World Applied Sciences Journal 31 (11): 1983-1991, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.31.11.1731

A Heuristic Model for Estimating Component-Based Software System Reliability Using Ant Colony Optimization

[1]Kirti Tyagi and [2]Arun Sharma

[1]Department of Computer Science and Engineering,
Ajay Kumar Garg Engineering College, Ghaziabad, India
[2]Department of Computer Scienceand Engineering,
Krishna Institute of Engineering and Technology, Ghaziabad, India

Abstract: Most software reliability models for component-based software systems (CBSSs) depend on either the system architecture or component reliability and glue code reliability. It is important to optimize the usage of paths to obtain more accurate estimates of CBSS reliability. By using bio-inspired optimization techniques that are currently being developed, we can find the value of the most-used path for any CBSS. *This* paper develops a new model for estimating CBSS reliability which accounts for path usage. The proposed model is proved to perform better than other models. *This* paper proposes a model for estimating CBSS reliability using a bio-inspired algorithm, ant colony optimization (ACO). This model introduces heuristic component dependency graphs (HCDGs), which help to estimate CBSS reliability. This paper also proposes using the reliability of components and assessing their impact on overall system reliability. *The* results of this paper show that HCDGs give better reliability estimates than other existing methods. A case study of a security management system is presented, along with its results. *The* proposed approach is fully dependent on CBSS characteristics and provides a good reliability estimation method for CBSSs, taking into consideration both component reliability and CBSS reliability.

Key words: Reliability Estimation · Component-Based System · Component Dependency Graph (CDG) · Path · Heuristic Method

INTRODUCTION

For any software, there are two user requirements: reliability and availability. Reliability will be required if the product's nonperformance has the greatest impact, and availability will be required if downtime has the greatest impact. Most of the time, reliability may be measured probabilistically as

R_{SYS} = (1- probability of failure).

Software reliability is thus defined as the probability of the failure-free operation of a software system for a specified period of time in a specified environment.

Component-based software engineering (CBSE) is a specialized form of software reuse concerned with building software from existing components (including commercial off-the-shelf components, COTSs) by assembling them together in an interoperable manner. Achieving a highly reliable software application is a difficult task, even when high-quality software components that are pre-tested and trusted are composed together. As a result, several techniques have emerged to analyze the reliability of component-based applications. These can be categorized in one of the following two ways:

- Reliability is estimated for the application as a whole.
- The application's reliability is estimated using the reliability of the individual components and their interconnection mechanisms.

The major difference between software and other engineering artifacts is that software is pure design. Software unreliability is always the result of design faults, which in turn arise from human failures. Hardware systems fail through design and manufacturing defects more often than is desirable. CBS reliability depends heavily

Corresponding Author: Kirti Tyagi, Department of Computer Science and Engineering,
Ajay Kumar Garg Engineering College, Ghaziabad, India.

upon the interaction among components. Interaction promotes dependencies, and greater dependencies lead to complex systems. Hence, reliability estimation will be difficult for these systems.

Traditional approaches to software reliability analysis treat the software as a whole and use test data during the software test phase to model only the interactions with the outside world. These are known as black-box models. Because software applications are continually becoming more complex and there is greater emphasis on reuse, component-based software system (CBSS) applications have emerged. Black-box models ignore the structure of software constructed from components as well as the reliability of individual components and are therefore not appropriate for modeling CBSS applications.

Now a days bio inspired optimization technique are in existence and considered to be most suitable for optimization problems, since problem of reliability is an optimization problem so we want to optimize the reliability of any software application.

In this paper, we propose a new model for CBSS reliability estimation using ant colony optimization (ACO). We chose ACO for reliability estimation based on its success in other domains such as bioinformatics and electronics. The main difference between hardware reliability and software reliability is that software reliability depends on freedom from human error, which makes it more difficult to estimate than hardware reliability. In this paper we focus on estimating reliability with the heuristic ACO approach. We also propose a new data structure called the *heuristic component dependency graph* (HCDG). This graph is the same as the CDG proposed by Yacub *et al*., but with minor modification: heuristic information is associated with the HCDG, namely, the transmission probability of an ant from one component to another. This heuristic information is used to calculate the most-used path in the CBSS, which in enables us to determine the overall reliability of the application.

