FUZZY-CONTROLLED GENETIC ALGORITHM FOR FAULT DETECTION IN DISTRIBUTED SYSTEMS

Krishna Prasad

Department of Computer Science & Engineering, SRM University, A.P, India

ABSTRACT

This work investigates the Fuzzy-Genetic approach for fault detection in distribution systems with having at most (N-1)/2 faulty systems out of N systems tested. This study also reviews different types of faults occurring in the distributed system and fault detection techniques. A fault can occur due to many reasons like link or resource failures or other and is to be detected and provisioned for working of the system smoothly and accurately. These faults need be detected and recovered by suitable techniques according to the requirement. An efficient fault detector can avoid loss due to system crash by triggering the required fault tolerance mechanism. This work provides how a fuzzy-controlled genetic algorithm can be applied to detect faults in Distributed Systems.

KEYWORDS

Fault Detection, Genetic algorithms, Fuzzy mutation, Distributed System.

1. INTRODUCTION

A distributed system or part of it getting into a faulty state due to any reason while processing a task may damage entire system or a part of it. To avoid damage, the involved faults have to be properly detected and recovered within a specified time. A task running on real time distributed system should be feasible and reliable. These systems must function with high availability under all types of hardware and software faults. Fault-tolerance mechanism for a distributed system is crucial for users trust on these systems. Hardware and software redundancy are well-known effective methods. Hardware fault-tolerance achieved through applying extra hardware like processors, communication links, memory, I/O devices, whereas in software fault tolerance tasks, messages are added into the system to deal with faults. Fault should be detected by applying reliable fault detector followed by some recovery technique. Many fault detection techniques are available but it is also necessary to apply appropriate fault detector. Unreliable fault detector can make mistake by erroneously suspecting correct process or trusting crashed process.

A crucial problem in this area, known as the fault diagnosis problem, is to identify the fault status of all processors in the system, i.e., to answer the question which of them are faulty and which are fault-free. Preparata et al. [1], first introduced a formal graph-theoretical model for system diagnosis. Since the introduction of the PMC model, significant progress has been made in both theory and practice associated with the original model and its offshoots. Many studies have investigated two classical goals: one step t-fault diagnosis, where the objective is to locate all faulty units provided the number of faulty units present does not exceed t, and the sequential t-fault diagnosis which aims to identify at least one faulty unit under the same assumption.

Natarajan Meghanathan et al. (Eds) : ICCSEA, WiMoA, SCAI, SPPR, InWeS, NECO - 2019
pp. 17-26, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.91802

1.1. Types of Faults

There are different types of faults which can occur in distributed System [2]. These faults can be classified with factors such as:
Network fault: A Fault occur in a network due to network partition, Packet Loss, Packet corruption, destination failure and link failure, etc.
Physical faults: This Fault can occur in hardware like fault in CPUs, Fault in memory, Fault in storage, etc.
Media faults: Fault occurs due to media head crashes.
Processor faults: fault occurs in processor due to operating system crashes, etc.
Process faults: A fault which occurs due to shortage of resource, software bugs, etc.
Service expiry fault: The service time of a resource may expire while application is using it.
Different types of faults based on their nature with respect to time [3] which usually occurs in a distributed system are as follows:
Permanent Fault, Transient Fault, Intermittent Fault and Byzantine Fault.
The following two approaches are followed to achieve the fault tolerance in distributed system.
Method 1: Detecting and Diagnosing faulty nodes by N-modular Redundancy, Information Redundancy, Computational Redundancy, Dynamic Redundancy, Logic circuit- level testing, Component or System level testing
Method 2: Masking of faulty nodes and reconfiguration of the system.
In method 1, during the detection and diagnosis of faulty nodes, the first 3 techniques incur hardware overhead and computational time overhead for large computations.
Dynamic redundancy is proved as the best among different redundancy schemes as the monitoring of the units are repeated after a specific interval. Fourth technique needs large amount of data to be generated, stored and produced that again leads to the problem of hardware overhead. The last technique i.e. component or system level testing is cheapest among all and overcomes the problem of redundancy as well as complexity of testing at chip-level. System level testing is the most popular among all the diagnosis techniques, where the testing operation is accomplished among nodes of a distributed system at system level means at the unit level. In this work, basically system level fault diagnosis technique is applied to K-connected distributed systems.

