International Journal on Recent and Innovation Trends in Computing and Communication
Volume: 6 Issue: 4

ISSN: 2321-8169
106 -112

Predicting Software Reliability Using Ant Colony Optimization Technique with Travelling Salesman Problem for Software Process – A Literature Survey

[1]D. Hema Latha , [2]Prof. P. Premchand

[1]Research Scholar, Dept of Computer Science, Rayalaseema University, Kurnool, Andhra Pradesh, India

[2]Professor, Dean, Faculty of Informatics, Dept of Computer Science and Engineering, UCE, Osmania University, Hyderabad, TS, India

Abstract:-Computer software has become an essential and important foundation in several versatile domains including medicine, engineering, etc. Consequently, with such widespread application of software, there is a need of ensuring software reliability and quality. In order to measure such software reliability and quality, one must wait until the software is implemented, tested and put for usage for a certain time period. Several software metrics have been proposed in the literature to avoid this lengthy and costly process, and they proved to be a good means of estimating software reliability. For this purpose, software reliability prediction models are built. Software reliability is one of the important software quality features. Software reliability is defined as the probability with which the software will operate without any failure for a specific period of time in a specified environment. Software reliability, when estimated in early phases of software development life cycle, saves lot of money and time as it prevents spending huge amount of money on fixing of defects in the software after it has been deployed to the client. Software reliability prediction is very challenging in starting phases of life cycle model. Software reliability estimation has thus become an important research area as every organization aims to produce reliable software, with good quality and error or defect free software. There are many software reliability growth models that are used to assess or predict the reliability of the software. These models help in developing robust and fault tolerant systems.

In the past few years many software reliability models have been proposed for assessing reliability of software but developing accurate reliability prediction models is difficult due to the recurrent or frequent changes in data in the domain of software engineering. As a result, the software reliability prediction models built on one dataset show a significant decrease in their accuracy when they are used with new data. The main aim of this paper is to introduce a new approach that optimizes the accuracy of software reliability predictive models when used with raw data. Ant Colony Optimization Technique (ACOT) is proposed to predict software reliability based on data collected from literature. An ant colony system by combining with Travelling Sales Problem (TSP) algorithm has been used, which has been changed by implementing different algorithms and extra functionality, in an attempt to achieve better software reliability results with new data for software process.

The intellectual behavior of the ant colony framework by means of a colony of cooperating artificial ants are resulting in very promising results.

Keywords*: Software Reliability, Reliability predictive Models, Bio-inspired Computing, Ant Colony Optimization technique, Ant Colony Optimization, Travelling Salesman problem and Quality Models.*

*****

I. LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

Considering the importance of software reliability in software engineering, its prediction becomes a very fundamental issue. Machine learning and soft computing techniques [18],[19],[20] have been leading the statistical techniques in last two decades as far as their applications to software engineering are concerned. The related study of this paper is as follows:

1).Shrivastava and Shrivastava (2014) propose applying GP with Artificial Bee Colony (ABC). Experiments are conducted on NASA data sets KC1 and PC1 as well as the mushroom data set. The hybrid approach is compared to neural gas, support vector machines (SVM) and symbolic regression. The accuracy on testing is computed using 10-fold cross validation. In KC1 and PC1, SVM out-beats all other approaches by 4.65% and 2.19% respectively. The hybrid GP-ABC obtains the best results on the mushroom data set and outperforms others by 2.9% on average.

2).A hybrid of ACO and GA is proposed by Maleki, Ghaffari, and Masdari (2014) to optimize software cost estimation utilizing the KLoC metric only. Ten NASA data sets are used as benchmarks. The approach proves more efficient than the COCOMO model (Boehm, 1981) according to the Magnitude of Relative Error (MRE). 2.3.7 Artificial Bee Colony . Very little work exists using ABC for software quality prediction models. Moreover, to the best of the author’s knowledge, no ABC approach has yet been proposed for prediction of stability of classes in an OO system.

3).Di Martino, Ferrucci, Gravino, and Sarro (2011) propose a hybrid of GA and Support Vector Machines (SVM) for inter-release fault estimation using the metrics of Chidamber and Kemerer (1994). The GA's objective is to find a suitable SVM parameter setting. The approach is compared to six famous machine learning techniques namely Logistic Regression, C4.5, Naïve Bayes, Multi-Layer Perceptrons, K-Nearest Neighbor (K-NN) and Random Forest. Ten fold cross validation is applied. The measurements for comparison are accuracy, precision, recall and F-measure. When tested on the jEdit PROMISE data set from (Promise), the hybrid is found to be very effective especially for inter-release fault estimation.