The rest of this paper is organized as follows. Section 2 presents related work on reliability estimation for CBSs. Section 3 explains the reliability assessment and estimation problem for the proposed model. Section 4 explains the proposed Estimation of Reliability using ACO (ACOREL) algorithm. Section 5 presents a case study and results, and Section 6 presents our conclusion and future research directions.

Related Work: Many approaches and models have been proposed for reliability estimation of CBSS, some are based on mathematical formula and others are based on soft computing techniques. Some models are characterized as architecture based reliability models; those are based on the architecture of the application.

We divided the related work into three categories namely: architectural models, mathematical models and soft computing based approaches.

Architectural Models: Architecture-based reliability models such as state-based and path-based models are most suitable for CBSSs [1-3]. Common requirements for architecture-based reliability models are (1) identifying the components, (2) describing the software architecture, (3) describing the failure behavior, and (4) combining the architecture with the failure behavior. State-based models [5-8] consider the control flow between objects. Transfer among components is modeled as a Markov behavior, which means that current component behavior is independent of past behavior. A limitation of these models is that they assume a constant failure rate for components, which is not realistic. Path-based models [9] [10] consider possible execution paths for estimating the application's reliability. These models define.

R_p = path reliability of path p = $\Pi_{i=0}^{i=n} Ri$, and

R_s = system reliability = $\sum_{j=0}^{N} Rp / P$,

where n is the number of components in path p, R_i is the reliability of component i, N is the total number of paths, and P is the path number.

A limitation of the path-based models is that they only provide an approximate estimation of software reliability. They are additive models which estimate reliability based on component failure data and do not explicitly consider the software architecture.

Mathematical Models: Researchers have also proposed mathematical models for estimating CBSS reliability.

Heiko and Koziolek *et al*. [4] described the role of a component as a transformer from one operational profile to another.

Yacoub *et al*. [11] proposed an approach to reliability analysis called *path-based reliability analysis*. This approach introduces CDGs, which can be extended for complex distributed systems. Using this algorithm, sensitivity is analyzed as a function of component reliabilities and link reliabilities. The approach is based on paths which can be captured with sequence diagrams, so that this approach can be automated. A limitation of

this approach is that it does not consider failure dependencies among the components.

Fan Zhang *et al*. [12] proposed a model based on a CDG. In this model, an operational profile of a system is given, and the model can be used to check whether reliability changes when the operational profile changes. Assuming that control flow transits from component *i* to component *j*, component *j*'s reliability is calculated as $T_{ij} \times (R_{ij} \times W_{ij})$, where

T_{ij} = The transition probability from component *i* to component *j*,
R_{ij} = The reliability vector for each subdomain of component *j*, and
W_{ij} = The weight vector for each subdomain of component *j* for the transition from component *i* to *j*.

Dong *et al*. [13] proposed a method for CBSS reliability estimation in which component relationships are analyzed and solved using a Markov model. This technique extends the scope of Markov models. A limitation of this model is that it assumes that all component reliabilities and transition probabilities are given, but in practice this is not always true.

Huag *et al*. [14] proposed a technique based on algebra which provides a framework that can be implemented on Maude for describing syntax and predicting reliability.

Goswami and Acharya [15] proposed an approach to CBSS reliability analysis which takes into consideration the system's component usage ratio, calculated through mathematical formulas. Due to the flexibility of the component usage ratio, this approach may be used for real-time applications.

Si [16] proposed a framework for estimating reliability through a component composition mechanism. The approach proposes five basic component composition mechanisms and techniques for their reliability estimation. After calculating the reliability for each composition, a procedure estimates the overall application reliability based on the component composition mechanisms and component utilization frequencies. It is possible to recognize additional composition mechanisms.