1.2. Related works

Several models for system-level diagnosis can be found in the literature. Based on the system-level diagnosis theory, Kuhl and Reddy [4] proposed the concept of distributed diagnosis. There are three basic phases of a distributed diagnosis process: detection, localization and recovery. Detection refers to the ability of a test, a combination of tests, or a diagnosis strategy to determine that a failure in some system element has occurred. In distributed diagnosis, every node maintains the diagnosis information about every other node in the system. Subbiah and Blough [5] and distributed diagnosis algorithms are executed on many or all the processors of the system simultaneously. Based on the outcomes of the test results (known as syndrome) at various. A survey on fault management in wireless sensor networks is presented in [6].

1.3. PMC model

A system consists of $\boldsymbol{n}$ units, denoted by the set $U = \{u_o, u_1 .. . u_{n-1}\}$. Each unit $u_i \epsilon U$ is assigned a particular subset of the remaining units to test usually based on the directed edges between them, and it is assumed that no unit test itself. The complete collection of tests by and on all the system nodes called the test assignment, is represented by a directed graph $G(U, E)$. Here each unit $ui \epsilon U$ is represented by a vertex and each edge $(u_i, u_j) \epsilon E$ represents a test link by which a test is carried out by unit *ui* on unit *uj*.

An outcome a_{ij} is associated with each edge $(u_i\ u_j) \epsilon E$, where a weight of *0 or*1, is assigned to *aij* if *ui* evaluates *uj* to be fault-free or faulty. Since the faults to be considered here are permanent, the outcome *aij* of a test is reliable if and only if *ui* is fault-free. With each unit $ui \epsilon U$, we associated all possible
sets $T(u_i) = \{ u_j : (u_i, u_j) \epsilon E \}$, $T^{-1}(u_i) = \{ u_j: (u_j, u_i) \epsilon E \}$, and the quantities $\boldsymbol{d_{in}(u_i) = |T^{-1}(u_i)|}$ and $\boldsymbol{d_{out}(u_i) = |T(u_i)|}$. The set of test outcomes of the system is called the *syndrome* and is denoted by *S*. Formally a syndrome is a mapping function:
$S: E \rightarrow (\mathbf{0}, \mathbf{l})$, defined such as for all $(u_i, u_j) \epsilon E, S(u_i, u_j) = a_{ij}$. We denote by $S(u_i)$ and $S^{-1}(u_i)$ the subsets of syndrome *S* corresponding, respectively, to tests carried out by unit u_i and by $T^{-1}(u_i)$, and are defined by:

$$\boldsymbol{S(u_i) = \{ S(u_i, u_j): u_j \epsilon T(u_i)\}}$$

$$\boldsymbol{S^{-1}(u_i) = \{(s(u_j, u_i): u_j \epsilon T^{-1}(u_i) \}}$$

A system is called *one-step t-fault diagnosable,* or simply *t-diagnosable,* if all faulty units within the system can be unambiguously identified provided the number of faulty units present does not exceed t[1].

2. FUZZY-GENETIC ALGORITHM FOR FAULT DETECTION

The design of the proposed fuzzy controlled genetic algorithm for fault detection is detailed here. Genetic algorithms are highly efficient stochastic search techniques based on analogy in biology in which a group of candidate solutions evolves through Darwinian principle of natural evolution

2.1. Fuzzy-Genetic Algorithm

Here, the proposed fuzzy-genetic algorithm (FGA) for fault detection is described in detail. The FGA includes all the basic features of a Basic Genetic Algorithm (BGA) [7]. The specific features that are incorporated in FGA are chromosome (genetic string) coding and decoding, elitism, and the mutation probability control using a fuzzy rule base. All these features are discussed below in detail.

2.2. Chromosome Coding and Decoding

In the proposed FGA, the chromosome or genetic string is chosen to consist of *n* bits where *n* is the number of tested components/sub-systems. Further, each bit 0/1 represents faulty or non-faulty respectively.

2.3. Initial Population

Initial population of size 10 is used and the length of each chromosome is the number of nodes in the network. The initial population is generated randomly by choosing 1 or 0 for each bit subject to the problem domain constraints. Here in each chromosome the number of 1's (faulty nodes) cannot be more than (N-1)/2 and it is assumed there is at least one faulty node and hence minimum one 1 should be there.

