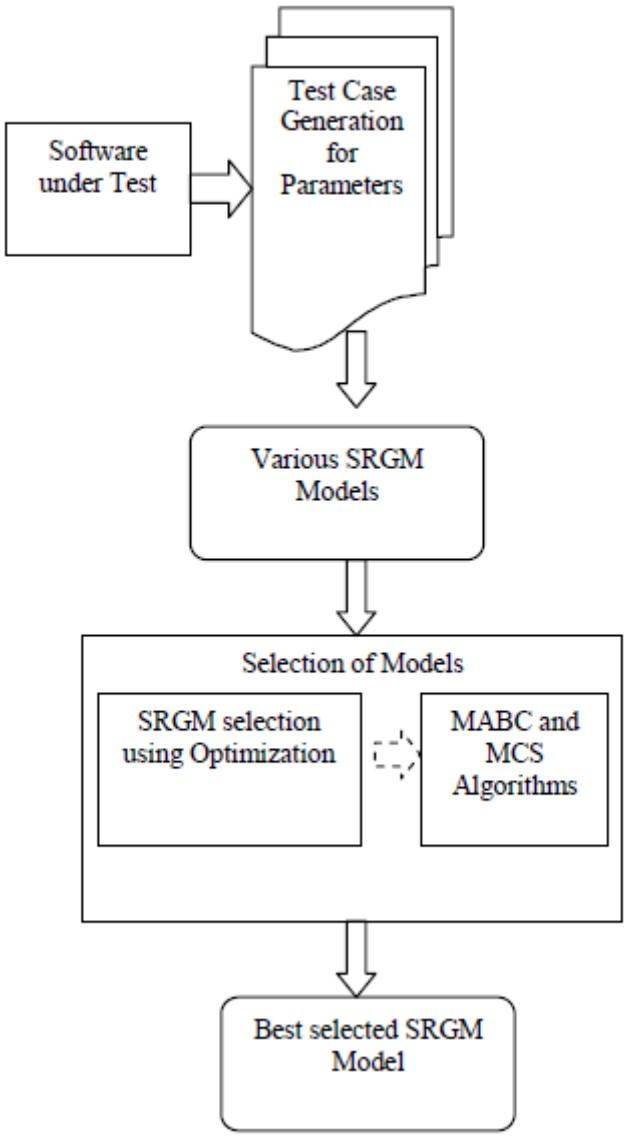

Fig. 1. Architectural representation for proposed SRGM model selection.

3.1. Test Case Generation

Test cases are employed to test all possible combinations in the application, and as well it offers the user just to replicate the steps that were assumed to expose a defect that as identified during the test. Test cases can be charted directly and obtained from use cases. Test cases can as well be obtained from system requirements. Moreover, when the test cases are produced early, Software Engineers can frequently discover ambiguities and inconsistencies in the requirements specification and design documents. It will get down the cost of building the software systems as faults are eradicated early during the life cycle. Test case generation is a method where the test cases are produced not based on an algorithm but based on the one's statement of the application. Classes will be checked and test different inputs will be offered to make sure for the faults in the application.

3.2. Parameter Estimation Criteria

Generally, software reliability growth model success rate determines based on the collected failure data.

In order to estimate the parameters of SRGM, Maximum Likelihood Estimation (MLE) is employed because of its simplicity with the combination of Logistic exponential TEF. Usually, the logistic exponential TEF follows the Non-homogeneous Poisson process (NHPP) and it is highly helped in fault exclusion process.
The estimation of various parameter measures and optimization of the parameter values can result in an enhanced reliability model. The various parameters estimation criteria we used are illuminated as follows,

1) Mean value of number of Faults [15]

$$m(f)=R(1\text{-}\exp(-n*T))/(1+\beta*\exp(-n*T) \quad (1)$$

It is useful to find the mean value of number of faults after finding the total number of failures.

2) Mean value of failures measures the failure rate [15]

$$f(t) = f_i / f_n \quad (2)$$

3) Software reliability measure is used to verify the performance of the particular software component. Also, it helps in software quality prediction. It is defined as [15]

$$R_S(\varphi/T_\pi, C_\pi) = \exp\{-[M_f((T_\pi+\varphi), C_\pi/h) - M_f(T_\pi, C_\pi/h)]\} \quad (3)$$

where

R_S is the software reliability measure

T_Π is the *testing* termination time

C_Π is the *testing* coverage

h represents the total parameter estimate

M_f is the *mean* number of faults detected

3.3. Various Software Reliability Growth Models Used

Numerous researchers proposed various SRGMs to predict the quality of software components. The Several SRGMs we have employed are given in below sections,
chosen some of them such as Yamada Rayleigh Model, Two-Dimensional S-shaped Model, and Huang Logistic Model.

1) Two-Dimensional S-shaped model [16]

Two-Dimensional S-shaped Model evaluates the reliability growth based on the combined effect of testing time and testing coverage during testing. It is defined as

$$F_N = \mu_f - \mu_{fr} \quad (4)$$

where,

μ_f *is* the Mean number of faults detected with respect to the Coverage and time.

μ_{fr} is the mean number of failures with respect to the Coverage and time

2) Yamada Rayleigh Model [16]

The Yamada Rayleigh model estimates the cumulative number of faults in the software application and plots in terms of S-shaped curves. Usually, the failure rate is found using the Rayleigh function. It is expressed as

$$F(t) = f(K, t_m, t) \quad (5)$$

where K is total defects observed, t_m is the maximum peak time, and t is the actual time.