Hsu [17] proposed an adaptive reliability estimation technique using path testing for complex component-based systems. Three methods are proposed for estimating path reliability, namely, sequence, branch and loop structures. The proposed path reliability can then be used for estimating the reliability of the overall application. This approach can provide a promising estimation of software reliability when testing information is available. A sensitivity analysis is also performed to determine the effect of each node on the system's reliability.

Wang [18] proposed an approach for reliability estimation based on rewrite logic (RABRL). This method considers systems whose specification is given with an operational profile. Maude's rewrite technique is used to estimate the reliability. This technique statistically analyzes an application's execution process and uses this to approximately estimate the transition probabilities between components and the expected number of visits to components. However, this approach has some limitations: first, it can only be applied to simple CBSs, and second, it does not consider failure dependencies between components.

Hu *et al*. [19] proposed an approach to software reliability estimation using modified adaptive testing (MAT). This testing allows test case histories to be used.

Soft-Computing-Based Approaches: Lo [20] proposed a software reliability estimation model based on a support vector machine (SVM) and a genetic algorithm (GA). This model assumes that recent failure data alone are sufficient for estimating reliability. Reliability estimation parameters for the SVM are determined by the GA. This model is less dependent on failure data than are other models.

Dimov [21] proposed a fuzzy reliability model for CBSSs, based on fuzzy logic and possibility theory. A mathematical fuzzy model based on necessity and possibility is proposed to predict the reliability of a CBSS. Like many other models, this model does not require component failure data because it is based on uncertainty. However, a mechanism is necessary to model the propagation of failure between components and failure behavior.

Singh [22] presented a Bayesian reliability estimation model using a unified modeling language (UML) technique for reliability prediction and assessment. The technique provides reliability analysis at the design level, i.e., before the system development and integration level. Because this approach is based on UML diagrams, reliability can be predicted in the early design phase. This approach is scalable because all calculations are done by an automated tool. The approach has one limitation: if new paths are taken into account, the reliability prediction algorithm considers this to be a new system, so that separate operational profiles are generated.

Tyagi *et al*. [23] proposed a rule-based approach for estimating CBSS reliability. This approach proposes four critical factors for reliability estimation and uses these factors to create a rule base. The results are defuzzified.

The present paper proposes a heuristic reliability model based on ACO. The reason to propose a model based on optimization is that usage frequency of every component is not same in the overall application,hence the reliability of every component cannot contribute equally to the overall reliability estimation of CBSS. By using ACO for reliability estimation we find most used path. On the basis of components involved in the most used path we estimate the reliability of overall application. The most common way to represent a CBS is with a component dependency graph (CDG), which can fully capture dependencies among components. After identifying the dependencies among components, we use ACO to determine the reliability of the overall application. For this reason, our technique uses a heuristic CDG (HCDG) instead of a CDG.

Problem Definition (Reliability Assessment Using ACO): CBSS reliability is commonly expressed in terms of the reliabilities of components and their interconnection mechanisms. Component reliability can be defined in terms of defect densities: the reliability of a component is inversely proportional to its defect density. That is, the greater the defect density is, the less the reliability will be.

Let R_{CBS} be the reliability of the overall software application. Then for some function F, we want:

$$F(R_1, R_2, ..., R_n) = R_{CBS}, \quad (1)$$

where *n* is the number of components in the system and R_i is the reliability of component *i*. There will be a number of operational profiles for the variables R_i. To obtain an actual estimate of reliability, the method should be based upon both theory and experience. Hence, obtaining the reliability of the CBS at the initial phase, that is, the design and analysis phase, is a key problem. This paper introduces a new technique based on ACO for estimating the CBS reliability at this phase. This new model of reliability estimation makes the following assumptions:

- Paths are the main focus of this algorithm. For any CBSS, paths are the main execution areas.
- Every CDG has some heuristic information associated with it: an amount of pheromone and the probability of an ant's transition from node *i* to node *j*.

We define a new data structure, the HCDG, based on the CDG proposed by Yacub *et al*. [11]. The difference between a CDG and an HCDG is that the latter includes heuristic information about the ants.