2.4. Fitness Function

Since the fitness function is the only guiding factor that influences the search mechanism, it should be carefully chosen so as to meet the multiple objectives. Fitness of chromosome is defined as the degree of correctness of an expected solution. The fitness function used for our genetic algorithm is based on a performance measure of a given chromosome. The measure

used in [8] is the probability that the diagnosis is correct. In this paper we use a similar performance measure that corresponds to the probability of correctness of a potential solution. Consider a chromosome *U*, let *F* and *S* be its corresponding fault set and one of its consistent syndromes, respectively. We denote by *w[i]*the i_{th} gene on chromosome U. The performance of a single gene can be formalized as follows:

$$f(v[i]) = (f_{out}(v[i]) + f_{in}(v[i]))/2 \quad (1)$$

where $$f_{out}(v[i])) = (|S(u_i) \cap S^*(u_i)|)/d_{out}(u_i) \quad (2)$$

$$f_{in}(v[i])) = (|S^{-1}(u_i) \cap (S^*)^{-1}(u_i)|)/d_{in}(u_i) \quad (3)$$

2.5. Reproduction and Crossover

In this proposed FGA, the genetic operators, reproduction and crossover, are implemented exactly in the same way as those in BGA. The selection mechanism is based on the Tournament selection. Two-point crossover is used with a crossover probability P.

2.6. Fuzzy-Adaptive Mutation and Elitism

Premature convergence that leads to same individuals in the whole population after some generations can be prevented by the use of mutation operator. Mutation diversifies the search by bringing new information. The mutation probability (p_m) should be sufficiently small and is critical. In this work, a fuzzy system based approach has been used to control the mutation probability at each generation.

The diversity of population can generally be measured by an assessment of the standard deviation (σ) and the convergence trend of the GA can be evaluated by observing the incremental change in the average fitness (Δf_{av}) of the population from fitness distribution of the population as shown in the Fig.1.

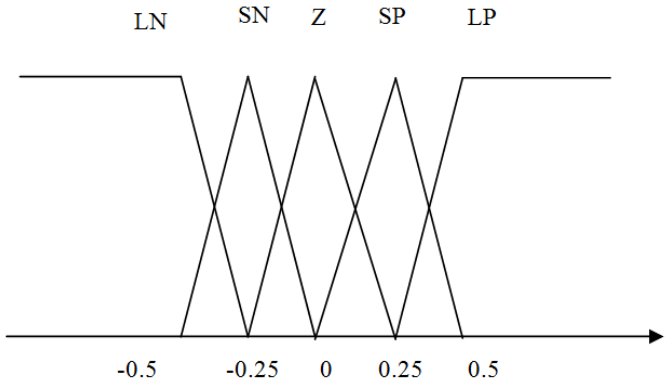


Fig. 1 Membership functions for Δf_{av}

The membership functions are LN: large negative, SN: small negative, Z: zero, S: small positive, L: large positive.

The central values of fuzzy singletons used in the output space of the fuzzy system in FGA are chosen to be 0.03, 0.06, 0.09, 0.12 and 0.15, respectively, for the different fuzzy sets, i.e., VS (very small), S (small), M (medium), L (large) and VL (very large) as shown in Fig.2.

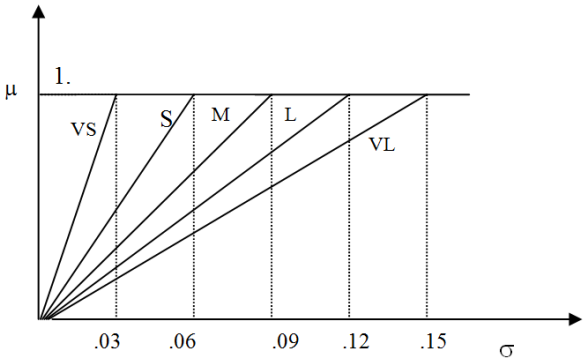


Fig. 2 Membership functions for standard deviation σ

Use of elitism operator is an elegant way of preserving the good solutions of early generations by ensuring the survival of the fittest individuals (strings) in each generation. This is can be easily achieved by passing the fittest string of the old generation into the new generation. A tentative solution is called non-dominated and Pareto optimal, if it cannot be replaced with another solution which improves an objective without worsening another one.