4).Sarro, Di Martino, Ferrucci, and Gravino (2012) extend the work of the hybrid approach proposed by Di Martino, Ferrucci, Gravino, and Sarro (2011). In this study, the approach is not only tested on six machine learning techniques, but also on variants of SVM. It is validated on several PROMISE data sets from (Promise): Log4j (versions 1.0, 1.1, 1.2), Lucene (versions 2.0, 2.2 and 2.4), POI (versions 1.5, 2.0, 2.5 and 3.0), Xalan (versions 2.4, 2.5, 2.6 and 2.7), and Xerces (versions init, 1.2, 1.3 and 1.4). The hybrid is shown to be effective even though it requires more runtime than other models.

5). Farshidpour and Keynia (2012) propose training a multi-layer perceptron (MLP) neural network using ABC for software fault prediction. Their approach is compared to MLP with back propagation. The authors conclude that when the proper parameters are set to ABC, the neural network can be effectively trained. Two approaches are compared: MLP trained using ABC (MLP-ABC) versus MLP trained using back propagation (MLP-BP). Experiments are conducted on the NASA data sets CM1, KC1, KC2. In terms of testing accuracy and testing precision, MLP-ABC out-performs MLP BP on average by 1.4% and 1.8% respectively. Kayarvizhy, Kanmani, and Uthariaraj (2014) propose optimizing the prediction accuracy of artificial neural networks (ANN). They propose training the ANN by using swarm intelligence techniques, namely PSO, ACO, ABC and firefly. The aim is to obtain the best parameters for the ANN, such as the number of input neurons, number of hidden layers and hidden neuron, number of output neuron, weights, etc. The authors compare ANN-PSO, ANN-ACO, ANN-ABC and ANN-Firefly on the following NASA data sets: Arc, Camel (versions 1.0, 1.2, 1.4 and 1.6), Intercafe and Tomcat. The authors conclude that the best approach is ANN-PSO which reaches the best results in 7 out of the 8 data sets. ANN-ABC is ranked second best, reaching best results in 3 out of the 8 data sets.

6). Mohanty et al. (2010) [1], [2] the most recent state-of-the-art review, by, justifies this declaration. In this paper, the authors employed machine learning techniques [1], specifically, Back propagation trained neural network (BPNN), Group method of data handling (GMDH), Counter propagation neural network (CPNN), Dynamic evolving neuro–fuzzy inference system (DENFIS), Genetic Programming (GP), TreeNet, statistical multiple linear regression (MLR), and multivariate adaptive regression splines (MARS), to accurately forecast software reliability.

7). Rappos and Hadji constantinou (2004), in order to design two-edge connected flow networks, use two types of ant colonies sharing information about their pheromone levels. This problem is about configuring a network in order to satisfy demand nodes, provided that an extra arc is considered to keep the network flowing in the case that one of the arcs in the network fails.
The solution for this problem is constructed in two phases, each of which solved by a different type of ants. One ant colony is inhabited by flow ants and the other colony by reliability ants.
The number of flow ants is the same as the number of demand nodes and, although they all start constructing their solution from the source node, each ant is assigned to reach just one specific demand node. When all flow ants have constructed their partial solutions, reaching their demand node destination, the network is created. The next step involves the reliability ants whose objective is to decide upon the extra arc, called reliability arc, to be added to the solution. For every flow ant a reliability ant is created and associated with each arc visited by the flow ant. Therefore, for each flow ant there is a set of reliability ants, as many as arcs in the solution of the flow ant. The objective of a reliability ant is to find an alternative path from the root node to the same destination node of the flow ant as long as it does not use a particular arc, from the ones used in the solution of the flow ant. This ACO algorithm provides a single feasible solution at each iteration, which is only entirely defined when all partial solutions of the flow ants have been assembled together, and the extra arc found by the reliability ants is identified.

8).Baykasoglu et al (2006) solve a dynamic facility layout problem, where each ant has to decide, for each period t, the location of the n departments considered. The authors use a string with size t × n to represent the final solution, where the first n consecutive values identify the department locations for the first period; the second n consecutive values give the locations for the second period, and so on.

Therefore, to construct a solution, all an ant has to do is to choose t × n elements of the type department location, accordingly to the pheromone levels, and provided that, within a time period, no department location is repeated, thus guaranteeing the construction of a feasible solution.