The Heuristic Component Dependency Graph (HCDG): An HCDG is a 4-tuple < N,E,s,t>, consisting of

- A set of nodes N, initialized as {n},
- A set of edges E,
- A source node s,
- A destination node d, and

where n represents a component of the CBSS.

Every node n has the following information associated with it:

n = $< C_i, \mu C_i, AC_i >$, where
C_i = component *i*,
μC_i = the reliability of component C_i, and
Ac_i = the average execution time of component C_i in the path.

Every edge e has the following information associated with it:

E = $< t_{ij}, rt_{ij}, Pt_{ij}, \rho_{ij} >$, where
t_{ij} = transition from n_i to n_j,
rt_{ij} = the probability that information delivered from C_i to C_j is error-free,
Pt_{ij} = the transition probability: the probability that C_j will execute next, given that C_i is currently executing (the sum of the outgoing transition probabilities from any node must be unity), and
ρ_{ij} = transition probability of ant moving from C_i to C_j, which depends upon the amount of pheromone.

An example of an HCDG is shown in Figure 1.

Problem Solution (Estimation of Reliability Using ACO, or ACOREL): Ant colony optimization (ACO) is a relatively new random heuristic approach for solving optimization problems as given by Dorigo *et.al* (1999). It is derived from the way real ants optimize their tracks. In the early 1990s Colorni et.al (1991) proposed the first ACO algorithm, called Ant System (AS). Ant system is the first member of ACO class algorithms. The main underlying idea behind this is the parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result. ACO is the paradigm for designing

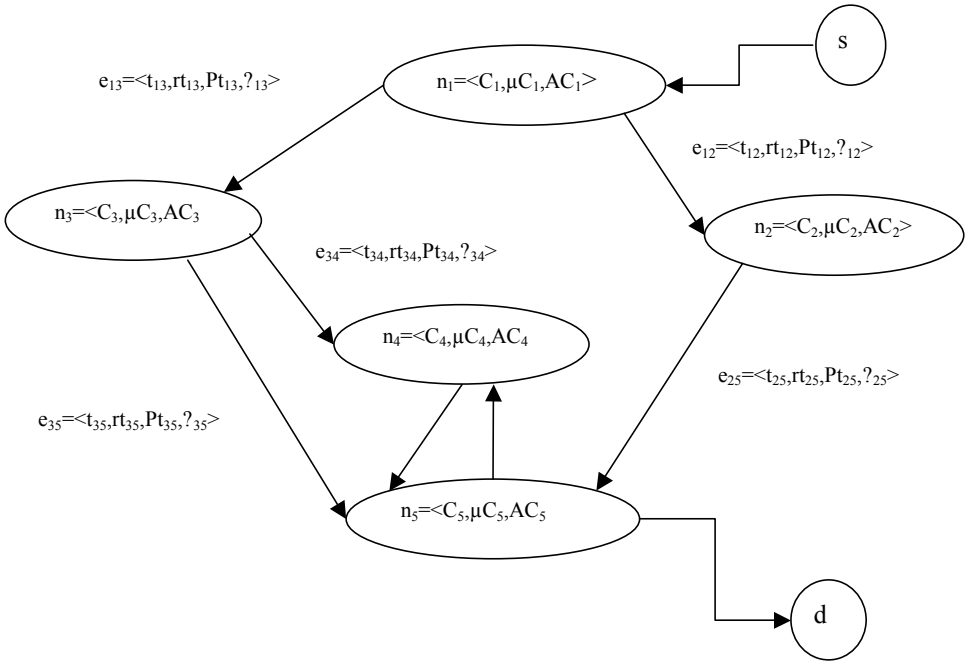

Fig. 1: Representation of an HCDG

metaheuristic algorithms for combinatorial optimization problems. It is the main application of the ACO algorithm. The collective behavior of different search threads of ant system has made it effective for solving combinatorial optimization problems.

Ant colony optimization algorithm solves the optimization problem by repeating the following two steps:

- Candidate solutions are constructed using a pheromone model,
- The candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling toward high quality solutions.