3) Huang Logistic Model [16]

The Huang logistic model uses the logistic functions. Because of simplicity, the logistic exponential Testing effort function is widely used technique in modeling software reliability. The following expression measures the logistic functions

$$L(t) = \sum_{n=1}^{k} \mu_{cf_k} / \mu_{f_k} \tag{6}$$

where

μ_{cf} - mean value of the cumulative number of failures

μ_{f_k} - mean value of failure.

3.4. Proposed SRGM Selection Using Modified Cuckoo Search Algorithm

The Cuckoo search algorithm is an optimization technique which is a biologically inspired algorithm based on the behaviors of the bird cuckoo. In Cuckoo search, the main factor is the levy flight equation employed updation procedure. In our proposed modified cuckoo search algorithm, the levy flight equation is modified by including Gaussian function. The flow chart for modified cuckoo search algorithm as shown in below Fig. 2,

The various phases of the cuckoo search algorithm is given in the below steps,

Step1: Give software application as input

Step2: Compute the Reliability and Fitness values of the software application.

Step 3: Initialization **Phase**

In this phase we initialize the population randomly i.e. m_i, where i=1, 2,…, n.

Step 4: Generation of New Cuckoo Phase

New solution is generated randomly by utilizing levy flights. The worst and the best cuckoo are selected based on the objective function obtained and we neglect the worst case.

Step 5: Fitness Evaluation Phase

The best solution is selected by calculating the fitness value using the expressions given below,

$$Q_M = \frac{Q_{Sel}}{Q_{To}} \tag{7}$$

$$\text{fitness} = \max \text{popularity} = Q_M \tag{8}$$

where,

Q_{Sel} - signifies the selected population

Q_{To} - represents the total population

Step 6: Updation Phase

In updation phase the solution is optimized by the levy flights. The nest is selected randomly depending on the quality of the new solution. The replacement of solution is made if the previous solution is inferior to the obtained new solution or vice versa. The expression given below shows the levy flight equation for finding out the best solution:

$$z_i^* = z_i^{(t+1)} = z_i^{(t)} + \sigma \oplus Levy(p) \tag{9}$$

The above normal levy flight equation is modified by incorporating the gauss distribution which is given in below expressions,

$$z_i^{*} = z_i^{(t+1)} = z_i^{(t)} + \sigma \oplus \lambda_g \tag{10}$$

where,

$$\lambda_g = \lambda^{*} \ \exp(-\rho C) \tag{11}$$

$\lambda^{*}{}_0, \rho$ - represents the constants

C – Symbolizes the current generation

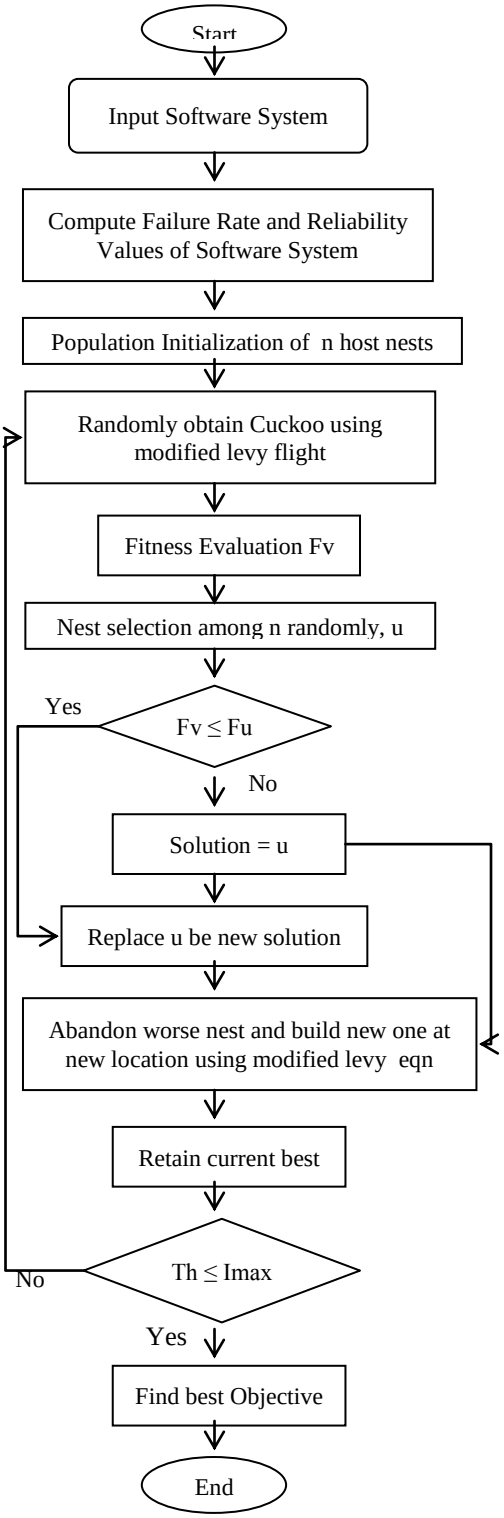

Fig. 2. Flow diagram for modified cuckoo search algorithm.

Step 7: Worst Nest rejection Phase

In worst nest rejection phase the worst cases are ignored based on the fitness value and new solutions are selected. And the selected best solutions are ranked depending upon their fitness value

Step 8: Termination

.The above process will be continued till the termination criteria is achieved. Thus we utilized the MCS algorithm for selecting the SRGM which can help in effectively estimating the software reliability.

In our proposed system we employ another optimization process to select the better model. We can compare the outcome of the two optimization techniques to choose the better model. The algorithm utilized here is modified ABC algorithm, and the process is explained in the below section.