3. FAULT DETECTION PROCEDURE

In this work we used symmetric comparison model of PMC model with permanent faults and test results will be:

Fault-free node testing Fault-free node will give: 0
Fault-free node testing Faulty node will give: 1
Faulty node testing Fault-free node will give: 0/1
Faulty node testing Faulty node will give: 0/1

The string representation considered in this paper corresponds to the state (faulty or fault-free) of each unit. We use a binary vector as a chromosome (a potential solution) to represent the states of all units. Hence, the length of the vector is *N, if there are N units.* The chromosome (00011001) implies that the set of faulty units is $F = \{u_3, u_4, u_7\}$, i.e., those with a bit value 1**.** For example, for the network having 6 nodes and out of which nodes 4 and 6 are faulty the chromosome representing the network will be {010100}. As per the connections U1 can test $\{U_2, U_4, U_5\}$ giving the test values {1, 1, 0} respectively.

Similarly, U1 can be tested by $\{U_2, U_3, U_4, U_5\}$ giving values {0/1, 0, 0/1, 0} respectively. All such tests by each node on all other nodes and tests by all other nodes on each node as per the connections is known as **Real Test Syndrome**.

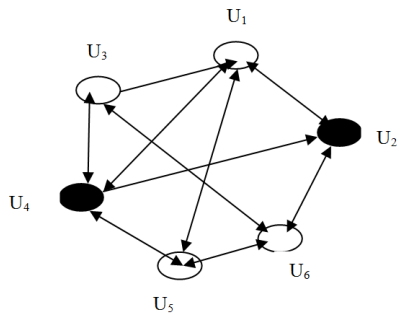


Fig. 3 Example network with 6 nodes. U_2 and U_4 Faulty.

For example consider the Fig.3, the test done by all nodes {U_1, U_2, U_3, U_4, U_5,U_6} on other nodes may yield {110 11 010 1111 010 100}. Here the test outcome done by faulty nodes is taken as 1 but in the actual case they can be 0 or 1. Similarly the tests done by all other nodes on {U_1, U_2, U_3, U_4, U_5, U_6} may yield the values {1010 111 10 111 010 100} respectively. Again, here the test outcome done by faulty nodes is taken as 1 but in the actual case they can be 0 or 1.

3.1. Steps of the Algorithm

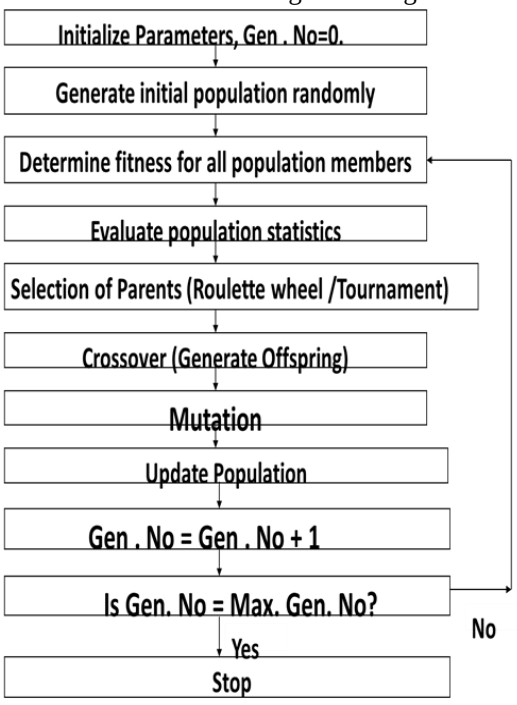
Step 1: A initial population of 10 chromosomes is generated randomly, which may look like:
001001, 010100, 110000,000010,100001, 000110, 01100, 010000, 100000,001000,010001, 100100
Here it can be observed that no chromosome has more than 2 faulty nodes and each one has at least one faulty node

Step 2: Each chromosome represents one network with bit 1 representing a faulty node and bit 0 representing non-faulty node. The fitness of each bit contributes to the overall fitness of the chromosome as per the equations (1), (2) and (3).

Step 3: In the process of reproduction, a new generation is created from old population, they have better fitness value than the previous generation. But the population size remains unchanged. Then supervised mutation is applied on chromosomes with mutation probability p_m.

In this work mutation is applied on the minimum fitness bit from all the fitness values of every bit in the chromosome. All the genetic operators: selection, crossover, mutation are applied to improve the fitness of population from generation to generation.

