GA-Based Fault Diagnosis Algorithms For Distributed Systems

Swastisudha Punyatoya

(Roll No: 607CS004)

Department of Computer Science and Engineering

National Institute of Technology, Rourkela

Rourkela-769 008, Orissa, India

July, 2011.

GA-Based Fault Diagnosis Algorithms For Distributed Systems

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Technology

(Research)

in

Computer Science and Engineering

by

Swastisudha Punyatoya

(Roll No: 607CS004)

under the guidance of

Dr. Pabitra Mohan Khilar

Department of Computer Science and Engineering

National Institute of Technology, Rourkela

Rourkela-769 008, Orissa, India

July, 2011.

Dedicated to my parents

Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela-769 008, Orissa, India.

Certificate

This is to certify that the work in the thesis entitled **"GA-Based Fault Diagnosis Algorithms For Distributed Systems"** submitted by ***Swastisudha punyatoya*** is a record of an original research work carried out by her under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology (Research) in Computer Science and Engineering, National Institute of Technology, Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr.Pabitra Mohan Khilar
Assistant Professor
Department of CSE
National Institute of Technology
Rourkela-769008

Department of CSE
National Institute of Technology
Rourkela-769008

Place: NIT, Rourkela
Date: 21 - 07 - 2011

Acknowledgment

First of all, I would like to express my deep sense of respect and gratitude towards my advisor and guide Prof (Dr.) Pabitra Mohan Khilar, who has been the guiding force behind this work. I want to thank him for introducing me to the field of Distributed System level Fault Diagnosis and giving me the opportunity to work under him. His undivided faith in this topic and ability to bring out the best of analytical and practical skills in people has been invaluable in tough periods. Without his invaluable advice and assistance it would not have been possible for me to complete this thesis. I am greatly indebted to him for his constant encouragement and invaluable advice in every aspect of my academic life. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

Secondly, I would like to thank Prof. B. D. Sahoo, Prof. A.K.Turuk, Head CSE; Prof. S. K. Patra for serving on my Masters Scrutiny Committee.

I wish to thank all faculty members and secretarial staff of the CSE Department for their sympathetic cooperation.

During my studies at N.I.T. Rourkela, I made many friends. I would like to thank them all, for all the great moments I had with them, especially Saubhagya, Alekha, Honey, Swati, Baikuntha, Suraj, Devi, Amita Di, Sonali, Shaida, Pushpendra sir, Arunanshu sir for their selfless help and cooperation in my tough periods.

When I look back at my accomplishments in life, I can see a clear trace of my family's concerns and devotion everywhere. I am forever indebted to my parents and my sisters for their encouragement and support.

Swastisudha Punyatoya

Abstract

Distributed Systems are becoming very popular day-by-day due to their applications in various fields such as electronic automotives, remote environment control like underwater sensor network, K-connected networks. Faults may affect the nodes of the system at any time. So diagnosing the faulty nodes in the distributed system is an worst necessity to make the system more reliable and efficient. This thesis describes about different types of faults, system and fault model, those are already in literature. As the evolutionary approaches give optimum outcome than probabilistic approaches, we have developed Genetic algorithm based fault diagnosis algorithm which provides better result than other fault diagnosis algorithms. The GA-based fault diagnosis algorithm has worked upon different types of faults like permanent as well as intermittent faults in a K-connected system. Simulation results demonstrate that the proposed Genetic Algorithm Based Permanent Fault Diagnosis Algorithm(GAPFDA) and Genetic Algorithm Based Intermittent Fault Diagnosis Algorithm(GAIFDA) decreases the number of messages transferred and the time needed to diagnose the faulty nodes in a K-connected distributed system. The decrease in CPU time and number of steps are due to the application of supervised mutation in the fault diagnosis algorithms. The time complexity and message complexity of GAPFDA are analyzed as O(n*P*K*ng) and O(n*K) respectively. The time complexity and message complexity of GAIFDA are O(r*n*P*K*ng) and O(r*n*K) respectively, where ′n′ is the number of nodes, ′P′ is the population size, ′K′ is the connectivity of the network, ′ng′ is the number of generations (steps), ′r′ is the number of rounds. Along with the design of fault diagnosis algorithm of O(r*k) for diagnosing the transient-leading-to-permanent faults in the actuators of a k-fault tolerant Fly-by-wire(FBW) system, an efficient scheduling algorithm has been developed to schedule different tasks of a FBW system, here ′r′ denotes the number of rounds. The proposed algorithm for scheduling the task graphs of a multi-rate FBW system demonstrates that, maximization in microcontroller′s execution period reduces the number of microcontrollers needed for performing diagnosis.

Dissemination of Work

Published

1. Swastisudha Punyatoya , Pabitra Mohan Khilar , ”Distributed Microcontroller-based Actuator Fault Diagnosis in Multi-rate fly-by-wire system”, proceedings of the 12th International Conference on Information Technology(ICIT-2009),McGraw Hill, pp.17-22,2009.

2. Pabitra Mohan Khilar, Swastisudha Punyatoya, ” A survey on System Level Fault Diagnosis in Distributed Networks”. proceedings of the 12th International Conference on Information Technology(ICIT-2009),McGrawHill,pp.213-217, 2009

3. Swastisudha Punyatoya, Pabitra Mohan Khilar , ”A Novel Fault Diagnosis Algorithm For K-Connected Distributed Clusters”,proceedings of IEEE sponsored International Conference on Industrial Electronics, Control and Robotics(IECR-2010),2010.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Distributed computing systems are becoming popular day-by-day due to their various applications in both computational and communication intensive tasks. Several definitions and view points are stated in the literature about the distributed systems. Coulouris defines a distributed system as "A system in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing" [8]. Tanenbaum defines it as "A collection of independent computers that appear to the users of the system as a single computer" [42]. Leslie Lamport, a famous researcher on timing, message ordering and clock synchronization in distributed systems once said that "A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed", reflecting on the huge number of challenges faced by distributed system designers. The nodes in a distributed system are connected by an interconnection network. The communication in between nodes in the distributed system takes place by exchanging messages. Therefore these distributed systems are commonly known as message passing distributed systems with contrast to shared memory communication, which is extensively followed in various multiprocessor and parallel system. Some of the distributed systems such as wireless ad-hoc networks follow an arbitrary network topology, where the nodes are randomly deployed in the environment. Another kind of distributed systems such as electronic automotive systems are extensively used in real time applications.

However the components of the systems are subjected to various kind of faults. The occurrence of faults not only affect the normal system functions but also causes the degradation in performance. Due to this fact, the failure handling has been a key research area since the development of these distributed systems. In some distributed systems such as Steer-By-Wire(SBW) system, Fly-By-Wire(FBW) system, Break-By-Wire(BBW) system , if faults are not handled within a specific deadline, the system leads to catastrophic failure. Therefore, these distributed systems are called safety-critical real time distributed embedded systems.

The different types of faults[3] which usually occur in a distributed system are as follows:

- Permanent Fault
- Transient Fault
- Intermittent Fault
- Byzantine Fault

The description of these faults have been discussed in section 1.3 of this chapter. In precise, fault avoidance and fault tolerance are two important techniques for handling faults. Fault avoidance in distributed systems are effective upto certain extent but is least efficient in real life application. However producing the correct outcome from the system inspite of the presence of faulty nodes is rather a better approach. The system that produces correct outcome by counteracting the effect of fault is known as fault tolerant system. 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(defined in chapter 3) and distributed embedded system(Fly-By-Wire system), as this technique is suitable for diagnosing faulty nodes of distributed system of small to large size. It is noted that, the system level fault diagnosis has been accepted as the universal method for fault diagnosis [38]. In method 2, the faulty nodes are masked by using certain voting on the replication of same units. Thereafter, the system is reconfigured for carrying out normal functions.

This chapter is organized as follows. Section 1.1 gives the introduction of the thesis work , section 1.2 discusses the factors that motivate us to carry out the research work . Section 1.3 describes the types of faults, system level fault diagnosis technique and the different types of models used in system level fault diagnosis . Application of different evolutionary approaches in system level fault diagnosis is described in section 1.4 , section 1.5 consists of the objectives of the thesis work . The organization of the thesis is described in section 1.6, followed by the conclusion of the chapter in section 1.7.

1.2 Motivation

The distributed computing history is as old as computer network. Primary motivation for building such system is to share geographically distributed computing resources. We have seen great strides over the past five years in the key area of distributed system technology. Distributed embedded systems are being realized as safety critical realtime distributed system and they are used in applications such as FBW system, SBW system, BBW systems. Since the occurrence of faults in the distributed systems affect the dependability of the system, the following factors motivated us to undertake the proposed research work:

- Study of the nature of different types of faults and different types of fault models is necessary for developing diagnosis algorithm for distributed systems.
- Application of Genetic Algorithm is considered as a new, effective and efficient approach for fault diagnosis in distributed systems.
- The topology of small and large scale distributed systems such as wireless adhoc network are generally K-connected.
- System level diagnosis is indispensible for safety critical real time distributed system like Fly-By-Wire, Break-By-Wire system to provide a robust system.
- The occurrence of permanent and intermittent faults are more likely in various nodes of the K-connected systems and transient , subsequently leading to permanent fault are more likely to occur in real time embedded system such as Fly-By-Wire system.