9).Crawford and Castro (2006), Solved partitioning and covering problems by ACO algorithms. In this case, given a set of columns and rows, the objective is to choose a subset of columns covering all rows while minimizing cover costs. The solution is represented by a subset of columns. This implies a different approach, from the ones we have been mentioning before, because the solution components are represented by nodes and not by arcs, a fact that simplify the calculations. The construction is straightforward. Each ant starts with an empty set of columns. Then, the ant adds columns one at the time, based on pheromone values, until all rows are covered. Solutions constructed in this way, can be unfeasible in the partitioning case because a row may be covered by more than one column. That is why post processing procedures, that will try to eliminate redundant columns, are added afterwards in order to turn unfeasible solutions into feasible ones.

10).Chen and Ting (2008) The Single Source Capacitated Facility Location Problem deals with the location of a set of facilities, with a limited capacity on the supply, and the allocation of a single facility to each customer so as to satisfy customer demands and minimize total costs propose an algorithm to solve it which integrates an Ant Colony System with two types of ants, location ants and assignment ants.

11). The first work to propose Genetic Programming (GP) in software engineering in general is that of Khoshgoftaar, Evett, Allen, and Chien (1998) and Evett, Khoshgoftaar, Chien, and Allen (1998). GP approach is introduced specifically for reliability enhancement of software modules. The GP is based on definitions by Koza (1994). The metrics used are related to lines of code, operator and cyclomatic complexity (McCabe, 1976). The metrics are proved to be associated with reliability estimation according to Fenton and Pfleeger (1993) . In this study, the evaluation of the GP relies on Pareto’s law which implies that “20% of the modules will typically account for about 80% of the faults.” The approach is found to be robust as opposed to random model when tested on two large industrial projects: Ada-written Command Control and Communications System (CCCS) and Pascal-written legacy telecommunication system.

Another GP approach is proposed by Liu and Khoshgoftaar (2001) to predict fault-proneness and change-proneness using large Windows-based applications written in C++ programming language. Accounting for over-fitting, GP only uses a random subset selection of the data. GP is then tested on the entire data set given product and process metrics. The proposed GP is novel in the sense that it integrates prior probability as well as misclassification cost into its fitness function. It is demonstrated that, when compared to logistic regression, GP achieves better results in terms of Type-I and Type II errors. GP also obtains an average of 9% improved accuracy over logistic regression.

12).Yongqiang and Huashan (2006) propose GP for predicting software reliability based on Mean Time Between Failures (MTBF). When compared to some statistical
5 This is simply a variation of the GP proposed by (Montana, 1995) models as well as Neural Networks, GP gives more accurate results and hence is shown to be a reliable model for this problem.

13).Khoshgoftaar and Liu (2007) [10] also use multi-objective fitness functions for their proposed GP. However, in their case, the first objective is minimizing their newly proposed measure called “Modified Expected Cost of Misclassification” (MECM), while the second objective is to optimize the number of fault-prone modules. The problem at hand is that of fault-proneness estimation using four product metrics related to the number of lines and one process metric related to the “number of times the source file was inspected prior to system test.” Results show how the newly proposed approach can give acceptable performance without much deterioration in accuracy (between 1% and 18%), Type-I misclassification rate (between 1% and 3%) and Type-II misclassification rate (between 1%and 4%) on testing data sets.

14). Tsakonas and Dounias (2008) apply GP to four NASA data sets, namely CM1, KC1, KC2 and PC1 to predict software defect. They use the metrics of Halstead (1977) and McCabe (1976). The approach is compared to that of Menzies, DiStefano, Orrego, and Chapman (2004) and Menzies, Greenwald, and Frank (2007) using the Probability of Defect (PD) and Probability of Failure (PF) measures. It is found to be competitive in terms of its simplicity and accuracy.

15).Sheta and Al-Afeef (2010) use the developed lines of code and methodology metrics only in order to predict effort. GP is applied to NASA data sets presented by Bailey and Basili (1981). When compared to other models such as fuzzy logic, Halstead, Walston-Felix, Bailey-Basili, and Doty models, the GP shows a higher efficiency.

16).Bouktif, Sahraoui, and Antoniol (2006) use Simulated Annealing (SA) and Bayesian Classifiers (BC) to predict software stability. Their approach takes the BC as an input and adapts it using SA. The approach is found to be superior to the results of the best initial expert used built using Bayesian Classifiers (BC).