Actually the ant colony optimization algorithm uses the artificial ants which imitates the behavior of real ants and co-operates the solution of a problem by exchanging the information via pheromones.

The ACOREL Algorithm: The ACOREL algorithm presented in this paper is a direct extension of the ACO algorithm proposed by [24]. This is the simplest algorithm among all propose ACO algorithm. We choose this initially and association with some better algorithm is left as a future work.The following are important assumptions and characteristics of this algorithm:

- Ants are the main part of the algorithm. We assume that initially there is one ant at each node. The ants act independently.
- Each ant lays down an amount of pheromone. The amount of pheromone may be updated on each move of the ant.
- Ants move step-by-step from source s to destination d.
- The ants are artificial ants. Each ant has:

- Memory (for updating pheromone),
- Pheromone, and
- Heuristic information: the transition probability that the ant moves from C_i to C_j.

Step 1: Let N = number of components.

Let μC_i = reliability of component C_i, for each *i*.
Each path has some number of components. Initially, at time t = 0, we put one ant at each node.

Step 2: If stopping condition is met then stop moving else x moves from starting condition.

Step 3: Each ant moves from the previous component C_i to the next component C_j with transition probability $\rho_{ij}(t)$, where $\rho_{ij}(t)$ is defined as follows:

$$\rho_{ij}(t) = ([\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}) / \Sigma ([\tau_{ik}]^{\alpha} [\eta_{ik}^{\beta}) \text{ if } j \in \text{allowed}_k \quad (2)$$

$k \in \text{allowed}_k$

where
τ_{ij} = Pheromone amount,
η_{ij} = Heuristic information = (1- rt_{ij}), (3)

where r_{ij} is the reliability of path *ij*,

α = relative importance of trail,
β = relative importance of heuristic information,
$\text{allowed}_k = \{M - \text{tabu}_k\}$
(M = set of nodes; tabu_k = the set of nodes visited by the k^{th} ant in the current tour), and W_{ij} = weight of transition from *i* to *j*.

Step 4: Pheromone updating.

Pheromone will be inserted into the path with the current highest reliability. Pheromone is

Updated according to the following formula:

$$\tau_{ij}^{\text{updated}} = (1- \rho)\ \tau_{ij}^{\text{prev}} + rt_{ij} \quad (4)$$

where ρ = evaporation coefficient.

Step 5: If $WR_i > R_{,j}$ pheromone will be vaporized according to

$$\tau_{ij}^{\text{updated}} = (1- \rho)\ \tau_{ij}^{\text{prev}} \quad (5)$$

Estimation of edge reliability rt_{ij}: The edge reliability rt_{ij} is the probability that the information delivered from C_i to C_j is error-free. This depends upon the following factors:

- μC_i ,
- Number of methods, and
- Operation profile of methods.

The probability rt_{ij} can be estimated using the following equation:

rt_{ij} = μC_i + correlation between C_i and C_j.

Average Execution Time of a Component (Ac_i): The average execution time of a component may be expressed in terms of the interaction frequency of the component in that particular path:

$$AC_i = \sum_{p=1}^{Q} IFi * (Total\ time\ taken\ by\ component\ Ci\ in\ Sp) \quad 6)$$

where Q is the total number of paths, IF is the interaction frequency, and the execution time of C_i is measured as the time between giving input to C_i and getting its output.

System Reliability Estimation: Let P_{ij} be the probability that C_i is present in S_j (path *j*)-this is known as the *component path probability*, let r_i be the reliability of C_i in S_j, defined as

$r_i = ([1- (1- \mu C_i)AC_i])$,

and let N be the total number of components in path S*j*. Then the reliability of S_j is defined as

$$rS_j = \Pi_{i=0}^{N} ri * tij\ , \quad (7)$$

and

R_{CBS} = the reliability of the most-used path S*j* = rS_j. (8)

By the *most-used* path, we mean the path for which ρ_{ij} is maximal.