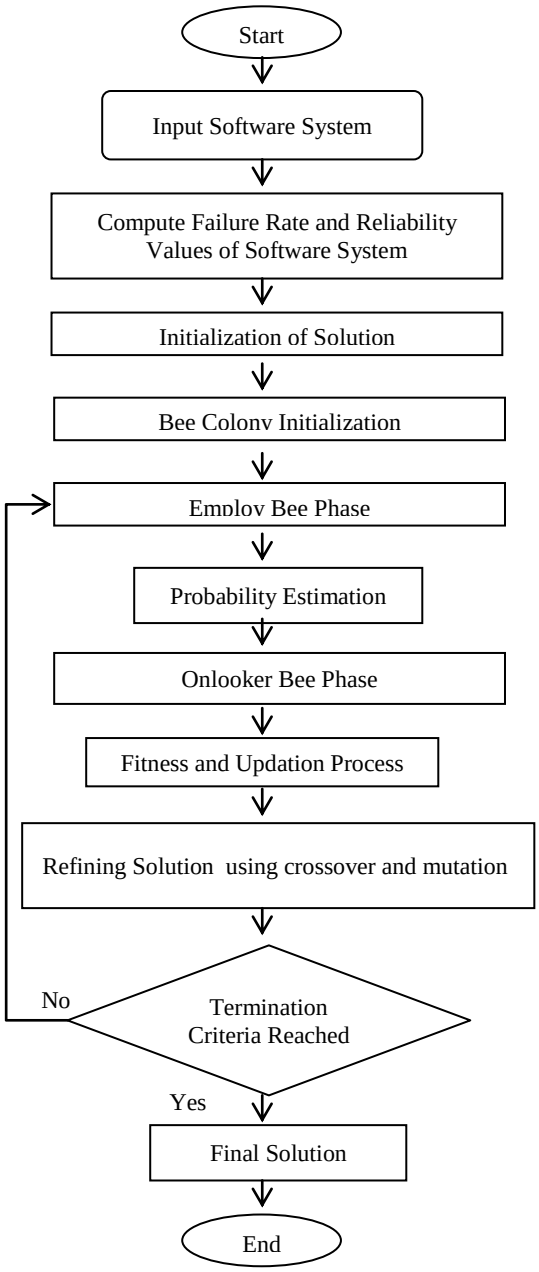

Fig. 3. Flow diagram for Modified ABC.

3.5. Proposed Modified Artificial Bee Colony for Optimization

In an ABC model, a food source position refers to a possible solution for the optimization problem,

whereas nectar quantity of a food source refers to the quality (fitness) of the respective solution. The objective of bees is set as to determine the best solution [17]. Each employed bee shares the information with an onlooker bee and flies back to the food source, which was visited by it in the previous wandering. This process is undertaken since the memory keeps the record of the food source. The employed bee selects a new food source using the visual knowledge about the neighborhood of the one stored in the memory and assesses the nectar amount [18]. The normal ABC contains three phases like employee bee phase, onlooker bee phase, and scout bee phase. In our proposed scheme, instead of the scout bee phase, we incorporate the genetic operators like crossover and mutation for the better optimization process. The flow diagram for modified ABC as shown in Fig. 3 above. The procedure for our proposed optimization is explained below,

3.5.1. Employee bee phase

The colony of artificial bees is home to three categories of bees such as the employed bees, onlooker bees and the scout bees. A bee hanging around on the dance area for taking a decision to select a food source is known as an onlooker bee. Whereas a bee moving towards the food source visited earlier is labelled as an employed bee. Further, a bee entrusted with the task of performing arbitrary search is termed as a scout bee. One half of the colony is occupied by the employed artificial bees and the other half is inhabited by the onlookers. In respect of each food source, there is only one employed bee. Thus, the number of employed bees matches exactly with the number of food sources around the hive. The employed bee whose food source is used up by the employed and onlooker bees emerges as a scout. At the initialization phase, a set of food source locations are arbitrarily chosen by the employed bees and their nectar quantities are estimated like E_i with m solutions, where each solution is the food source position and Psz is the population size is generated. The solution representation can be given as S_i *where* $1 \leq i \leq m$ and M is the number of parameters to be optimized Subsequently, these bees come into the hive and exchange the nectar data of the sources with the onlooker bees staying on the dance area within the hive.

3.5.2. Onlooker bee phase

In this stage, the selection of the food sources by the onlookers after getting the data from the employed bees and generation of novel solution are performed. The onlooker bee favors a food source area based on the nectar data supplied by the employed bees on the dance area. Along with the enhancement in the nectar quantity of a food source the probability of selection of the corresponding food source by an onlooker increases. When the nectar of a food source is thrown away by the bees, a fresh food source is arbitrarily decided by a scout bee and substituted with the discarded one. An artificial onlooker bee probabilistically generates an alteration on the location (solution) in her recollection for discovering a fresh food source and assesses the nectar quantity (fitness value) of the fresh source (new solution). The probability of food source selection by onlooker bee is expressed as below,

$$P_{fs} = \frac{F_j}{\sum_{i=1}^{n} F_i} \tag{12}$$

where,

F_j=F_i refers to fitness of the solution and

n refers to the number of food sources which is equal to the number of employed bees.

On reaching the chosen zone, onlooker selects a fresh food source in the vicinity of the one in the recollection according to the visual data, which is invariably dependent on the analysis of food source locations. When the nectar of a food source is thrown away by the bees, a fresh food source is arbitrarily decided by a scout bee and substituted with the discarded one.