1.3 Fault Diagnosis in Distributed System

1.3.1 Types of faults

Faults are categorized depending upon their behavior.

1)Permanent fault:When the node of a distributed system is subjected to permanent

fault, it responds with some erroneous outcome.
2)Intermittent Fault: The unit of the diagnosable distributed system, which is intermittent faulty, sometimes behaves as a faulty node and sometimes behaves as fault free.
3)Transient fault: This type of fault occurs due to environmental parameters like change in temperature, pressure etc. The existing period of transient fault is not long like intermittent fault, it affects the units for a small period and the effect vanishes after that period.
4)Byzantine fault: A byzantine fault is an arbitrary fault, which is the superset of all the above types of fault.

1.3.2 System Level Fault Diagnosis

A diagnosable distributed system D consists of n units,Suppose set N=$\{n_1, n_2, n_3, n_4, ..., n_n\}$, the nodes in the set N are connected to each other by communication links. Each unit n_i of the set N, tests the subset of the set $(N - n_i)$ but not to itself. A test link t_{ij} defines that the unit n_j is tested by the unit n_i. Complete collection of tests in N is called test connection assignment and is represented by a directed graph G(N,E), where each unit n_i is represented as a vertex in the graph and the test link t_{ij} represents the edge connecting to the tester node and tested node.
Application of system level diagnosis in the system D consists of two phases.
In the 1st phase, *the type and nature of fault that is to be diagnosed is studied, the number of faults that is to be diagnosed and the model used for test result interpretation is decided* i.e whether the fault is permanent or intermittent or transient or byzantine or the fault type is hybrid, whether the model to be used is PMC Model or Comparison Model .
In 2nd phase of System level fault diagnosis, *the process for identification of faulty nodes from the testoutcome* is developed. The fault diagnosis algorithm is applied on the system to identify the faulty nodes and faultfree nodes from the test outcome obtained in the 1st phase.

1.3.3 Types of model used for test result interpretation

During the first phase of system level fault diagnosis, the units of the system those perform testing operation are treated as tester nodes and the nodes, on which the tests are applied by the tester nodes, known as tested nodes. In SFD, the tests are performed by sending a signal to the tested node and waiting for the response. The rules followed to get the testoutcomes in tabular format is termed as model.

PMC model

The first theoretical model was introduced by Preperta, Merze, Chein in the year 1967, known as PMC model. Again the PMC model is categorized into 2 models. These are asymmetric PMC model and symmetric PMC model. In asymmetric PMC model, the test outcome is 0, if both the tester and tested nodes are faultfree.The test outcome is 1, if both the tested unit and tester units are faulty. The test outcome is unreliable(either 0 or 1), if the tester unit is faulty and the tested unit is faultfree. In case of symmetric PMC model, the result is unpredictable when the tester node is faulty. When the tester node is faultfree and the tested node is faulty in symmetric PMC model then the outcome of the test is 1. The test outcome is 0, if both tested node and tester node are faultfree. Table 1.1 shows the asymmetric PMC model. Table 1.2 shows the symmetric PMC model.

Table 1.1: Test Result Interpretation using Asymmetric PMC model

Tester	Tested	Outcome
Faultfree	Faultfree	0
Faultfree	Faulty	1
Faulty	Faultfree	0/1
Faulty	Faulty	1

Comparison model

The comparison model was introduced by Malek. Basically in comparison model, either one centralized comparator node exists that performs the comparison among

Table 1.2: Test Result Interpretation using symmetric PMC model

Tester	Tested	Outcome
Faultfree	Faultfree	0
Faultfree	Faulty	1
Faulty	Faultfree	0/1
Faulty	Faulty	0/1

the outcomes generated by the pair of the units, when the same job is assigned to the pair of processors, or one of the units performs the work of a comparator from the pair of units. Comparison model is of 2 types. These are asymmetric Comparison model and symmetric Comparison model. In asymmetric Comparison model, the comparison outcome is 0, if both the compared and comparator nodes are faultfree. The test outcome is 1, if the compared unit is faulty. The test outcome is unreliable(either 0 or 1), if the comparator unit is faulty and the compared unit is faultfree.

In case of symmetric Comparison model,the result is unpredictable when the comparater node is faulty but when the comparater node is faultfree, the outcome of the comparison is 1 and the compared node is faulty and the outcome is 0 if the compared node is faultfree. Table 1.3 shows the comparison result interpretation using asymmetric comparison model. Table 1.4 shows the comparison result interpretation using symmetric comparison model.

Table 1.3: Test Result Interpretation using Asymmetric Comparison model

Comparater	Compared	Outcome
Faultfree	Faultfree	0
Faultfree	Faulty	1
Faulty	Faultfree	0/1
Faulty	Faulty	1

Table 1.4: Test Result Interpretation using symmetric Comparison model

Comparater	Compared	Outcome
Faultfree	Faultfree	0
Faultfree	Faulty	1
Faulty	Faultfree	0/1
Faulty	Faulty	0/1

1.4 Evolutionary approaches and their applications in System level fault diagnosis(SFD)

Genetic Algorithms have become popular due to the development of robust stochastic searching algorithm for various optimization problem.As compared In these algorithms, it is necessary to represent the search individuals and generate the initial population by establishing the fitness function based on the genetic operators such as reproduction, crossover and mutation. In fact the use of genetic algorithm is a novel approach to the problem of system level fault diagnosis in various recently emerging distributed computing systems, following an arbitrary network topology.

This thesis proposes 2 algorithms to solve the diagnosis problem using genetic algorithms. The search space is represented in a binary vector of length n, where each bit indicates the status(faulty or faultfree) of its corresponding unit. Genetic operators are adopted to the context of distributed system level diagnosis. The fitness function is used as the objective function to optimize the performance with less number of iterations of fault diagnosis algorithm. In the problem of fault diagnosis in the distributed system, the estimation time for the fitness function is greatly in excess of the time taken to perform any genetic operations. Thats why the proposed algorithms are efficient by optimizing these operations. Infact, the time taken to compute the fitness values for all generations is considerably low. The efficiency of these methods have been illustrated in the thesis by considering permanent and intermittent faults in various nodes of the system. Instead of random mutation, the proposed algorithm uses supervised mutation, which allows to construct an efficient mutation process. Both crossover and mutation operators were developed to take into account the necessary conditions for a K-connected distributed system and guarantees that the newly generated strings are legal.

The time needed to detect the faulty nodes from the set of nodes in a K-connected network using GA-based approach depends mainly upon the time taken for finding out the fitness of the chromosomes in the population at every generation. The time taken to perform repeated selection and supervised mutation is negligible as compared to the time taken to find the fitness of each chromosome. As all the nodes in the network are not participating to detect the faulty nodes in the network, the number of messages that is exchanged between nodes across the network is lesser than that of other fault diagnosis algorithms. The GA-based approach therefore leads to less traffic in the K-connected network due to which more bandwidth is available for application specific tasks. It is experimentally verified that the time taken to diagnose the faulty nodes using the GA-based approach varies within a certain range for a specific size network. Searching the accurate solution from a small solution pool of constant size is a better approach than the formation of huge network traffic by sending and receiving of diagnosis related information. It can be also noted that many small scale distributed networks exist where the nodes are energy or power constrained. In such type of distributed networks GA-Based approach consume less energy as compared to that of other fault diagnosis algorithms.

1.5 Objective

Motivated by the need of permanent, intermittent and transient fault diagnosis in distributed systems, the following objectives are undertaken in this work.

- To develop a GA-based distributed diagnosis algorithm for diagnosing permanent faulty nodes in a K-connected distributed system.
- To develop a GA-based distributed diagnosis algorithm for diagnosing intermittent faulty nodes in a K-connected distributed system.

- To develop a fault diagnosis algorithm for a safety critical Fly-By-Wire system to diagnose transient-leading-to-permanent fault under time and resource constraints.
- To develop a real time scheduling algorithm for scheduling actuator control and diagnosis tasks of a multi-rate FBW system.

1.6 Organization of the thesis

The thesis is organized as follows.
In *Chapter 1*, a brief introduction to fault diagnosis in distributed system is presented, followed by motivation , contribution and objective of the thesis. The genetic algorithm approach to the fault diagnosis and different types of faults that may occur in distributed system have been presented.
In *Chapter 2*, a brief description of exhaustive survey of the works, those are already available in the literature have been presented.
Chapter 3 describes a GA-based fault diagnosis algorithm for diagnosing permanently faulty nodes in a K-connected distributed system .
In *Chapter 4*, a GA-based intermittent fault diagnosis algorithm is presented.
In *Chapter 5*, a fault diagnosis algorithm for a multi-rate fly-by -wire(FBW) system is designed. Here the fault affecting the actuators of FBW system is taken into consideration. Along with the fault diagnosis algorithm, a real time task scheduling algorithm is proposed to schedule the actuator control and diagnosis tasks within a designer specified deadline.
Chapter 6 contains the concluding remarks, description about the problems which is still an open problem for us in the future.

1.7 Conclusion

In this chapter we presented the importance of fault diagnosis in distributed systems. System level fault diagnosis is one of the most popular technique among different fault diagnosis techniques. The factors those motivated us to do the contributory work are also described. Different types of fault such as intermittent fault, transient fault, byzantyne fault and permanent fault have been described. The types of model used for test result interpretation such as PMC and Comparison model are presented. Finally this chapter is concluded with the objectives and organization of the thesis.