17). Another SA approach is that of Uysal (2008) for software component effort estimation. The approach is compared to that of Sheta (2006) and found to be a better estimation model.

18).Vandecruys et al. (2008) suggest the use of Ant Colony Optimization (ACO) for software fault-estimation. The proposed model, called Ant Miner+, uses a graph implementation of the classification rules. Each metric is a node. The path that the ant takes is considered to be the classification itself. For testing purposes, three open-source data sets of NASA software projects were used: PC1, PC4 and KC1. The approach is found to be competitive with techniques like C4.5, logistic regression and SVM, especially in the case of intuitiveness and comprehensibility. Azar and Vybihal (2011) propose an adaptive approach which takes as input already existing models and adapts them to new unseen data. An ACO is built for this purpose. Here also, the approach is tested on stability of classes in an Object-Oriented system. Experimental results prove the superiority of the proposed model over C4.5 and random guessing.

19). Cai et al. (1991) presented a review on software reliability modeling. The review discussed different types of probabilistic software reliability models and their shortcomings.

20).Karunanithi et al. (1991) [5] employed the BPNN to predict the software reliability and found that the NN models were consistent in prediction and their performance is comparable to that of other parametric models.

21). Karunanithi et al. (1992b) [5] predicted software reliability using a new connectionist approach which offers easy construction of complex models and estimation of parameters as well as good adaptability for different dataset. They used dataset collected from different software systems to compare the models. Based on the experiments, they found that the scaled representation of input–output variables provided better accuracy than the binary coded (or grey coded) representation. The experimental result obtained by them showed that the connectionist networks had less end point prediction errors than parametric models.

22).Karunanithi et al. (1992a) [6] ,[15] presented a solution to the scaling problems in which they used a clipped linear unit in the output layer.

23).Khoshgoftaar et al. (1992) [7], [8] explored the use of NN for predicting the number of faults in a program. They used static reliability modeling and compared its performances with that of regression models. They used dataset obtained from Ada development environment for the command control of a military data line communication system. They found that the absolute relative error of NN is less compared to regression model. Two decades as far as their applications to software engineering.

24).Khoshgoftaar and Szabo (1994) [9], [10], [16] used the principal component analysis (PCA) on NN for improving predictive quality. They used regression modeling and NN modeling.

25).Okamura and Dohi introduced a method by using Expectation-maximization (EM) principal. This paper considers an EM (expectation-maximization) based scheme for record value statistics (RVS) models in software reliability estimation. The RVS model provides one of the generalized modeling frameworks to unify several of existing software reliability models described as non-homogeneous Poisson processes (NHPPs). The proposed EM algorithm gives a numerically stable procedure to compute the maximum likelihood estimates of RVS models.

II. PROPOSED METHODOLOGY

A. Ant Colony Optimization Technique

Ant Colony is one of the techniques of bio inspired computing. The main concept of this is technique is that the self-organizing rules which allow the highly synchronized behavior of real ants can be utilized to manage populations of artificial agents that cooperate to solve computational problems. Various distinctive attributes of the behavior of ant colonies have inspired different kinds of ant algorithms. Examples are foraging, distribution of labor, issue sorting, and cooperative transport. Ants coordinate their activities via stigmergy, a form of implicit interaction mediated by changes in the environment. For example, a foraging ant deposit a chemical on the ground which raises the probability that other ant will follow the same path. Biologists have presented that many colony-level behaviors witnessed in social insects can be described through relatively simple models in which only stigmergic communication is present. In other words, biologists have shown that it is often sufficient to consider stigmergic, indirect communication to explain how social insects can attain self-organization. The notion behind ant algorithms is to use a form of artificial stigmergy to coordinate societies of artificial agents. One of the most effective examples of

ant algorithms is known as ‘‘ant colony optimization’’, or ACO. ACO is motivated by the foraging behavior of ant colonies, and targets discrete optimization problems. The ants may deposit a pheromone on the ground while returning back to their nests. The ants follow with high probability pheromone trails their sense on the ground.
Each Ant evaluates the next move to another vertex based on Gambardella et al., [21], [22],[23]

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}} & \text{if } j \in \text{ allowed} \quad k \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$p^{k}{}_{ij}$ *is the probability for a worker K to move to vertex “ij”*
ij τ is the amount of pheromone deposited on edge to “ij”
η is the inverted distance, describes how fast ants select their path.
The tour cost of each ant is given by *di j* the tour cost from the city *i* to city *j* (edge weight) is calculated and hence the shortest path is found. This is applied to the Travelling Sales Person Problem and optimized solutions are obtained using