3.2 Flow-chart of Genetic algorithm is given below



The above Flowchart illustrates standard steps of a Genetic algorithm. After initial population is generated randomly, the fitness of each chromosome need to be determined according to the

stated Fitness function. Average fitness and variance in the fitness can be calculated which are used in Fuzzy control of the mutation. After that standard genetic operators like Selection of parents Crossover and Mutation will follow to generate new population.

4. SIMULATION RESULTS

Simulation is done in C++ as it is flexible to study the system behaviour in detail even though it is tedious. The details are presented below. The fitness function as discussed previously for guiding the FGA is chosen for the optimal results. The program is flexible enough to simulate for any size distributed system. Just by changing the max value to any desired size of the system, the program adjusts itself and ready for simulation. The population size is fixed (10) in the program but as a further study, can be modified for optimum population with suitable testing for optimality.

The initial test graph create is: 000011110000, i.e., with 4 faulty nodes and 8 non-faulty nodes and the systems has 12 nodes. With this information a 12×12 appropriate matrices are generated for the network.

The initial population of size 10 generated is given below:

Chromosome 0 : 0 0 0 0 1 1 0 0 0 0 0 1
Chromosome 1 : 0 1 0 0 0 1 0 1 1 0 1 0
Chromosome 2: 0 0 0 0 0 0 0 0 1 1 0 0
Chromosome 3: 1 0 0 1 0 0 0 1 0 1 0 0
Chromosome 4: 1 0 0 0 0 0 0 0 0 0 0 0
Chromosome 5: 0 1 0 0 0 0 0 0 1 0 1 0
Chromosome 6: 0 0 0 0 0 1 0 1 0 0 0 1
Chromosome 7: 0 0 0 0 0 1 0 0 0 0 0 0
Chromosome 8: 1 0 0 0 0 0 0 0 0 0 1 0
Chromosome 9: 0 0 0 0 0 0 0 0 0 1 0 1 0

The fitness of each chromosome is calculated and shown here:

fitness of chromosome:0=0.588599
fitness of chromosome:1=0.536107
Fitness of chromosome:2=0.493755
fitness of chromosome:3=0.514099
fitness of chromosome:4=0.52851
fitness of chromosome:5=0.446137
fitness of chromosome:6=0.648838
fitness of chromosome:7=0.61998
fitness of chromosome:8=0.455808
fitness of chromosome:9=0.486628

Chromosome number 6 is the fittest in this generation, and Mutation probability= 0.120392.

Population generated for generation Number= 10 is shown below:

Chromosome 0: 0 0 0 0 0 0 1 0 0 0 0 0
Chromosome 1: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 2: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 3: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 4: 0 0 0 0 0 1 1 0 0 0 0 0

Chromosome 5: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 6: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 7: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 8: 0 0 0 0 0 1 1 0 0 0 0 0
Chromosome 9: 0 0 0 0 0 1 1 0 0 0 0 0

In the generation number 10, bit number 7 and bit number 5 are rightly identified as the bits as least fit and second least fit bits. The fitness of chromosomes are given below:

fitness of chromosome:0=0.664567
fitness of chromosome:1=0.73297
fitness of chromosome:2=0.763112
fitness of chromosome:3=0.749917
fitness of chromosome:4=0.739335
fitness of chromosome:5=0.71751
fitness of chromosome:6=0.749917
fitness of chromosome:7=0.728753
fitness of chromosome:8=0.774107
fitness of chromosome:9=0.726736

So far best chromosome is 8. Mutation probability= 0.131241.

The solution is achieved in the generation no 27 and The population created in generation 27 shown below:

Chromosome 0: 0 0 0 0 1 1 1 1 0 0 0 0
Chromosome 1: 0 0 0 0 1 0 1 1 0 0 0 0
Chromosome 2: 1 0 0 0 1 1 1 1 0 0 0 0
Chromosome 3: 0 0 0 0 1 0 1 1 0 0 0 0
Chromosome 4: 0 0 0 0 1 0 1 1 0 0 0 0
Chromosome 5: 0 0 0 0 1 0 1 1 0 0 0 0
Chromosome 6: 0 0 0 0 1 1 1 1 0 0 0 0
Chromosome 7: 0 0 0 0 1 0 1 1 0 0 0 0
Chromosome 8: 1 0 0 0 1 1 1 1 0 0 0 0
Chromosome 9: 0 0 0 1 0 1 1 0 0 0 0 0