Let the old location be characterized by $z_{j,k}$ and the new position by $y_{j,k}$, which is expressed by the equation as shown below, which manages the generation of a neighbour food source location around $y_{j,k}$ and the renovation, characterizes the analysis and contrast of the neighbour food locations visually by the bee.

$$z_{j,k} = y_{j,k} + \phi_{j,k}(y_{j,k} - y_{i,k}),\ i \neq j \tag{13}$$

where,

$$k = \{1,2,\ldots,n\}$$
$$i = \{1,2,\ldots,N\}$$

$\varphi_{j,i}$ is an arbitrary number in the range[−1, 1].

The above equation can be reformulated by rearranging the position values which is shown in the eqn below,

$$z_{j,k} - y_{j,k} = \varphi_{j,k}(y_{j,k} - y_{i,k}) \tag{14}$$

For the above equation we define the time domain form by substituting certain factors for each position value like $y_{j,k}$ as X_l when $z_{j,k}$ is taken as X_{l+1}.

$$X_{l+1} - X_l = \varphi_{j,k}(y_{j,k} - y_{i,k}) \tag{15}$$

The above eqn can be reformed since $X_{l+1} - X_l$ is the discrete version of the derivative of order $\alpha = 1$,

$$D^{\alpha}[X_{l+1}] = \varphi_{j,k}(y_{j,k} - y_{i,k}) \tag{16}$$

After the onlooker bee phase we have incorporated the genetic operators like crossover and mutation in order to refine the optimization process. The process is explained in the section below,

3.5.3. Crossover

After selecting the individuals, the next step is the crossover where two parents are made to mate each other. In the crossover process, the particles from different position are interchanged to generate better offspring so that the particle can generate better fitness value compared to the parent particle. Select the particle with the best fitness value, reinitialize its position. Along with this evaluate the particle with the worst fitness value, whether its new position is acceptable, if it is within an acceptable range then update its position or else a new position is assigned to the particle randomly in its neighbourhood and then renew the position and velocity of other particles. In our proposed technique we employ two point crossover techniques with the crossover rate of CO_{Rate}. The eqn 17 and 18 shows the crossover point selection. The genes in between the two points c_{ovr1} and c_{ovr2} are interchanged between the parent chromosomes and so $N_p/2$ children chromosomes are obtained. The crossover points c_{ovr1} and c_{ovr2} are determined as follows

$$c_{ovr1} = \frac{\left|S_f^{(i)}\right|}{3} \tag{17}$$

$$c_{ovr2} = c_{ovr1} + \frac{\left|S_f^{(i)}\right|}{2} \tag{18}$$

Pseudo code for Crossover

```
function C = CrossOver(C,Psize)
for t = 1 : size(C,1)
%    tmp1 = C(t,1);
%    tmp2 = C(t+1,1);
%    C(t+1,1) =  randi([1,Psize]);
 C(t,1) =  randi([1,Psize]);
end
```

3.5.4. Mutation

After crossover, a new set of populations is produced. In order to provide individuality to each chromosome, mutation operation is performed where we replace any value with a new value to form a new individual.

Pseudo code for Mutation

```
function M = Mut(M,Psize)
r = randi([1 2],1);
for t = 1 : r
 r1 = randi([1 size(M,2)],1) ;
M(t,r1) =randi([1,Psize]);
end
```

Thus using the proposed MABC the parameters are optimized and the models are selected. The selected models using MCS and MABC are the compared to find out the better model. Along with these proposed algorithms, we compare the performance of Genetic Algorithm as well.

4. Results and Discussions

Table 1. Cumulative Number of Failures before and after Optimization at Various Time Intervals

Time (sec)	Cumulative number of failures	
	Before Optimization	After optimization
1	4.23	0.38
5	3.15	0.25
10	6.65	0.19
20	5.21	0.29
25	5.01	0.13

Normally, SRGMs are means that is utilized in order to determine the quantifiable temperament of software, to extend test status, to plan status and to observe reliability values. The quality of software can be estimated based on the prediction of software reliability. In our proposed scheme we have developed a novel technique for software reliability growth model selection, where we utilize optimization techniques to select the models based on the failure rate of the software. This prediction method proves to be more efficient in terms of selection of SRGMs as it utilizes double optimization scenario. The implementation of the proposed scheme is done using JAVA and Netbeans tools and the software's being used as inputs are banking application software, hospital healthcare management systems, and library management software. The various measures that we employ in our proposed system to show the effectiveness of the technique

are mentioned below.

For various time instants, the aggregate number of failure prior and after optimization are presented in Table 1 given below.

The graphical analysis of cumulative number of failure for the above table values is shown in Fig. 4.

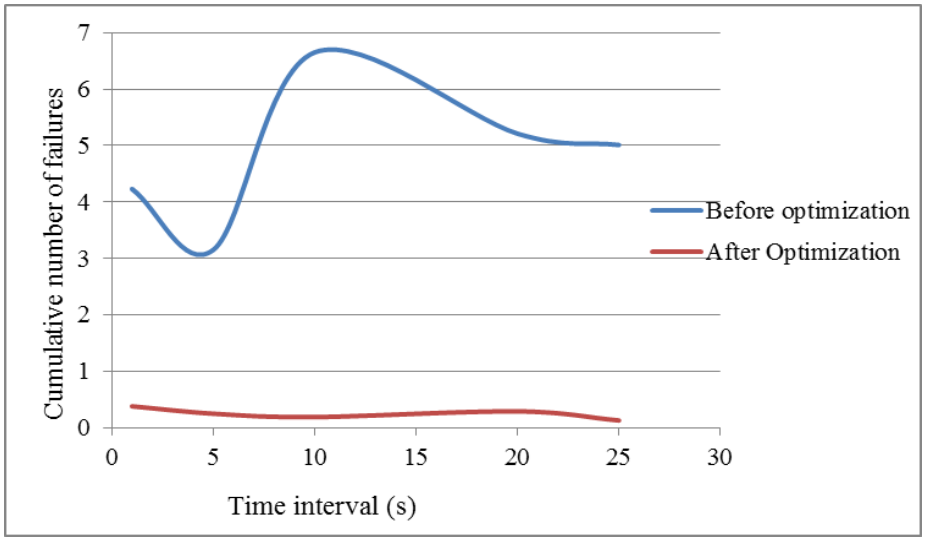

Fig. 4. Graphical depictions for comparison of cumulative number of failures before and after optimization.