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

The amount of pheromone deposited by each ant is given by

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \tag{3}$$

The value of $\Delta\tau_{ij}$ is calculated using

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{4}$$

$\Delta\tau_{i,j}^{k}$ is calculated using

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & if\,(i,j) \in \text{bestTour} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

B. Algorithm for Ant Colony Optimization with travelling salesman concept

The ACO algorithm which has been proposed based on the study that real ants are skilled in finding the shortest path from a food source to the nest without using visual signals. From the originating point the ants start the tour selecting randomly any path. The ACO algorithm is as follows:

1. Set the initial parameters.
2. Initialize pheromone trails.
3. Calculate the maximum specific ways in which the ants can travel.
4. Loop //iteration
5. Each ant is positioned at a given node randomly selecting the node according to some distribution strategy (each node has at least one ant)
6. For k=1 to m do //steps in a loop
7. The first step: move each ant in a different route
8. Repeat //till all the nodes are visited
9. Select node j to be visited next // the next node must not be an already visited node
10. Apply local updating rule
11. Until ant k has completed a tour
12. End for
13. Apply sub tour that is sub Local search // to improve tour
14. Apply global updating rule
15. Compute entropy value of current pheromone trails
16. Update the heuristic parameter
17. Until End_condition
18. End

C. Flow chart for Ant Colony Optimization

The flow chart for Ant Colony Optimization (ACO) algorithm with travelling salesman problem is shown in fig (1)

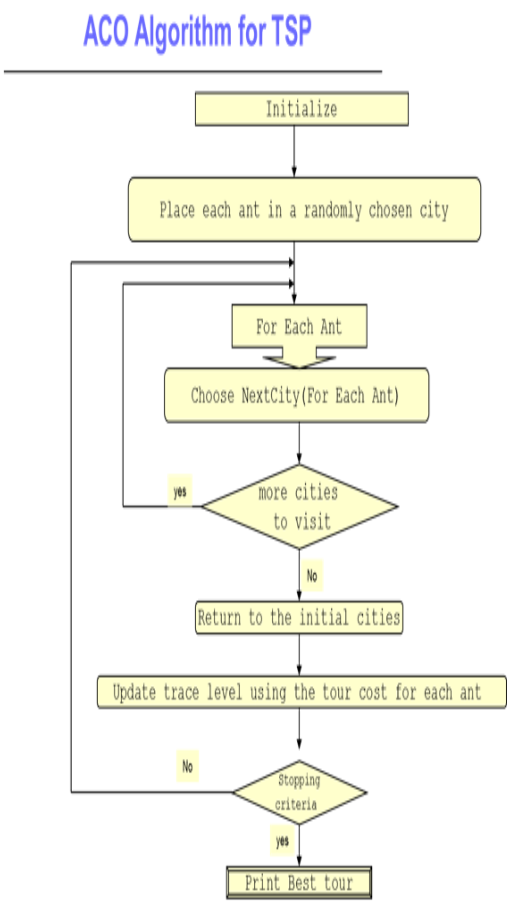
ACO Algorithm for TSP
Initialize
Place each ant in a randomly chosen city
For Each Ant
Choose NextCity(For Each Ant)
yes
more cities to visit
No
Return to the initial cities
Update trace level using the tour cost for each ant
No
Stopping criteria
yes
Print Best tour

Figure. 1. Flow chart of the ACO algorithm with travelling salesman problem

III. CONCLUSION

In this paper the authors discussed about the literature review for predicting software reliability and the proposed methodology.

The algorithms employed in this work are inspired by an observation emphasizing on real ants nature i.e. foraging and searching abilities that can provide good answers to genuine and real time optimization and software reliability solutions.

The exploration is still in progress as many of the facets of ACO algorithm are still to be unraveled. It is expected that this study stimulates further discussion for better reliability solutions.

REFERENCES

[1] R. K. Mohanty, V. Ravi, and M. R. Patra, “Hybrid intelligent Systems for predicting Software reliability,” Elsevier, Applied Soft Computing, vol. 13, No. 1, pp. 189-200, 2013.

[2] R. K. Mohanty, V. Ravi, and M. R. Patra, “Application of Machine learning Techniques to Predict software reliability,” International Journal of Applied Evolutionary Computation, vol. 1, No.3, pp. 70-86, 2010.

[3] K. Cai, C. Yuan, and M. L. Zhang, “A critical review on software reliability modeling,” Reliability engineering and Systems Safety, vol. 32, pp. 357-371, 1991.