Case Study: For our case study, we choose a small component-based security system application. This system has a total of eight components and five paths. The graphical representation of the security management system is given in Figure 2. The components and their parameters are shown in Table 1. This software is developed by combining the components. The domain of the application is the set of components and the execution paths from the start node to the destination node.

Description of Paths: The cycle completion from the initial state to the final state is analyzed through the five paths shown in Table 2. The details of the paths are as follows:

Table 1: For Component reliability and average execution time

Name of Component	Reliability of Component (μC_i)	Average Execution Time (AC_i)
Login (C_1)	.6578	6
Server(C_2)	.4	15
Time Management System (C_3)	.4878	2
Alarm Management System (C_4)	.4878	18
Access Management System (C_5)	.5	8
Calculation Script(C_6)	.689	8
Door Management System(C_7)	.789	.8
Devices(C_8)	.3245	12

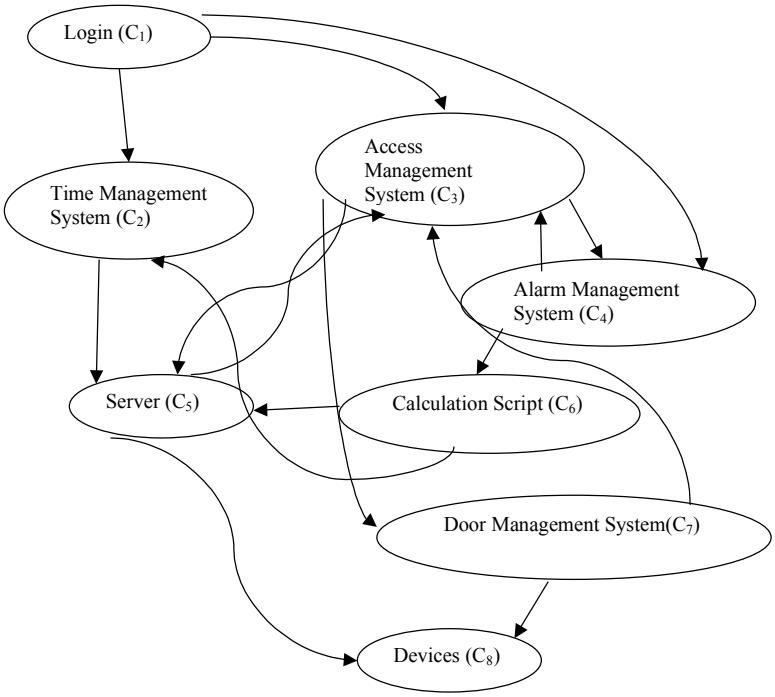

Fig. 2: HCDG of srcurity system

Table 2: Adjacency matrix for safety system

Component	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
C_1	0	1	1	1	0	0	0	0
C_2	0	0	0	0	1	0	0	0
C_3	0	0	0	1	1	0	1	0
C_4	0	0	1	0	1	1	0	0
C_5	0	0	1	0	0	0	0	1
C_6	0	1	0	0	1	0	0	0
C_7	0	0	1	0	0	0	0	1
C_8	0	0	0	0	0	0	0	0

Table 3: For rt_{ij} for edges

Component	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
C_1	0	.55	.34	.48	0	0	0	0
C_2	0	0	0	0	.60	0	0	0
C_3	0	0	0	.80	.25	0	.55	0
C_4	0	0	.25	0	.82	.55	0	0
C_5	0	0	.70	0	0	0	0	.72
C_6	0	.37	0	0	.45	0	0	0
C_7	0	0	.28	0	0	0	0	.75
C_8	0	0	0	0	0	0	0	0

Table 4: For transition probability

Component	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
C_1	0	.2761	.4049	.3190	0	0	0	0
C_2	0	0	0	0	1	0	0	0
C_3	0	0	0	.1429	.5357	0	.3214	0
C_4	0	0	.5435	0	.1304	.3261	0	0
C_5	0	0	.5172	0	0	0	0	.4828
C_6	0	.5339	0	0	.4661	0	0	0
C_7	0	0	.7432	0	0	0	0	.2577
C_8	0	0	0	0	0	0	0	0