The fitness of the chromosomes in the generation 27 is shown below:

fitness of chromozome:0=0.806959
fitness of chromozome:1=0.750624
fitness of chromozome:2=0.668795
fitness of chromozome:3=0.788058
fitness of chromozome:4=0.750227
Fitness of chromozome:5=0.771325
fitness of chromozome:6=0.771526
fitness of chromozome:7=0.748178
fitness of chromozome:8=0.693613
fitness of chromozome:9=0.748259

The best chromosome in this is generation is the first one and also observe that this is also our test network. Also note that the fitness of this chromosome is not 1. The reason for this is obvious that the test outcomes of faulty nodes are arbitrary (0/1). Hence few bits can be different even for same network.

The simulation is done with both BGA and FGA and convergence for best fitness found to be faster with Fuzzy controlled mutation (FGA) compared to fixed mutation (BGA) as is shown Fig.4.

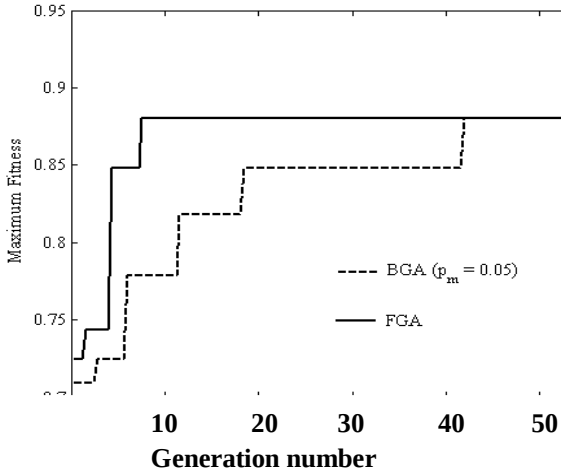


Fig. 4 Comparison of evolution of maximum fitness between FGA and BGA {mutation probability = 0.05}

5. CONCLUSIONS & FUTURE WORK

In this paper we presented a robust stochastic search method based on genetic algorithm with Fuzzy controlled mutation and Tournament selection. In this work the crossover operator has taken into account the constraints of t-diagnosable systems and in future work it has to be done with mutation also. The program is implemented in C++ and was tested with random test graphs. The results are satisfactory and demonstrated the efficiency of genetic operators used for this study. The authors are confident that a more rational approach for deciding of population size and crossover operator will be developed in future and the scope of study can be extended.

REFERENCES

[1] F.P. Preparata, G. Metze, and R.T. Chien. On the connection assignment of diagnosable systems. IEEE Trans.on Electron.Comput.,16(6), 12 1967.

[2] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski. A fault detection service for wide area distributed computations. In Proc. 7th IEEE Symp. on High Performance Distributed Computing, pages 268-278, July 1998

[3] Banarjee, N. and Khilar, P., "Performance analysis of distributed intermittent fault diagnosis in wireless sensor network," (India), NIT,surathkal, August 2010.

[4] Dejan P. Jovanovi´c, "Fault Detection in Complex and Distributed Systems" A thesis submitted for the degree of Doctor of Philosophy at The University of Queensland in 2014

[5] Lilia. P, Qi. H, A Survey of Fault Management in Wireless Sensor Networks. Journal of Network and Systems Management, vol. 15, no.2, pp. 171 -190, 2007.

[6] Gajendra Sharma and Ankita Yadav,"Framework for Detecting Fault in Real Time Distributed System" Trends in Scientific and Technical research, Juniper Publishers,October 10, 2018

[7] D. M. Blough, G. E Sullivan, and G. M.Masson. Efficient diagnosis of mutltiprocessor systems under probabilistic models. 41(9):1126-1136,1992. IEEE Trans. on Computers.

Author

Dr. Krishna Prasad has 30 years' experience as Professor in Computer Science, guiding Thesis and Projects of Post-Graduate and Ph.D. level students. M.Sc. program Coordinator for De Montfort University, UK in Singapore and Malaysia. International exposure for 20 years, working in Malaysia, Singapore, Bosnia and Herzegovina, Sri Lanka, Ethiopia and India.

NUMBER OF PUBLICATIONS: 37

International Journals – 13 (2 papers in IEEE Transactions), International Conferences – 24