4.1. Application Examples

DS1: We have proved the effectiveness of our proposed system by considering various application examples. First among the application example is the banking application software. The different parameters like the number of failures, the cumulative number of faults, the mean value of number of faults, and reliability are measured for the banking application software. The proposed MCS and MABC have delivered better results. Table 2 given below shows the decreased number of failures obtained for various models for banking application software.

Table 2. Reduced Failure Rate for Different SRGM Models at Different Time Intervals

Time Interval (sec)	Number of failures Reduced		
	Yamada Rayleigh Model	Huang Logistic Model	Two-Dimensional S- Shaped Model
1	2.1345	2.0213	8
5	4.1526	3.9832	12
10	7.3265	7.1212	13
20	9.1235	8.2312	9
25	10.2301	9.9621	15

The graphical analysis of the values for failure rate decreased at various time instant for different SRGM models are plotted as shown in the Fig. 5 given below

DS 2: The second application example we employed is the library management system software. The different parameters like the number of failures, the cumulative number of faults, the mean value of number of faults, and reliability are measured for library management application. The proposed MCS and MABC have delivered better results. Table 3 given below shows the decreased number of failures obtained for various models for library management system software.

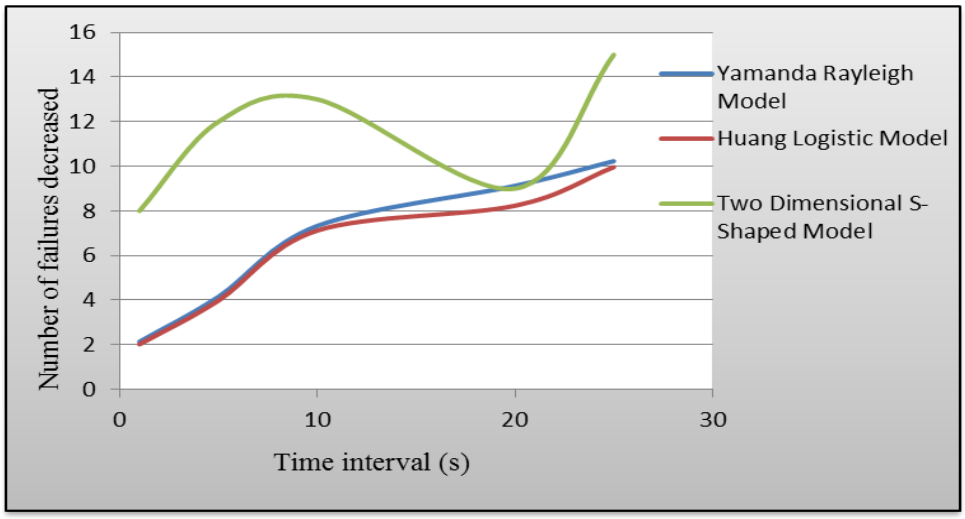

Fig. 5 Graphical representation of failure rate for different SRGMs at different time intervals.

Table 3. Reduced Failure Rate for Different SRGM Models at Various Time Intervals

Time Interval (s)	The Number of failures Decreased		
	Yamada Rayleigh Model	Huang Logistic Model	Two-Dimensional S- Shaped Model
1	12.1563	12.9621	19
5	7.4123	9.2315	15
10	5.2314	6.3251	10
20	4.2135	2.3265	12
25	8.2314	5.3658	9

The graphical analysis of the values for failure rate decreased at various time intervals for different SRGM models are plotted as shown in Fig. 6 given below

DS 3: The final example application is hospital healthcare management system software. The different parameters like the number of failures, the cumulative number of faults, the mean value of number of faults, and reliability are measured for library management application. The proposed MCS and MABC have delivered better results. Table 4 given below shows the decreased number of failures obtained for various models for hospital management system software.

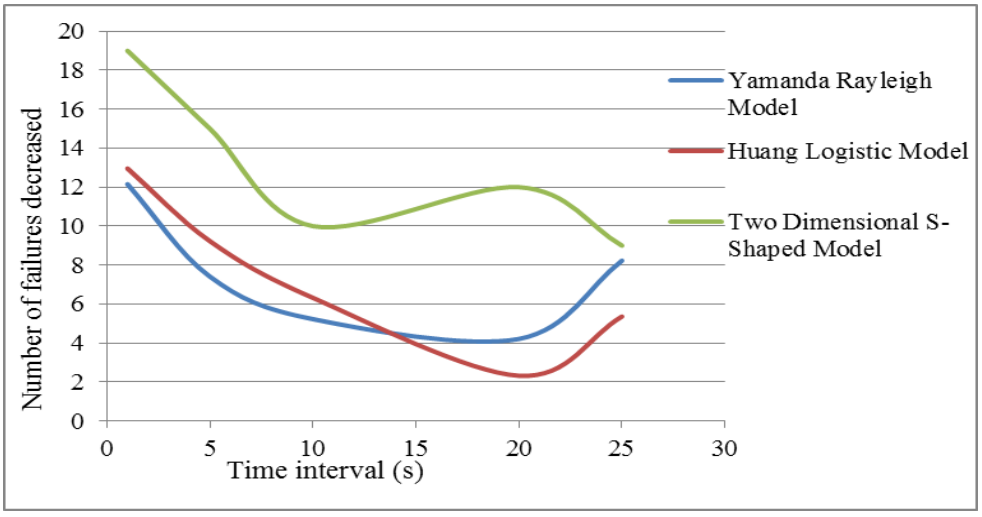

Fig. 6. Graphical representation of failure rate for different SRGMs at various time intervals.