Chapter 2

Literature review

2.1 Introduction

Fault diagnosis in a distributed system is an important problem and needs necessary steps to devise algorithms to diagnose faulty system components. System level diagnosis serves as a major tool for providing universal diagnosis solutions for such systems. System-Level Diagnosis, which was introduced by Preparata, Metze and Chien [31] in 1967, models the system by a graph and uses a diagnostic algorithm to determine the status of each unit in multiprocessor systems [16] [23] and distributed systems [21]. It assumes a system composed by a set of units, where each unit can be either faulty or fault-free. Units test each other exploiting the available system interconnections. Here the units can be any system component that can diagnose the rest of the component or can be diagnosed by any other units.

The diagnosis can be either centralized or distributed. In centralized diagnosis, it is assumed that a centralized reliable node exist that collect the test results and maintains the diagnosis information. The main bottleneck of this diagnosis is that it causes a single point of failure and thus unacceptable for a large class of distributed systems such as wireless adhoc networks. In distributed diagnosis, every node maintains the diagnosis information about every other node in the system [40]. 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 nodes in a distributed system, the diagnosis can be achieved by every fault-free node. In the past, the target application of the system level diagnosis theory is the diagnosis of massive parallel processing systems such as multiprocessor and wafer-scale VLSI systems. In these applications, the system is represented by a regular or quasi-regular diagnostic graph, a fact that makes the classical results of system-level diagnosis inadequate. The application of SLD theory to the diagnosis of communication networks has been proposed in [36]. The computer networks such as wireless adhoc network and distributed embedded systems are two major emerging types of networks where these diagnosis techniques are useful. The diagnosis is based on testing the units of the network. The results of the tests are used to infer about the status of a unit. The units in the network are either homogeneous or heterogeneous but are assumed to be able to test other nodes in order to determine their fault status. While the existing model makes no assumption about the nature of the tests and how they are performed, the fault coverage of the tests is supposed to be 100 percent. It implies that the tests are sufficient to determine the fault status of a unit. The rest of the chapter is organized as follows. Section 2.2 , describes the characterization of distributed system level diagnosis, In section 2.3 various diagnosis approaches and evaluation methods are described. Section 2.4 describes the work that have already being done on system level diagnosis using various evolutionary approaches. The summary of the chapter is given in section 2.5.

2.2 Characterization of distributed diagnosis

The distributed diagnosis algorithms are characterized by their system model, fault model, diagnostic model, the fault environment and the performance measures for evaluation. The system model specifies the hardware and software components and the topology of the network. For example, the nodes in a hypercube or completely connected network constitute a regular network topology, the nodes in wireless adhoc network and Internet form an arbitrary network topology where the nodes are connected arbitrarily with each other. The behavior of a faulty component specifies the fault model in a distributed system. For example, a crash faulty node does not respond whereas a value faulty node responds with an incorrect value. Based on the duration that a fault persists in a node, the faults can be classified into permanent, transient, intermittent and Byzantine [19]. A permanent faulty node does not respond when exercised on a node. A transient fault in a system node persists only for a small duration i.e., occurs once and disappears. An intermittently faulty unit sometimes gives correct results and sometimes erroneous results. A lot of work on system-level diagnosis has focused on probabilistic methods, which can diagnose intermittently faulty processing nodes and can be applied in general situations on general interconnection networks. An intermittently faulty node may sometime give correct results which is difficult to capture. Several tests and diagnosis rounds are necessary to capture the intermittently faulty nodes. A solution for the intermittent fault diagnosis has been proposed in [35, 3]. A Byzantine faulty node behaves arbitrarily. However it is possible to specify a limited Byzantine behavior. The diagnostic tests are executed in order to detect the faulty components. There are generally two ways to detect a fault in a network: (i) test-based and (ii) heartbeat based testing mechanism. The test-based mechanism involves the two messages(request and reply message) to detect a fault whereas heartbeat based testing mechanism relies on the message transfer between every pair of nodes. Irrespective of the type of testing mechanism a diagnosis algorithm uses, the message transmission is used as the means to detect faulty components. Since the tests are difficult to obtain in practice, the comparison based diagnostic model has been suitable for the practical distributed systems.

System-level diagnosis is intended to identify faulty nodes of a distributed system such as multiprocessor and multi computer system. Multiprocessor systems are tightly coupled system where nodes communicate each other by shared memory. Multicomputer systems are message passing systems where the nodes communicate by passing messages. Both the invalidation(PMC) and comparison based diagnosis models are applicable to both multiprocessor as well as multi computer systems.

As far as diagnosis is concerned, two issues are involved: i) how every node can acquire the information necessary for the fault location and ii) how can every node identify the faulty nodes at a low time overhead. For the first issue, there are two different classes of models: the test-based models and the comparison-based models. Under a testbased model, every node of a system is assigned to test some other nodes [31]. Under a comparison based model, the response of two nodes to the same task is compared [44, 27, 22] and the collection of all of the comparison results (the comparison syndrome) is analyzed for locating the faulty nodes. The second issue deals with the efficient algorithm for accomplishing the diagnosis function.

To perform automatic fault diagnosis of graph G, each unit u_i belongs to V must be capable of testing units of a particular subset of the remaining units in G. The complete collection of ordered pairs of units (u_i,u_j) where u_i (the tester unit) tests u_j (the tested unit) in G is represented by a simple directed graph referred to as test digraph. Let D = (V, E) be the test digraph. The result of the test of u_j by u_i is called test outcome and may be 0 or 1 corresponding to tester ui evaluating u_j as fault free or faulty, respectively. The test outcome may be arbitrary (either 0 or 1) regardless of the fault status of u_i and u_j when the outcome is unpredictable. The assumption about the inability of a faulty tester unit to correctly test other units is known as the fault invalidation model. The well known fault invalidation models in the theory of system-level diagnosis are the symmetric PMC [31] and asymmetric BGM [16] invalidation models. In symmetric invalidation model, it is assumed that when tests performed by faulty units can produce any result. Whereas in case of asymmetric invalidation model, it is assumed that faulty units always fail tests, even if units can invalidate the test are faulty. Both the models have importance in large class distributed systems based on a combination of faulty units some of which behave according to the PMC model and the others which behave according to the BGM model.

In comparison based model, the diagnosis tasks are assigned to various nodes in the system and results of the diagnosis tasks are compared. Based on the agreement in the test results, every node in the system achieves diagnosis. The comparator unit can be either one of the testing or tested unit, or a third unit. This method is supposed to be most practical method of diagnosis and has been proposed by many researchers in this area. The summary of the works based on comparison model is as follows: The MM* model proposed by Maeng and Malek [22] is a realistic comparison-based model where a node is a comparator of two nodes if and only if the comparator is connected to them through direct communication links. The MM* model reasonably assumes that an agreement in the responses of two nodes being compared implies that these two nodes are fault-free provided the comparator is fault-free, whereas a disagreement implies that at least one of the three relevant nodes is faulty. If the results of all comparison are broadcast to all the connected units, the resulting system can be described by the Broadcast Comparison model [6]. When a fault-free node compares the outputs produced by two faulty nodes, the result always show a difference (asymmetric comparison model) or same (symmetric comparison model). The comparisonbased system-level diagnosis for both hypercube and its variations can be found in [10, 43, 20]. The comparison based distributed diagnosis has been recently used for diagnosis of wireless adhoc networks and distributed embedded systems [33, 34, 45]. In fact, the comparison based distributed makes it suitable for a use in the diagnosis of Wireless Ad Hoc Networks and distributed embedded systems. When the automatic diagnosis process is applied to these systems, they are called t-diagnosable system. A t-diagnosable system is one in which all faulty nodes can be identified correctly provided no more that ’t’ nodes fail simultaneously.

2.3 Description of different fault diagnosis approaches and evaluation methods

2.3.1 Phases of distributed system level fault diagnosis

The main objective of system-level diagnosis is to identify which units of the system are faulty and which are fault-free [10]. Several models for system-level diagnosis can be found in the literature. Based on the system-level diagnosis theory, Kuhl and Reddy

[21] 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. The detection may be associated with built-in tests (BITs) and may actually be the design requirement for BIT. Localization means that a faulty system element has been identified by all other nodes in the network. This is also known as diagnosis phase. Recovery is the regain of original status from its fault status. These three phases of the diagnosis has been referred to as testing, collecting and dissemination of diagnosis information [26].

Another diagnosis approach known as adaptive distributed system-level diagnosis (ADSD) algorithm is, at the same time, distributed and adaptive. Each node must be tested only one time per testing interval. All fault-free nodes achieve consistent diagnosis in at most N testing rounds. There is no limit on the number of faulty nodes for fault-free nodes to diagnose the system [5, 28, 45]. In adaptive distributed diagnosis, the algorithms operate in a distributed fashion and adapt their testing assignment to the prevailing fault situation. Diagnosis occurs locally at each unit and evaluates the system units as faulty or fault-free.