Table 5: path parametrs

Path Name	Components and edges include in the Path	Heuristic Information ρ_{ij}
Time Management(Path 1)	$C_1e_1\ C_2\ e_2\ C_5\ e_3\ C_8$	.020004
Access Management (Path 2)	$C_1\ e_4\ C_3\ e_5\ C_7\ e_6\ C_8$	.027145
Alarm Management (Path 3)	$C_1\ e_7\ C_4\ e_8\ C_5\ e_3\ C_8$	.028684
Calculation (Path 4)	$C_1\ e_7\ C_4\ e_9\ C_6\ e_{10}\ C_5\ e_3\ C_8$	.038588
Serving (Path 5)	C1 e7 C4 e9 C6 e11 C2 e2 C5 e3 C8	.054466

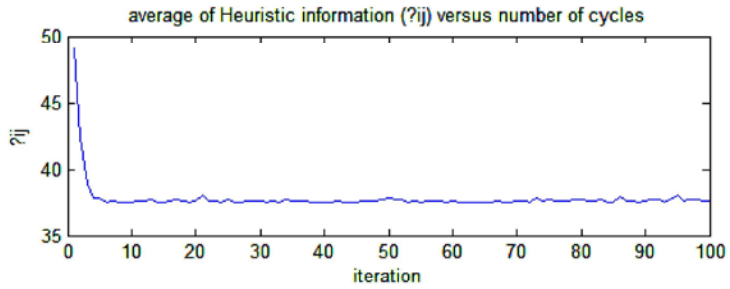

Fig. 3: Average heuristic information and number of cycles

Time Management: This path fetches the data and user id from the login component and then supplies these to the Time Management System (TMS), which gives all time-related parameters to the server as InTime, Out Time, BreakTime and Absence Time. The server gives details of the alarm and door status to devices.

Access Management: This path fetches the user details from the login component and passes them to the Door Management System (DMS), and DMS sends its output as input to the devices.

Alarm Management: This path fetches data from the login component and supplies this data to the Alarm Management System (AMS). The AMS sends its output as input to the server, and the server sends all the details of the alarm id and door status to the devices.

Calculation: This path fetches data from the login component and supplies it to the AMS. The AMS sends its output to the calculation scripts where necessary calculations about the alarm and its time are performed. These calculations are supplied to the server, and from the server they go to the devices.

Serving: This path estimates the details of the alarm and manages the time according to the estimated details. Path related parameter is shown in Table 5.

By using above table and formula from (7) we can find that reliability of the application is 88%.

CONCLUSION

This paper proposes a new model for estimating reliability of CBSS. This proposes a heuristic algorithm named as ACOREL algorithm. The proposed algorithm is direct extension of ACO algorithm. Algorithm results in identifying the most used path and then on the basis of this path reliability of application may be estimated. The proposed model has some limitations also it check the reliability of most used scenario.

REFERENCES

1. Goseva Popostojanova, K. and K.S. Trivedi, 2001." Architecture based approach to Reliability Assessment of Software Systems," Performance Evaluation Journal, 45(2): 179-204.
2. Kalyuan Cai, Chenggang Bai, and Xiaojun Zhong, 2003." Introducyion to Reliabi;ity Models of component Based Software System", Journal of Xi'an Jiaotong University, 37(6): 560-564.
3. Swapana S. Gokhle, 2007. " Architecture Based Software Reliability Analysis: Overview and Limitations," IEEE Transactions on dependable and Secure Computing, 4(1): 32-40.
4. Heiko koziole, Steffen Becker. Transforming operational profiles of Software Components for quality of service predictions. Proc. 10th WCOP'05, pp: 1-8.
5. Cheung, R.C., 1980. " A user oriented software reliability model," IEEE Transaction on Software Engineering, 6(2): 118-125.