Table 4. Reduced Failure Rate for Different SRGM Models at Various Time Instants

Time Interval (sec)	Number of failures Decreased		
	Yamada Rayleigh Model	Huang Logistic Model	Two-Dimensional S- Shaped Model
1	16.2314	15.2314	18
5	12.3245	11.2312	14
10	9.2356	8.2325	9
20	5.2134	2.3265	11
25	6.2154	5.3658	7

The graphical analysis of the values for failure rate decreased at various time instant for different SRGM models are plotted as shown in the Fig. 7given below

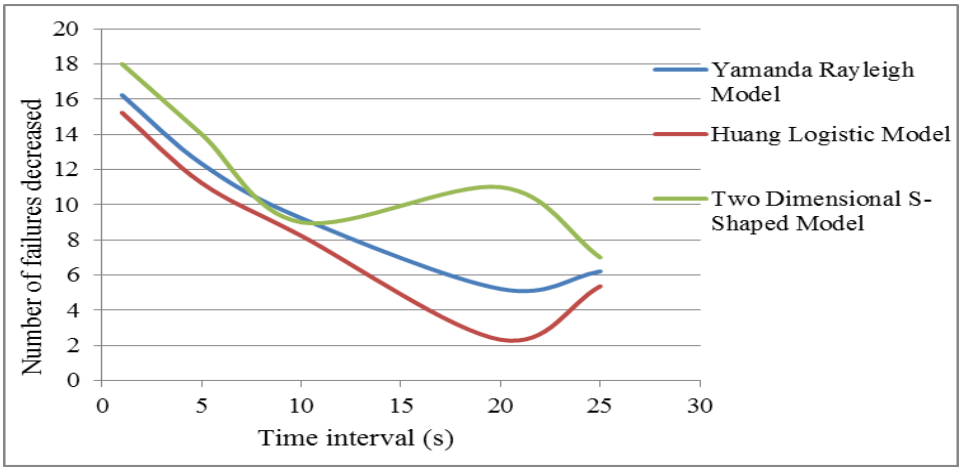

Fig. 7. Graphical representation of failure rate for three different models at various time intervals.

4.2. Comparison of SRGM's Using MCS with MABC and GA

Table 5 given below shows the failure numbers that are generated before and after applying optimization techniques. From the table, we can find that the number of failures has bettered when optimization technique are applied to the system.

Table 5. Number of Failures at different Time Intervals before and after Optimization Using MCS, MABC, and GA Algorithms

Time (s)	No of failures			
	Before Optimization	After optimization (MCS)	After optimization (MABC)	After optimization (GA)
1	19	13	15	17
5	23	16	21	22
10	41	29	34	38
20	30	19	28	29
25	35	23	32	33

The Fig. 8 given below shows the representation of the number of failures obtained before and after optimizations in graphical form. The proposed approach has delivered better results when compared to existing techniques.

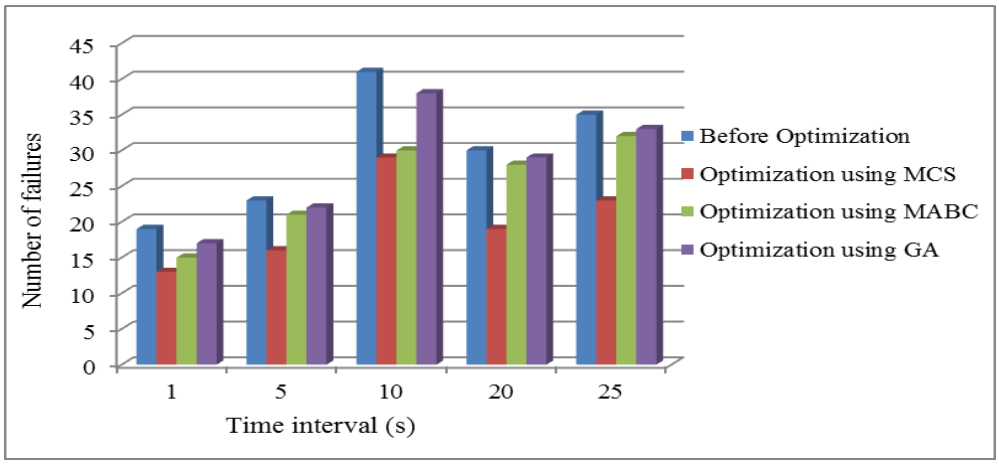

Fig. 8. Comparison of no of failures before and after optimizations at various time intervals.

The reliability values obtained for the software is shown in Table 6 below. From the values obtained, we can infer that once optimization is done, the reliability of the software gets improved, and our proposed algorithms display better reliability values than other algorithms.