2.3.2 Evaluation Methods

The metrics such as CPU time needed to execute the diagnosis algorithm, the number of iterations required to diagnose the faulty nodes and the number of messages exchanged(message complexity) are used to evaluate the diagnosis algorithms analytically as well as through simulation. The diagnostic latency is the time elapsed between the occurrences of a fault means every newly recovered node gains the status about the entire system within a bounded time. Accuracy means the developed diagnosis algorithm diagnoses the nodes correctly within bounded time. These properties can be satisfied in a synchronous system but difficult to adapt this approach for asynchronous system. This is due to the fact that the task execution time and communication delay in a synchronous system is bounded whereas there is no upper limit either in execution time or delay in communication in an asynchronous system. The analytical methods for evaluating distributed diagnosis algorithms largely depend on the formal analysis, probability theory, finite automata, graph theory and evolutionary methods. Many times, the researchers find difficulty in formalizing the notion of diagnosis algorithms using these methods. The validation of algorithms through simulation is an alternative to study the behavior of algorithms and for verifying the analysis and correctness proof of the proposed algorithms. Most of the diagnosis results are based on graph theory [17].

2.4 Application of Genetic Algorithm in System level fault diagnosis

2.4.1 Evolutionary approaches

Knowledge-based information systems are designed to mimic the performance of biological systems [2] .Figure 2.1 shows the hierarchy of Knowledge-based Information Systems. As shown in the diagram, there are two approaches of the knowledge-based information systems such as approximate reasoning and optimization approaches. As the genetic algorithms are based on optimization approaches and that leads to accurate diagnosis results, the genetic algorithm based fault diagnosis algorithms are followed in this work. Also, the evolutionary computing algorithms are applied for efficient fault diagnosis of distributed systems as compared to the traditional method of fault diagnosis.

2.4.2 Genetic Algorithm

Genetic algorithms [2] are search methods that employ processes found in natural biological evolution. These algorithms search or operate on a given population of potential solutions to find those, that approaches some specification or criteria. To do this, the algorithm applies the principle of survival of the fittest to find better and better approximations. At each generation, a new set of approximations is created by the process of selecting individual potential solutions (individuals) according to their

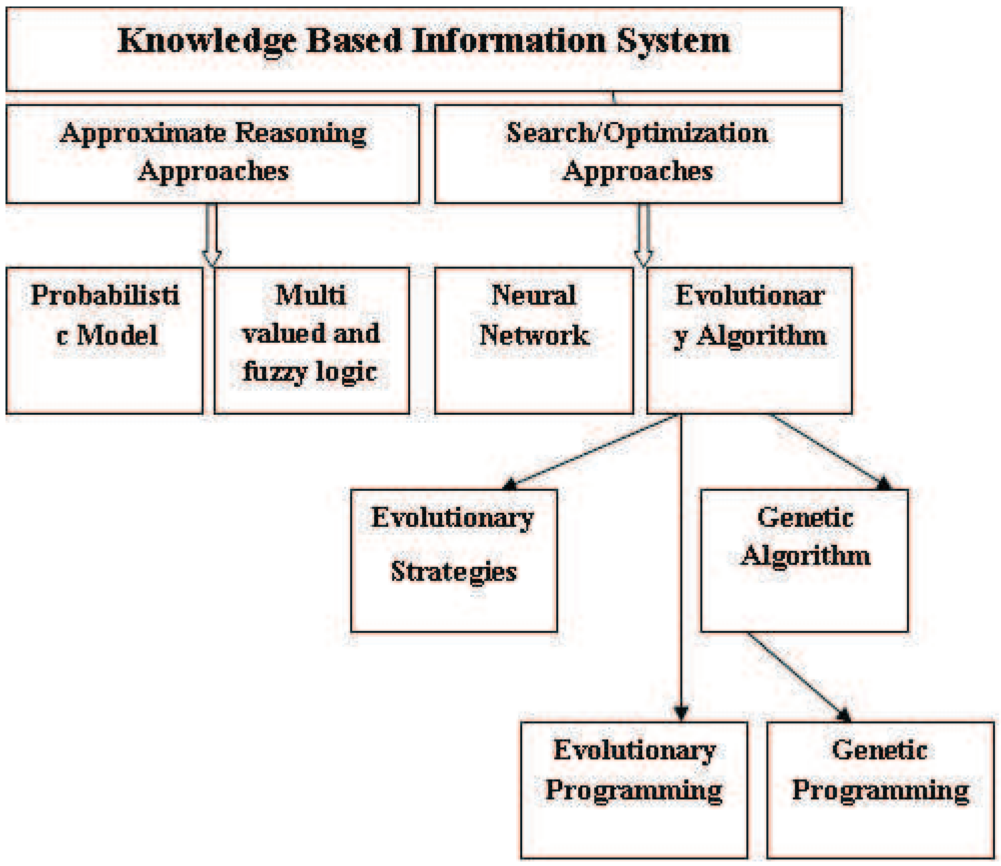


Figure 2.1: Hierarchy of KnowledgeBased Information Systems(Evolutionary computing).

level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals that they were created from, just as in natural adaptation.

The Genetic algorithms generally include four fundamental operators. (i) Selection (ii)Reproduction (iii)Crossover (iv)Mutation. In selection operation, a particular solution reaching towards the global optima are selected and new population is reproduced. The crossover operation are applied on each pair of possible solutions in the new population. The mutation operation is applied on each possible solution in the solution space in order to generate different solutions in the solution space.

The initial population is generated. If a special condition is satisfied, the optimal solution is found. If a special condition is not satisfied, selection operation is taken place and the individuals having highest fitness(best individuals) are selected for new population. Fitness is the function that measures the quality and optimality of each possible solution in the solution space. The solution having highest fitness can survive in the new generation. Crossover and mutation operations are applied in the new generation to produce new and more efficient individuals. The selection of high fitness individuals and the application of crossover and mutation on those individuals will be taken place, until a given condition is satisfied and the exact solution is found. The steps followed by genetic algorithm are shown in Figure 2.2.

2.4.3 GA-based System Level Fault diagnosis

There are various analytical model, developed to solve the problem of system level fault diagnosis, One of the suitable approach is the use of evolutionary approach to solve the problem of system level fault diagnosis. An algorithm based on GA have been developed by M. Elhadef and B. Ayeb, that work upon completely connected multiprocessor system [11]. In the year 2001,GA-based algorithm have been designed for diagnosing faults in small-scale multiprocessor system, where the test result is interpreted using comparison model [12].

2.4.4 Application of other evolutionary approaches

Apart from genetic algorithm, an artificial immune system based algorithms are available in the literature to solve diagnosis problem for large scale multiprocessor systems [14, 30, 46]. To the best of our knowledge, no GA-based fault diagnosis algorithms exist for K-connected distributed systems. A comparison among different evolutionary system level fault diagnosis algorithms are available in the literature [29]. A perceptron neural network based approach is developed by Elhadef [13, 9], which applies perceptron neural network concept in system level fault diagnosis algorithm.

2.5 Conclusion

Any system which can be decomposed in to several units where units can test each other, the system level fault diagnosis techniques can be applicable. A large number of fault diagnosis algorithms for distributed systems such as multiprocessor, parallel and message passing distributed systems are already available in the literature. There are two types of fault diagnosis algorithms such as centralized diagnosis and distributed diagnosis. The different types of fault and their behaviors, fault detection methods such as test and heart-beat based, fault diagnosis and fault recovery methods for various kinds of system models for distributed systems such as computer networks, wireless adhoc networks and distributed embedded systems have been used for characterizing the fault diagnosis algorithms. Fault diagnosis algorithms based on different evolutionary approaches have also been discussed in this chapter.

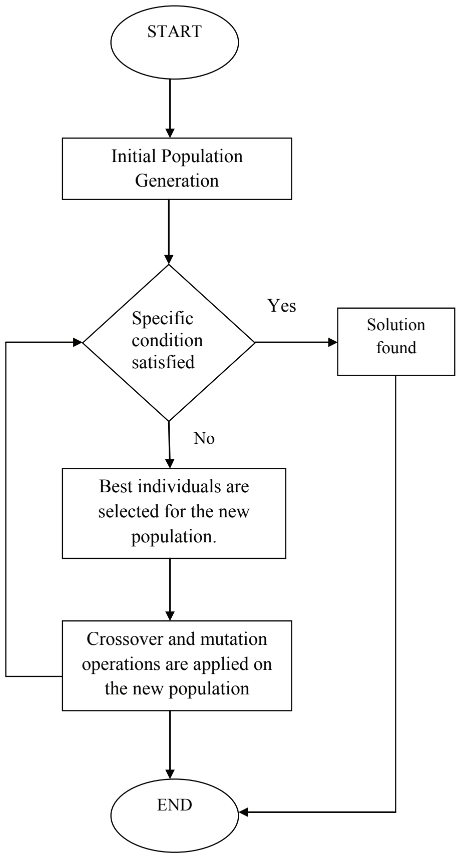

START
Initial Population Generation
Specific condition satisfied
Yes
Solution found
No
Best individuals are selected for the new population.
Crossover and mutation operations are applied on the new population
END

Figure 2.2: Steps followed by Genetic algorithm

Chapter 3

GA-based Permanent Fault Diagnosis in K-connected Distributed System

3.1 Introduction

In the real life applications, the topology of many distributed systems such as world wide web(WWW), Internet, ecological network, cellular networks are random and not completely-connected. These systems are categorized by taking different parameters like connectivity, maximum degree of the network (the degree of the node having maximum number of neighbors), number of links between nodes and diameter of network) [18]. Connectivity of the network in a distributed system is the minimum degree of the network.