[4] T. Dohi, Y, Nishio, and S. Osaki, “Optional software release scheduling based on artificial neural networks,” Annals of Software engineering, vol. 8, pp. 167-185, 1999.

[5] N. Karunanithi, Y. K. Malaiya, and D. Whitley, “The scaling problem in neural networks for software reliability prediction,” In Proceedings of the Third International IEEE Symposium of Software Reliability Engineering, Los Alamitos, CA, pp. 76- 82, 1992a.

[6] N. Karunanithi, D. Whitley, and Y.K. Malaiya, “Prediction of software reliability using connectionist models,” IEEE Transactions on Software Engineering, vol. 18, pp. 563-574, 1992b.

[7] T. M. Khoshgoftaar, D. L. Lanning, and A. S. Pandya, “A neural network modeling for detection of high-risk program,” In Proceedings of the Fourth IEEE International Symposium on Software reliability Engineering, Denver, Colorado, pp. 302-309, 1993.

[8] T. M. Khoshgoftaar, and P. Rebours, “Noise elimination with partitioning filter for software quality estimation,“ International Journal of Computer Application in Technology, vol. 27, pp. 246-258, 2003.

[9] T. M. Khoshgoftaar, A.S. Pandya, and H.B. More, “A neural network approach for predicting software development faults,” In Proceedings of the third IEEE International Symposium on Software Reliability Engineering, Los Aiamitos, CA, pp. 83- 89, 1992.

[10] T. M. Khoshgoftaar, E. B. Allen, and J.P. Hudepohl, S.J. Aud, “Application of neural networks to software quality modeling of a very large telecommunications system,” IEEE

[11] Transactions on Neural Networks, vol. 8, No. 4, pp. 902-909, 1997.

[12] T. M. Khoshgoftaar, E.B. Allen, W. D. Jones, and J. P. Hudepohl, “Classification –Tree models of software quality over multiple releases,” IEEE Transactions on Reliability, vol. 49, No. 1, pp. 4-11, 2000.

[13] J. R. Koza, “Genetic Programming: On the Programming of Computers by Means of Natural Selection”. Cambridge, MA: The MIT Press, 1992.

[14] J. D. Musa, Iannino, A., and K. Okumoto, “Software Reliability, Measurement, Prediction and Application,” McGraw-Hill, New York, 1987.

[15] J. D. Musa,”Software reliability data,” IEEE Computer Society- Repository, 1979.

[16] N. Karunanithi, D. Whitley, and Y.K. Malaiya, “Prediction of software reliability using neural networks,” International Symposium on Software Reliability, pp. 124-130, 1991.

[17] T.M. Khoshgoftaar, and R.M. Szabo, “Predicting software quality, during testing using neural network models: A comparative study,” International Journal of Reliability, Quality and Safety Engineering, vol. 1, pp. 303-319, 1994.

[18] L. Tian, and A. Noore, “Evolutionary neural network modeling for software cumulative failure time prediction,” Reliability Engineering and System Safety, vol. 87, pp. 45-51, 2005b.

[19] N. Rajkiran, and V. Ravi. “Software Reliability prediction by soft computing technique,” The Journal of Systems and Software, vol. 81, No.4, pp. 576-583, 2007.

[20] N. Rajkiran, and V. Ravi, “Software Reliability prediction using wavelet Neural Networks,” International Conference on Computational Intelligence and Multimedia Application (ICCIMA, 2007), pp. 195-197, 2007

[21] V. Ravi, N. J. Chauhan, and N. RajKiran., “Software reliability prediction using intelligent techniques: Application to operational risk prediction in Firms,” International Journal of Computational Intelligence and Applications, vol.8, No. 2, pp. 181-194, 2009.

[22] L. M. Gambardella, E. D. Taillard, and M. Dorigo, “Ant Colonies for the Quadratic Assignment Problem,” Journal of the Operational Research Society, The Journal of the Operational Research Society, vol. 50, No.2, pp. 167-176,1999.

[23] L. M. Gambardella, E. D. Taillard, and G. Agazzi, “MACSVRPTW: A multiple ant colony system for vehicle routing problems with time windows,” In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pp. 63–76. Hill, London, UK, 1999.

[24] L. M. Gambardella, E. D. Taillard, and M. Dorigo, “Ant colonies for the quadratic assignment problem,” Journal of the Operational Research Society, vol.50, No.2, pp.167–176, 1999.