Table 6 Reliability at Different Time Intervals before and after Optimization Using MCS, MABC, and GA Algorithms

	Reliability			
Time (s)	Before Optimization	After optimization (MCS)	After optimization (MABC)	After optimization (GA)
1	0.3216	1.2352	0.9821	0.6524
5	0.2923	1.9231	1.2352	0.8962
10	0.3621	1.5231	1.0124	0.7542
20	0.4962	1.6214	1.3214	0.9623
25	0.2536	1.1132	1.0035	0.8213

The graphical representation of the comparison of reliability value using proposed MCS algorithm, MABC and GA are shown in Fig. 9 below. The graph shows that our proposed method delivers better reliability when compared to that of the existing MABC, GA process.

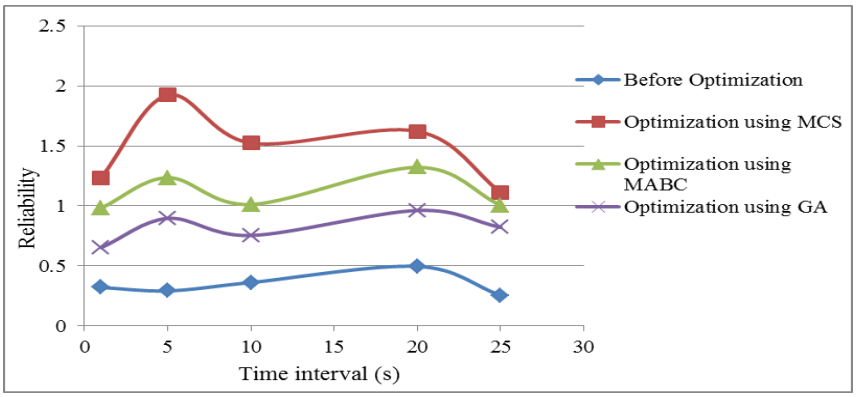

Fig. 9. Reliability comparison before and after optimizations at different time intervals using different optimization techniques.

The decreased failure rate for software reliability growth model with different optimization technique is shown in Table 7 below,

Table 7. Decreased Failure Rate for Software Reliability Growth Models Using different Optimization Techniques

Methods	The Number of failures Decreased		
	Two-Dimensional S-shaped Model	Yamada Rayleigh Model	Huang Logistic Model
MCS	19	18	18
MABC	16.2314	15.2314	14.2135
GA	12.9621	12.1214	11.2389

The graphical representation of reduced failure rate for software reliability growth models using different optimization techniques is shown in Fig. 10 below.

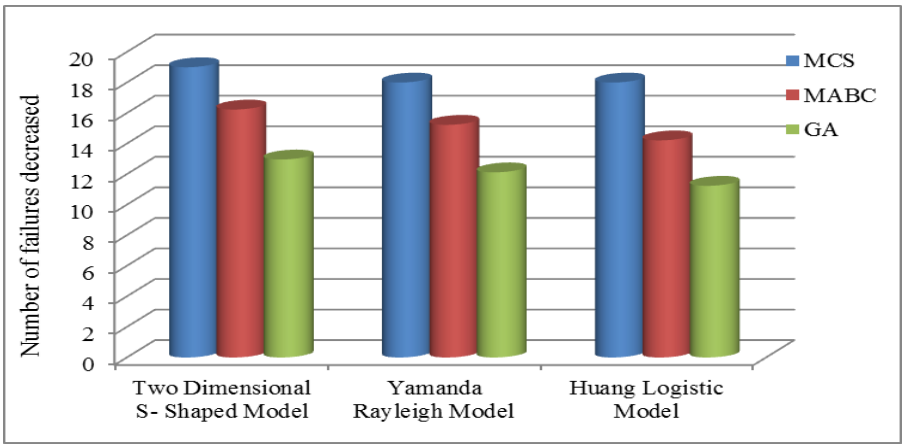

Fig. 10. Comparison of various software reliability growth models using MCS, MABC and GA optimization techniques.

Based on the failure rate for different models the efficiency is estimated which is tabulated in the below Table 8.

Table 8. Efficiency for Different Software Reliability Growth Models

S.No	Models	Efficiency (%)
	Two-Dimensional S-Shaped Model	79.23
2	Yamada Rayleigh Model	69.11
3	Huang Logistic Model	58.21

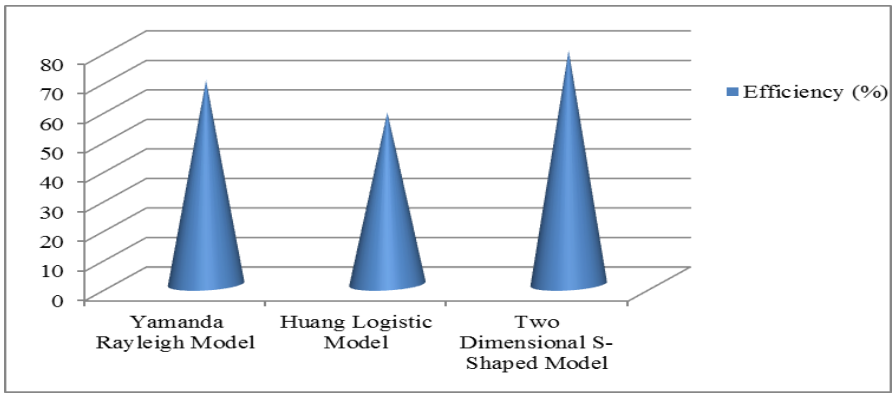

Fig. 11. Graphical representation of efficiency for different SRGM like two-dimensional S-shaped, Yamada Rayleigh model, and Huang logistic model.

The graphical analysis of the performance of the various SRGMs in terms of efficiency is displayed in Fig. 11 above.