The K-connected distributed systems can be modeled as a graph consisting of vertices and edges which correspond to nodes and links of the network in the distributed system respectively. The graph representation helps in analyzing the algorithms for these systems. As the permanent faults are more likely to occur in the nodes of the K-connected distributed systems, in this chapter, we have proposed a fault diagnosis algorithm for diagnosing permanently faulty nodes. The traditional fault diagnosis algorithms are not efficient in terms of ”need of a large number of message exchanges”, recently authors have proposed various genetic algorithm based fault diagnosis algorithms (GAFD) for diagnosing permanently faulty nodes in multiprocessor and other parallel systems which follow a completely connected network topology[11]. As the recent distributed systems are K-connected (or not-completely connected), we have proposed the fault diagnosis algorithm for diagnosing permanently faulty nodes in the K-connected distributed systems in this work. The computational overhead in terms of CPU time and message complexity for the GAFD algorithms are usually less as GA-based fault diagnosis algorithm always provides faster and accurate diagnosis solutions for small and constant size of solution pool.If a graph $G(n,l)$ is designed to represent the nodes and links in a distributed system, where node of system is denoted by n and the link connecting pair of nodes are denoted by l, then the connectivity of a node n_1 is the degree of n_1. In a not-completely connected distributed system, the connectivity of nodes varies with size of the system. A not-completely connected distributed system can be termed as K-connected distributed system, where K is minimum degree of the system.

Mathematically a graph G with vertex set V(G) is defined as K-connected , if the minimum degree of the graph is K. The minimum degree of a task graph is the degree of the node having minimum number of neighbors. The maximum degree of the network is (n-1). The maximum degree of the network is the degree of the node having maximum number of neighbors. The example shown in Figure 3.1 is an example of a K-connected system having connectivity K=2 , where the number of nodes, n=5. The value of K=3, makes the distributed system 3-connected. Similarly when K=(n-1), where n is the number of nodes in the distributed system, the system becomes completely-connected. Figure 3.2 and Figure 3.3 represents the K-connected system having K=3, K=1.

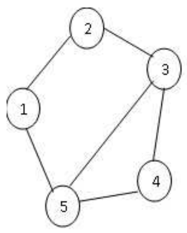


Figure 3.1: 2-connected Distributed System

This chapter is organized as follows. Section 3.1 introduced the various kinds

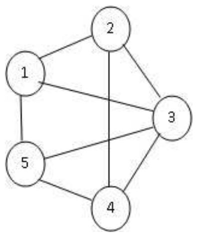


Figure 3.2: 3-connected Distributed System

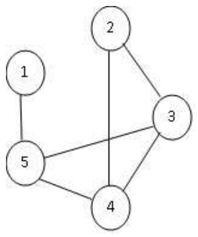


Figure 3.3: 1-connected Distributed System

of distributed systems those usually follow a K-connected topology. This section also discussed about the requirement of permanent fault diagnosis algorithm for K-connected distributed systems using genetic algorithms. In section 3.2, system, fault and diagnosis model for the proposed approach has been specified. The proposed algorithm and its analysis is presented in section 3.3. The simulation results are presented in section 3.4. Finally this chapter is concluded in section 3.5.

3.2 System, Fault and Diagnosis Model

System model of distributed system having connectivity K is represented by a graph $G_K(v,e)$, where nodes of the system are vertices of the graph and links among the nodes are the edges, that connect nodes of the system, In $G_K(v,e)$, v is the set of vertices and e is the set of edges connecting the pair of vertices.

3.2.1 System model

A K-connected distributed system is considered, where intermediate nodes relay messages between some source and destination pairs. A synchronous system is assumed where the communication delay is bounded. This is an implicit assumption, in all prior work on distributed system level diagnosis. A node in the distributed system can communicate directly with another node, provided there is a link exist between these pair of communicating nodes. The k-connected system is a t-diagnosable system that can allow a maximum number of t faults, where t=$\lfloor (n-1)/2 \rfloor$.

3.2.2 Fault and Diagnosis model

Nodes in the distributed system are subjected to permanent faults. Faults assumed to occur at node level rather than chip level. The communication link connecting the nodes of a distributed system is assumed to be faultfree. The link faults are usually handled by the underlying network protocols. A node becomes permanently faulty, if it provides erroneous outcomes, when it runs a task. In fact, the permanently faulty node gives incorrect response which leads to wrong decision at the central node (for example sink node in a wireless sensor network).

In the K-connected distributed system, the node that tests the subset of connected nodes, is known as tester node. The nodes, those get signal from the tester nodes and respond to that, are known as tested nodes. As the nodes are not connected with all their neighbors in the K-connected system, the tester nodes and tested nodes are decided among the connected nodes. In the system, all the nodes have to be tested atleast once, as all the nodes are likely to suffer from permanent faults. In this work, all the nodes get the chance to become tester as well as tested nodes. But no node tests itself. The maximum number of tests performed by a node in the system is K, where K signifies the connectivity of the system.

The model used here for test result interpretation is the asymmetric PMC model [31], where test results generated by faulty node is either 0 or 1. After the tests are performed by applying the PMC model, the test outcome is collected from each node. The collection of test results is a binary string of 0s and 1s. The collection of the testoutcome is known as real test syndrome. The sequence of test outcome in the test syndrome is maintained by the sequence of port number of respective nodes in the K-connected distributed system. The test coverage for all the tests in the K-connected system is assumed to be 100 percent. The system and fault model of one 2-connected system is explained by an example given below.

The figure 3.1 is a 2-connected system. Suppose permanently faulty node are node 3 and node 5 as shown in figure 3.4. Table 3.1 shows the tester and tested nodes of the 2-connected distributed system of figure 3.4. The real test syndrome generated after the testing phase is denoted by TT^*.

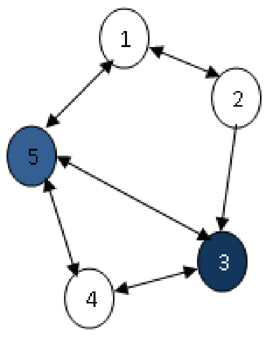


Figure 3.4: Permanently faulty nodes in 2-connected Distributed System

Table 3.1: Tester nodes and tested nodes of 2-connected distributedsystem

Tester node	Tested node
1	2,5
2	1,3
3	4,5
4	3,5
5	1,3,4

Here real test syndrome TT^*, generated after testing the nodes of Figure 3.4 is 10011111111. The test results of individual node in the system can be subdivided in 2 forms, i.e T^* and $T^{*'}$. T^*(node) denotes the testoutcome of the tests performed by the node. $T^{*'}$(node) denotes the testoutcome of the tests performed by the connected neighborhood nodes on that particular node. Table 3.2 shows the individual test outcome(T^* and $T^{*'}$) of every node after the testing phase is over.

Table 3.2: Test outcome obtained individually by each node of 2-connected distributed system

Nodes	T^*	$T^{*\prime}$
1	1,0	1,0
2	0,1	0,1
3	1,1	1,1,1
4	1,1	1,1
5	1,1,1	1,1,1

3.3 GA-Based Permanent Fault Diagnosis Algorithm for K-connected Distributed system(GAPFDA)

The proposed diagnosis approach consists of two algorithms. The algorithm 1 is the main algorithm, in which faulty and faultfree nodes of the K-connected system is found. The algorithm 1 is dependent on algorithm 2. Algorithm 2 returns the fitness of all the chromosomes in every generation to the algorithm 1. Proposed algorithm is similar to the genetic algorithm for permanent fault diagnosis algorithm in completly-connected system given in [11]. The notations, those are used in the algorithm are described below.

G(U,E)= Undirected testgraph represents the processors of the system in terms of vertex u_i and communication link among processors in terms of edge e_i.

TT^*=Binary vector, that contains the information of testoutcomes, after the test is performed on each processor. It is also known as real test syndrome.

TT= Binary vector, that contains the information of testoutcomes from the testgraph, those are generated from chromosomes. It is also known as chromosome generated test syndrome.

F= Set of permanently faulty nodes generated as output from algorithm 1.

FF= Set of faultfree nodes.

F^*= Fault set of real system.

FF^*= Faultfree set of real system.

C= Collection of binary bit strings of length 'n', where n equals to the number of nodes in the system. Number of binary bit strings in C are fixed.

C_{init}= Initial Population(Initial collection of binary bit strings of length 'n', where n equals to the number of nodes in the system.

FT(C_i)= Fitness of chromosome, it is defined as the degree of correctness of an expected solution(C_i) in the solution space.

HighFT= An array that is assigned by the chromosome/expected solution(C_i) of highest fitness value.

FIT= An array that contains the fitness value of all the chromosomes in the population in every generation. It is updated in every new generation.

P= Population size of population.

T^*(node) = Collection of testoutcome, when the tester node tests its connected neighborhood nodes(the nodes to be tested) in the real system.

$(T^*)'$(node)= Collection of testoutcome, when the node is tested by its connected neighborhood nodes in the real system.

T(node)=Set of testoutcome, when the node tests its neighborhood nodes of the *chromosome*-generated testgraph.

T´(node)= Set of testoutcome , when the node is tested by its connected neighborhood nodes of the *chromosome* generated testgraph.

Collect-SameOutcome1(node)= Total number of similar testoutcomes are collected by comparing testoutcome(T^*) of tests performed on real system and testoutcome(T) of the tests performed on chromosome generated testgraph .