6. Wen-Li Wang, D. Pan and M.H. Chen, 2006." Architecture-based software reliability modelling," Journal of Systems and software, 79(1): 132-146.
7. Swapana, S. Gokhle, W.E. Dong, K.S. Trivedi and J.R. Horgan, 1998. " An Analytical Approach to AμCitecture based Software Reliability Prediction," Proceeding of the third International Computer Performance and Dependability Symposium, Durham, USA, pp: 13-22.
8. Littlewood, B., 1979. " Software reliability Model for Modular program Structure," IEEE Transaction on Reliability, 28(3): 241-246.
9. Shooman, M.,1976. " Structural Models for Software Reliability Prediction,"Proceeding of Se Cond International Conference on Software Engineering, san Francisco, USA, pp: 268-280.
10. Krishnamurthy, S. and A.P. Mathur, 1997. " On the Estimation of Reliability of a Software System Using Reliabilities of its Components," Proceeding of the eighth international Symposium on Software reliability E ngineering,Albuquerque, USA, pp: 146-155.
11. Yacub, S., B. Cukic and H. Ammar, 2004. " Scenario based reliability analysis approach for Component Based Systems," IEEE Transaction on Reliability, 53(4): 465-480.
12. Zhang, F., X. Zhou, Y. Dong and J. Chen, 2009. Consider of fault propagation in Architecture-based software reliability analysis. Int Conf Comput Syst and App; pp: 783-6.
13. Wang Dong, Ning Huang and Ye Ming, 2008. Reliability Analysis of Component-based Software Based on Relationships of Components, IEEE Conference on Web Services, pp: 814-815.
14. Ning Huang, Dong Wang and J.I.A. Xiaoguang, 2008. FAST ABSTRACT: An Algebra-Based Reliability Prediction Approach for Composite Web Services, 19th International Symposium on Software Relability Engineering, pp: 285-286.
15. Vivek Goswami and Y.B. Acharya, 2009, Method for Reliability Estimation of COTS Components based Software Systems International. Symposium on Software Reliability Engineering (ISSRE 2009).
16. Yuanjie Si, Xiaohu Yang, Xinyu Wang, Chao Huang and Aleksander J. Kavs, 2011. An Architecture-Based Reliability Estimation framework Through Component Composition Mechanisms, 2nd International Conference on Computer Engineering and Technology, pp: 165-170.
17. Chao-Jung Hsu and Chin-Yu Huang, 2011. An Adaptive Reliability Analysis Using Path Testing for Complex Component based Software Systems, IEEE transaction on reliability, 60(1): 158-170.
18. Dong Wang and Ning Huang, 2008, Reliability Analysis of Component Based Software Based on Rewrite Logic, 12th IEEE International Workshop on Future Trends of Distributed Computing Systems, pp: 126-132.
19. Hai Hu, Chang-Hai Jiang, Kai-Yuan Cai, W. Eric Wong and Aditya P. Mathur, 2013. Enhancing software reliability estimates using modified adaptive testing. Information & Software Technology, 55(2): 288-300.
20. Lo, J.H., 2010. Early software reliability prediction based on support vector machines with genetic algorithms. Fifth IEEE Conf Ind Electron App, pp: 2221-6.
21. Aleksandar Dimov and Sasikumar Punnekkat, 2010, Fuzzy Reliability Model for Component-based Software Systems, 36th EUROMICRO Conference on Software Engineering and Advanced Applications, pp: 39-46.
22. Singh, H., V. Cortellessa, B. Cukic, E. Gunel and V. Bharadwaj ,2001. A Bayesian approach to reliability prediction and assessment of component based systems, in 12th IEEE International. Symposium on Software Reliability Engineering , (Hong Kong, Nov. 2001), pp: 12-21.
23. Tyagi, Kirti, and Arun Sharma, 2012. "A rule-based approach for estimating the reliability of component-based systems." Advances in Engineering Software, 54: 24-29.
24. Dorigo, M. and T. Stützle, 2010. Ant Colony Optimization: Overview and Recent Advances. M. Gendreau and Y. Potvin, editors, Handbook of Metaheuristics, 2nd edition. Vol. 146 in International Series in Operations Research & Management Science, Springer, Verlag, New York, pp: 227-263.