5. Conclusion

In this work, we have designed an innovative technique to select the software reliability growth model (SRGM). Here we have employed MCS and MABC optimization algorithms for the selection of SRGM. The selection of better model can assist in obtaining reliability of software. The major consideration for selecting the model in our proposed technique is the failure rate. The utilization of MCS and MABC have improved the selection rate as the optimization techniques are best suited for reducing the failure rate. The advantage of the proposed method is clear from the obtained results. In the proposed approach the major factor in selecting the reliability growth model is its efficiency, and the results prove that our proposed system has delivered better efficiency in selecting SRGM when compared with other algorithms.

References

[1] Kapur, P. K., Anshu, G., & Jha, P. C. (2007). Reliability growth modeling and optimal release policy under fuzzy environment of an n-version programming system incorporating the effect of fault removal efficiency. *International Journal of Automation and Computing, 4(4)*, 369-379.

[2] Singh, V. B., Kapur, P. K., & Mashaallah, B. (2012). Open source software reliability growth model by considering change – point. *International Journal of Information Technology, 4(1)*.

[3] Alaa, S. (2006). Reliability growth modeling for software fault detection using particle swarm optimization. *Proceedings of the IEEE* International *Conference on Evolutionary Computation* (pp. 3071-3078).

[4] Latha, S., & Lilly, F. (2013). Enhancement and comparison of ant colony optimization for software reliability models. *Journal of Computer Science, 9(9)*, 1232-1240.

[5] Carina, A. (2007). A replicated empirical study of a selection method for software reliability growth models. *Journal of Empirical Software Engineering*, *12(2)*, 161–182.

[6] Sultan, A. (2011). Development of software reliability growth models for industrial applications using fuzzy logic. *Journal of Computer Science, 7(10)*, 1574-1580.

[7] Garima, B., & Kulvinder, S. (2015). Comparative study of cuckoo search and simulated annealing technique on SRGM exponential models. *International Journal of Advanced Research in Computer Science and Software Engineering, 5(3)*.

[8] Shaik, M. R., & Shaheda, A. (2010). Software reliability growth model with gompertz TEF and optimal release time determination by improving the test efficiency. *International Journal of Computer Applications*, *7(11)*, 0975–8887.

[9] Neha, G., & Tarun, A. (2014). A review on particle swarm optimization for software reliability. *International Journal* of *Emerging Trends and Technology in Computer Science*, *3(3)*.

[10] Latha, S., & Lilly (2012). A comparison of parameter best estimation method for software reliability models. *International Journal of Software Engineering and Applications,* 3(5).

[11] Rita, G. A. G., & Nada, N. S. (2013). Software reliability prediction using artificial techniques. *IJCSI International Journal of Computer Science Issues*, *10(4)*.

[12] Karambir, B., & Adima, A. (2014). A review on parameter estimation techniques of software reliability growth models. *International Journal of Computer Applications Technology and Research, 3(4)*, 267-272.

[13] Gaurav, A., & Gupta, V. K. (2014). Software reliability growth model. *International Journal of Advanced Research in Computer Science and Software Engineering*, *4(1)*.

[14] Mohammed, A., Walid, A., & Hamza, A. (2015). Evolution of software reliability growth models: A comparison of auto-regression and genetic programming models. *International Journal of Computer Applications*, *125(3)*, 0975–8887.

[15] Rao, K. M., & Anuradha, K. (2014). An efficient method for parameter estimation of software reliability

growth model using artificial bee colony optimization. *Proceedings of the International Publishing Switzerland*, 765-776.

[16] Mallikharjuna, R. K., & Anuradha, K. (2015). An efficient method for software reliability growth model selection using modified particle swam optimization technique. *International Review on Computers and Software*, 1169-1178.

[17] Dervis, K., & Celal, O. (2010). Fuzzy clustering with artificial bee colony algorithm. *Journal of Scientific Research and Essays*, *5(14)*, 1899-1902.

[18] Dervis, K., & Bahriye, A. (2009). A comparative study of artificial bee colony algorithm. *Journal of Applied Mathematics and Computation*. 108–132.

Mallikharjuna Rao K. obtained his bachelor's degree in information science and technology engineering from University of Acharya Nagarjuna University, Guntur, Andhra Pradesh, India in 2004. Then he obtained his master's degree in computer science from Jawaharlal Nehru Technological University of Hyderabad, India in 2010, and pursuing Ph.D. in computer science and engineering majoring in software engineering-software reliability engineering at Jawaharlal Nehru Technological University Hyderabad, Andhra Pradesh, India. He has more than 10+ years of teaching experience in Engineering Institutions. Currently, he is working as assistant professor in the Department of Information Technology, GITAM University, Visakhapatnam, Andhra Pradesh, India. His specializations include software reliability, and software testing. His current research interests are studying about software reliability growth models. He is a member of MIAENG, AMIE, India.

Kodali Anuradha received the master's degree M.Sc. in mathematics from Nagarjuna University, Guntur, Andhra Pradesh, India, and a master's degree M.Tech in computer science from BITs, Ranchi, India, and a Ph.D in mathematics from JNTU Hyderabad, Andhra Pradesh, India, and a Ph.D in computer science and engineering from JNTU Anantapur, Andhra Pradesh, India in 1987, 2001, 2006, and 2011 respectively. Currently, she is working as a professor at the School of Computing at GRIET, Hyderabad, India. She has more than 25 years of Teaching and 6+ years of Research experience. She is currently guiding 6+ research scholars. Her research interests are software engineering, data mining, image processing, computer networks, and network security.