Collect-SameOutcome2(node)= Total number of similar testoutcomes are calculated by matching T'(node) and $(T^*)'$(node).

d_{out}(node) and d_{in}(node)= Outdegree and Indegree of a node in the system.

$P_{C1}(node)$ = Probability of correctness of tests performed by tester node(the node,that tests other nodes)

$P_{C2}(node)$ = Probability of correctness of tests performed on a node(tested node).

P_C = The average of $P_{C1}(node)$ and $P_{C2}(node)$

c_i = The i^{th} bit of the bitstring(Chromosome).

C_i = i^{th} chromosome of the population C.

Algorithm 1 GA-Based Permanent Fault Diagnosis Algorithm for K-connected Distributed system

Input:G(U,E) and the real test syndrome TT^*
Output:Faulty(F) and Faultfree(FF) nodes of the system.

```
1: Generate Initial population of chromosomes $C_{init}$.
2: Evaluate fitness of each chromosome of the population $C_{init}$ by calling the subfunction FT($C_{init}$).
3: C=$C_{init}$
4: HighFT←Chromosome of $C_init$ having highest fitness.
5: while ($sum(FIT) \neq P$) do
6:     Selection(C);
7:     Reproduction(C);
8:     Mutation(C);
9:     For all $C_i$ belongs to C,where i varied from 1 to P ,compute the fitness by calling the function FT($C_i$);
10:    HighFT←Chromosome $C_i$ having highest fitness.
11:    Collect the chromosomes of fitness having HighFT value generate new population(C).
12:    FIT←Fitness of all the chromosomes present in the population(C).
13: end while
14: The faulty nodes(Bit value 1) are identified from the chromosomes and stored in the set F. The nodes having bit value 0 are faulfree nodes of the system and stored in the set FF.
15: End
```

Algorithm 2 Algorithm for finding Fitness of each chromosome(FT(chromosome)

```
1: $T^*$(node) and $(T^*)'$(node) are derived from real test syndrome $TT^*$.
2: T($c_i$) and T'($c_i$) are derived from the testgraph $G_c(U,E)$.
3: Collect-SameOutcome1←$|(T^*)(node) \bigcap T(c_i)|$
4: Collect-SameOutcome2←$|(T^*)'(node) \bigcap T'(c_i)|$
5: DOUT←$|d_{out}(node)|$
6: DIN←$|d_{in}(node)|$
7: $P_{C1}(c_i)$←Collect-SameOutcome1/DOUT
8: $P_{C2}(c_i)$←Collect-SameOutcome2/DIN
9: $P_C(c_i)$←$(P_{C1}(c_i)+P_{C2}(c_i))/2$
10: Fitness of Chromosome←$P_C(c_1)+P_C(c_2)+......+P_C(c_n)$)/n ,where $c_1,c_2,c_3,c_4,.....,c_n$ of a chromosome are expected node status(faulty or faultfree) of nodes of real system.
11: return(Fitness of Chromosome)
12: End
```

3.3.1 Description of GAPFDA

In GAPFDA, the input given to the algorithm is the real test syndrome TT^* and the testgraph of the K-connected distributed system. Here the real test syndrome(TT^*) contains the information regarding the testoutcome generated after applying PMC model to the real system. The testgraph represents the connection link of tester node with tested nodes, the weight of each edge in the graph is the outcome of the test performed by tester nodes to the tested nodes. In GAPFDA, the output is the faulty nodes and faultfree nodes of the K-connected system. The basic steps followed by GAPFDA is the most fundamental steps of genetic algorithm. These are

- Selection
- Supervised Mutation
- Reproduction

The following example provides the brief explanation of the genetic algorithm based fault diagnosis algorithm. In the figure 3.5, node 1 is permanently faulty where this 3-connected distributed system consists of 5 nodes numbered in the sequence 1 to 5. Suppose the node 1 is permanently faulty. The generated real test syndrome , TT^* is (111 100 000 1000 100) as per the node sequence 1 to 5. Here all the connection links are utilized as test link. Test graph is shown in Figure 3.5. In order to get the test outcome of individual nodes in the system, we separate TT^* to T^* and $T^{*'}$ as shown in Table 3.3.

Table 3.3: Test outcome obtained individually by each node of 3-connected distributedsystem

Nodes	T^*	$T^{*'}$
1	(1,1,1)	(1,1,1)
2	(0,1,0)	(1,0,0)
3	(0,0,0)	(0,0,0)
4	(0,0,1,0)	(0,1,0,0)
5	(0,0,1)	(0,0,1)

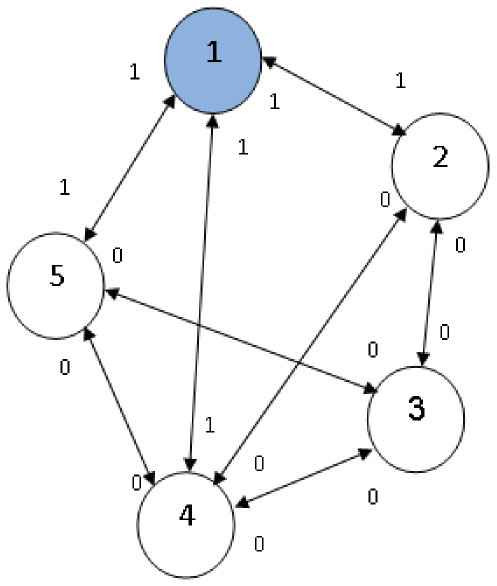


Figure 3.5: Test graph of 3-connected Distributed System

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3.6: Representation of Initial population of chromosomes

The first step of genetic algorithm based permanent fault diagnosis is, generating initial population of chromosomes i.e a set of possible solutions. Here the solutions are defined in the form of binary bit strings. In the above example, the initial population(C_{init}) is represented in figure 3.6. The size of initial population is taken as 4 here. In the initial population, each chromosome is of length n, where n is the number of nodes in the system. In a chromosome, 1 represents a faulty node and 0 represents a faultfree node. So one possible solution(chromosome) contains the information regarding the status of each node in the system(not real). Chromosomes in the population are randomly generated bitstrings. As the system is a t-diagnosable system, each chromosome contains atmost $\lfloor (n-1)/2 \rfloor$ number of 1[11]. Chromosome having all the zero bits is not an expected solution, because we assume atleast fault in the system. So finally chromosomes having all zeros and chromosomes having more than $\lfloor (n-1)/2 \rfloor$ number of 1s are excluded from the population.

The second step of genetic algorithm based permanent fault diagnosis(GAPFD) is, finding the fitness of every chromosome in the population.Here fitness of a chromosome is the average of the fitness of every bit of chromosome. If a test graph is retrived from the chromosome and testing is applied on it, fitness of every bit in the chromosome is the degree of correctness of the test outcome performed by that bit(node). In another way, fitness of chromosome can be defined as how much it is nearer to the actual solution of the diagnosis problem. Fitness of chromosome is denoted by FT, where chromosome is denoted by C and c_i is the i^{th} bit of the chromosome. Here the fitness of every bit of the chromosome of the initial population of the given example is shown in Table 3.4.

Table 3.4: Fitness of all the chromosomes in the initial population

Chromosomes	Fitness of each bit	Fitness of chromosome
10010	(1 ,.66 ,.66 ,.25 ,.66)	.6500
10100	(.83,.66 , 0 ,.75 ,.66)	.5800
11000	(1, .33 ,.66 ,.75 , 1)	.7500
00001	(.33,.66 ,.66 , 1 ,.33)	.6000

Step 3 of GAPFDA is the selection of the chromosome having highest fitness. Here (11000) is the chromosome, that has the highest fitness value i.e .7500.

Step 4 of GAPFDA is reproduction. In the process of reproduction, a new generation is created from old poulation, they have better fitness value than the previous generation. But the population size remains unchanged. In the given example, the chromosomes in the new generation is shown in figure 3.7. Then supervised mutation is applied on the chromosomes of figure 3.7 with probability of mutation p_m. As the value assigned to p_m for a particular chromosome depends on fitness value of each bit in the chromosome, it is termed as supervised mutation. p_m always holds the minimum fitness value from all the fitness values of every bit in the chromosome. According to the given example, for chromosome (11000), the individual fitness of the bits are (1,.33,.66,.75,1). So the value of p_m is taken as .33 .

As fitness of individual bits say about their degree of correctness as a testernode and tested node, the minimum fitness of a bit has the least probability of being correctly diagnosed as well as performing the correct diagnosis on its neighbourhood nodes. Thats why, if the bit having minimum fitness is mutated then there is a probability of increase in degree of correctness for that particular bit. So supervised mutation is applied on the chromosomes and the value of p_m is taken as the minimum fitness value among all the fitness values of bits. Another reason of performing supervised mutation is, by mutating a bit in a chromosome changes the total fitness of chromosome as well as the types of solution nearer to the global optima, increases in the solution space.

In the step 5 of GAPFDA, again the fitness of chromosomes of new population is evaluated. Step 3, Step 4, Step 5 will be repeated until the fitness value of each chromosome becomes 1 in a particular generation. The individual fitness of chromosomes in the new generation as per figure 3.7 is shown in the Table 3.5. The chromosomes generated in Table 3.5 is the optimum solution as they are having fitness as 1.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 3.7: Representation of chromosomes of the new population

By applying mutation to figure 3.7, the 2nd bit of each individual chromosome will be mutated from 1 to 0. According to step 5, the fitness of the chromosomes of the Figure 3.7 after mutation is shown in Table 3.6. The reason "Why the chromosome having fitness value 1 is the optimum solution" is proved in the Lemma 3.1.

Table 3.5: Fitness of all the chromosomes in the initial population

Chromosomes	Fitness of each bit	Fitness of chromosome
11000	(1.0 ,.33 ,.66 ,.75 ,1.0)	.75
11000	(1.0, .33 ,.66 ,.75 ,1.0)	.75
11000	(1.0, .33 ,.66 ,.75 ,1.0)	.75
11000	(1.0 ,.33 ,.66 ,.75 ,1.0)	.75

Table 3.6: Fitness of all the chromosomes after mutation

Chromosomes	Fitness of each bit	Fitness of chromosome
10000	(1.0 ,1.0 ,1.0 ,1.0 ,1.0)	1.0
10000	(1.0, 1.0 ,1.0 ,1.0 ,1.0)	1.0
10000	(1.0, 1.0 ,1.0 ,1.0 ,1.0)	1.0
10000	(1.0 ,1.0 ,1.0 ,1.0 ,1.0)	1.0

3.3.2 Analysis of GAPFDA

In this section, the correctness, efficiency and message overhead of the proposed algorithm GAPFDA are presented. A fault diagnosis algorithm is said to be correct if all the faulty nodes are diagnosed as faulty and all the fault free nodes are diagnosed as fault free. The message overhead of the diagnosis algorithm refers to the message complexity or the total number of messages exchanged during the process of diagnosis. The time complexity of any fault diagnosis algorithm is the total time needed for achieving the diagnosis. Now we prove the correctness, message complexity and the time complexity of the proposed algorithm by using the following lemmas and theorem.

Lemma 3.1:The proposed GA-based fault diagnosis algorithm for diagnosing faulty nodes is correct.

Proof:

To prove the correctness of the proposed algorithm, we consider a t-diagnosable system, where at most t number of permanent fault can exist. The proposed algorithm is implemented on a K-connected system, where K is the connectivity of the distributed network. Let U is assumed as the set of all nodes (identified by their node number) of the K-connected system. Let F^* is the real faulty node set and FF^* is the real fault free node set of the system. The outcome(fault set generated) by the algorithm is F. If F^*=F and FF^*=U-F, then the algorithm is correct.

The proof of correctness is trivial when the set of real faulty nodes are same as that of set of faulty nodes generated during the first generation and also FF^*=U-F. If $F^* \neq F$ and $FF^* \neq U - F$ in the 1st generation, we need to generate another population by selecting the highest fitness valued chromosomes from the previous generation. That leads to $TT^* \neq TT$, where TT^* is the real test syndrome and TT is the chromosome generated test syndrome. Since P_{C1} and P_{C2} of each bit c_i of chromosome C_i depends on TT and TT^*, their values will not be equal to 1. To improve the value of P_{C1} and P_{C2} towards 1, reproduction and supervised mutation are applied on each chromosome of the population C. When we apply reproduction and supervised mutation repeatedly over the intermediate generation, the fitness of each bit of the chromosome will be 1. When P_{C1}=1 and P_{C2}=1, this shows the proposed algorithm achieves the diagnosis within a bounded number of steps by diagnosing each faulty node in the K-connected system as faulty and each fault free nodes in the K-connected system as faultfree. Thus F^*=F and FF^*=U-F.

Lemma 3.2:The message complexity of GAPFDA is O(K*n) and the message complexity for completely connected system is O(n^2).

Proof: The message complexity of GAPFDA, which assumes a K-connected system is less than the GA-based diagnosis algorithm that assumes a completely-connected system. We prove this lemma by computing the message complexity of the algorithm that assumes a completely connected system. The total number of messages exchanged depends on the connectivity K of the K-connected system in the algorithm GAPFDA. Since the total number of nodes are n and each node tests at least K neighborhood nodes, the total number of messages generated for GAPFDA is 2*n*K. We multiply 2 to the term n*K due to two messages exchanged for each test i.e we are sending a test signal and receiving an acknowledgement from the tested node. The value of 2*n*K is reduced to n*K. Thus the message complexity of GAPFDA is O(n*K). If a completely connected system is considered , each node in the system tests (n*(n-1))/2 number of nodes. Each test consists of 2 messages as given above , which leads to O(n^2). This shows that the proposed algorithm reduces the number of messages exchanged than the completely connected system.

Lemma 3.3:Time complexity of GAPFDA is O(K*P*n*ng), where 'K' is the connectivity of the network, 'P' is the population size of the chromosomes and 'ng' is the number of generations, 'n' is the number of nodes.

Proof: Time needed to diagnose the faulty nodes by GAPFDA depends upon (i)time needed to create chromosome generated test syndromes of a population i.e O(K*n*P), where 'P' is the population size(number of chromosomes of the population), 'K' is the connectivity of the network and 'n' is the number of nodes(length of chromosome), (ii)time needed to get the optimum solution by (1)finding fitness of chromosomes, (2)reproduction and supervised mutation, i.e O(ng), ng is bounded for a distributed system of a specific network size due to the application of supervised mutation. Hence the time complexity of GAPFDA is O(K*n*P*ng).

Theorem:The proposed GAPFDA is correct and efficient in terms of message complexity and time complexity.

Proof: The correctness of the algorithm is trivial as all the theorom follows the Lemma 3.1, Lemma 3.2, Lemma 3.3. According to Lemma 3.1, the proposed algorithm GAPFDA is correct and efficient. According to Lemma 3.2, the proposed algorithm is efficient in terms of message complexity. According to Lemma 3.3, the proposed algorithm is efficient in terms of time needed to diagnose the faulty nodes in the K-connected system. Hence the proposed diagnosis algorithm GAPFDA is correct and efficient in terms of message complexity and time complexity.

3.4 Simulation Result

3.4.1 Simulation Model

A graph consisting of n number of nodes having connectivity K is created. The connection matrix is created by the connectivity among nodes(vertices). Each row of connection matrix is a random binary string of length n. Connectivity of each node(vertex) is checked from the connection matrix until the connectivity of the graph is K. If minimum connectivity of the system is found to be 1 or 0 then the connection matrix is rejected and created for the next time.

The designed system model is applicable to K-connected distributed system of varied network sizes starting from 5 nodes upto 500 nodes. These network sizes are suitable for small scale as well as large scale distributed systems. The cost of a link(edge) is directly proportional to the length of the link(edge) in the K-connected graph(system) [37].

3.4.2 Results

The GAPFDA is simulated using MATLAB. The algorithm is evaluated in terms of following parameters.

- Number of nodes in K-connected distributed system verses CPU time.
- Number of steps required to diagnose verses Number of nodes
- Number of message exchanged in completly-connected verses K-connected system.
- Number of steps required to diagnose faulty nodes by applying random mutation and supervised mutation in GAPFDA.
- Comparison of Cpu time needed to diagnose faulty nodes between non-GA based algorithm(FIA) and proposed GA-based algorithm(GAPFDA).

Number of nodes in K-connected distributed system verses CPU time

The total CPU time required by the ultra reliable node to diagnose the K-connected system varies with the number of nodes in the system. As per the graph shown in Figure 3.8, the CPU time taken to execute GAPFDA is directly propertional to the increase in number of nodes of the system. Figure 3.9 represents the total CPU time

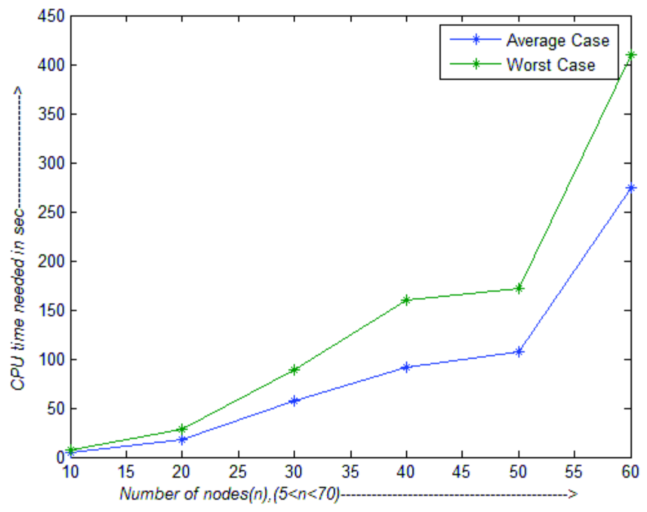


Figure 3.8: Graph showing the change in CPU time with increase in number of nodes where ,($10 <= n <= 60$)

needed to diagnose the faulty nodes in a largescale K-connected system, where number of nodes in the system varies from 70 to 170.

Number of steps required to diagnose verses Number of nodes

In the GA-based fault diagnosis algorithm, the number of steps is defined as the number of generation needed to find out the optimal solution. In the fault diagnosis algorithm, the evaluation of new generation include the steps of supervised mutation and reproduction. So the time required for each step depends upon the probability of mutation of each bit. In the proposed work, the probability of mutation is set to the minimum fitness value(i.e the bits having lowest fitness are to be mutated or flipped). The more the number of bits to be flipped, the more time is needed by the respective step. Here the population size and connectivity K remain fixed and the number of nodes is increased. When the number of nodes varies in the range 10 in the

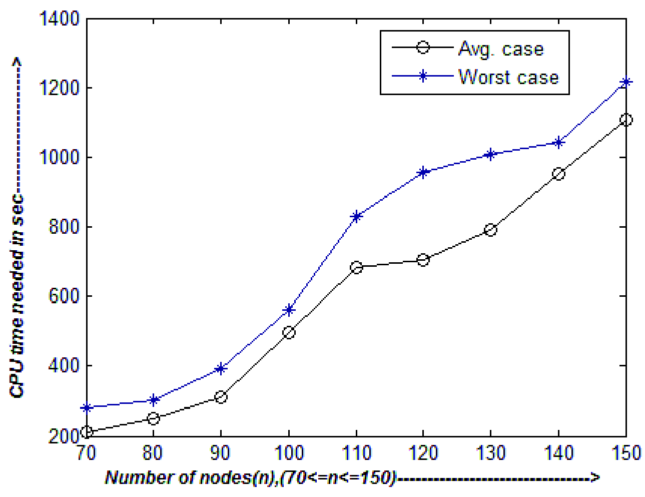
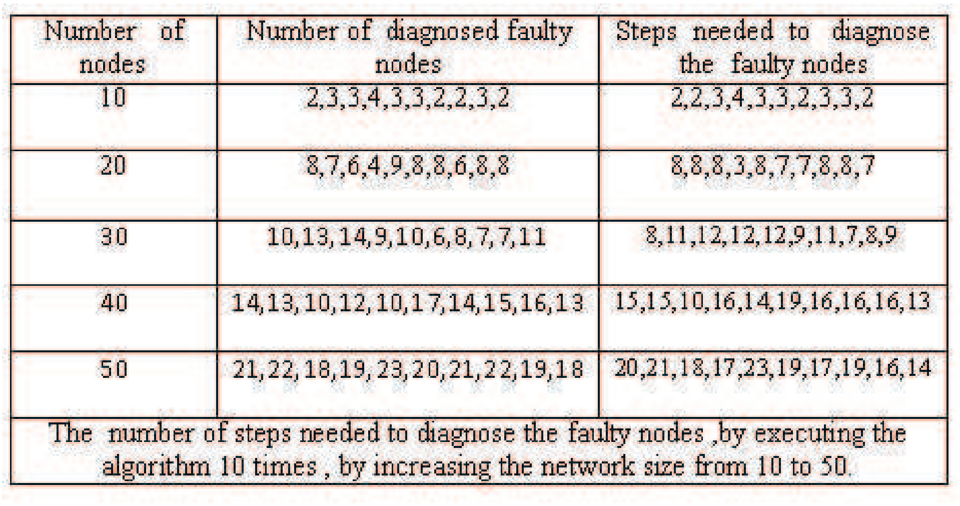


Figure 3.9: Graph showing the change in CPU time with increase in number of nodes where ,$(70 < n < 150)$

system, the number of steps are almost equal. The average case as well as worst case outcome shown in Figure 3.11 concludes that, the number of steps required to find the faulty nodes in the system increase with the size of network. The table 3.10 shows that bounded number of steps are needed due to supervised mutation for diagnosing faulty nodes of a system of constant size. Figure 3.12 shows the outcome of the algorithm when the number of nodes varies upto 500.

Graph showing the number of message transferred in completly connected vs K-connected system.

As K-connected distributed system works by following message passing mechanism, the number of messages transferred in between nodes during testing and diagnosis phase have effect on total time needed to diagnose the faulty nodes. Results show the different case studies regarding different types of systems, such as completly connected system and K-connected systems. In the K-connected system, the message transfer depends upon the connectivity of the node. So the number of messages transferred is lesser as at a time all the neighborhood nodes are not involved in testing a given node. Figure 3.13, Figure 3.14, Figure 3.15, Figure 3.16 shows the outcome when the



Number of nodes	Number of diagnosed faulty nodes	Steps needed to diagnose the faulty nodes
10	2,3,3,4,3,3,2,2,3,2	2,2,3,4,3,3,2,3,3,2
20	8,7,6,4,9,8,8,6,8,8	8,8,8,3,8,7,7,8,8,7
30	10,13,14,9,10,6,8,7,7,11	8,11,12,12,12,9,11,7,8,9
40	14,13,10,12,10,17,14,15,16,13	15,15,10,16,14,19,16,16,16,13
50	21,22,18,19,23,20,21,22,19,18	20,21,18,17,23,19,17,19,16,14
The number of steps needed to diagnose the faulty nodes ,by executing the algorithm 10 times , by increasing the network size from 10 to 50.		

Figure 3.10: The number of steps required with increase in number of nodes

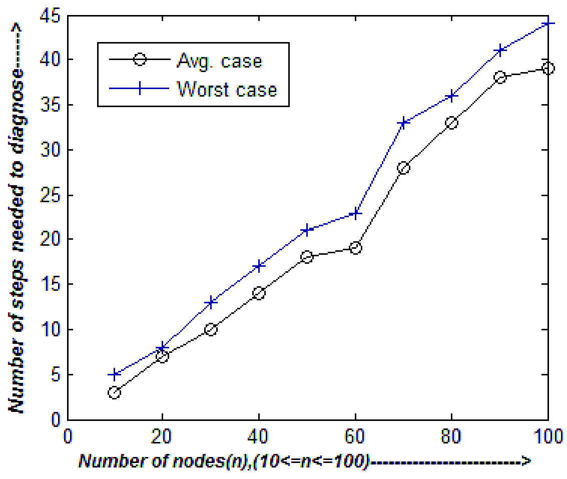


Figure 3.11: Graph showing the number of steps required with increase in number of nodes where ,(10 <= n <= 100)

number of nodes varies upto 100 nodes, 150 nodes and 200 nodes in the K-connected system.

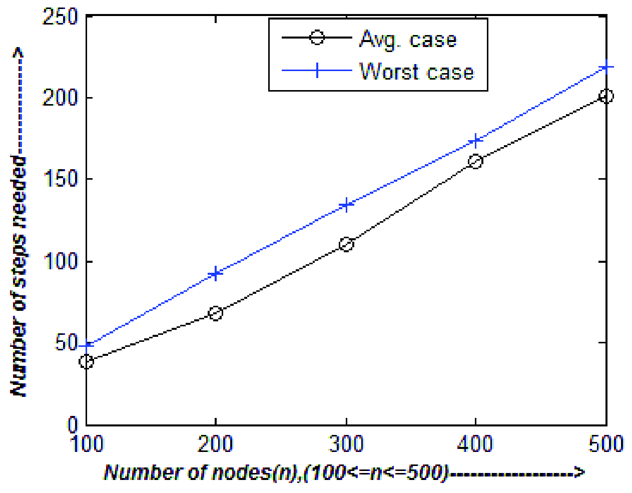


Figure 3.12: Graph showing the number of steps required with increase in number of nodes,where , (100 <= n <= 500)

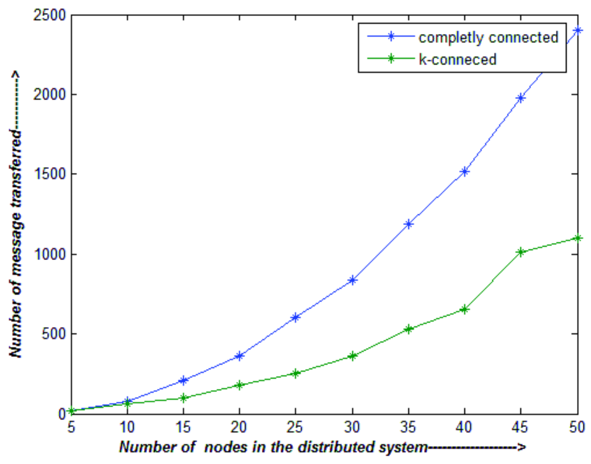


Figure 3.13: Graph showing the number of message transferred in completly connected as well as K-connected system,where , (5 <= n <= 50)

Random Mutation vs Supervised Mutation

Figure 3.17 represents the graph, where different types of mutation technique are used by the algorithm. By analyzing the simulation result, it is concluded that the number of generations(steps) reduces when supervised mutation is applied rather

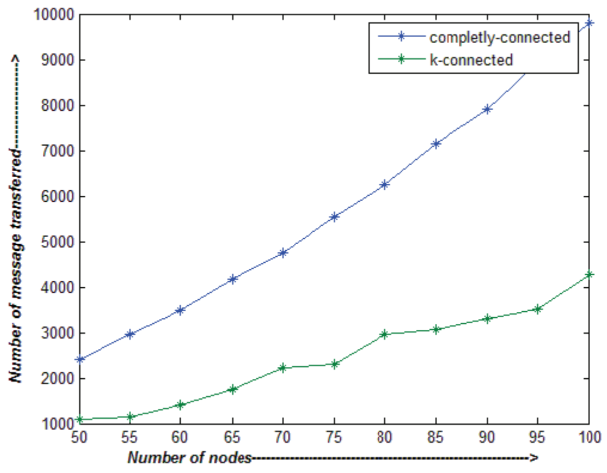


Figure 3.14: Graph showing the number of message transfered in completly connected as well as K-connected system,where , (50 <= n <= 100)

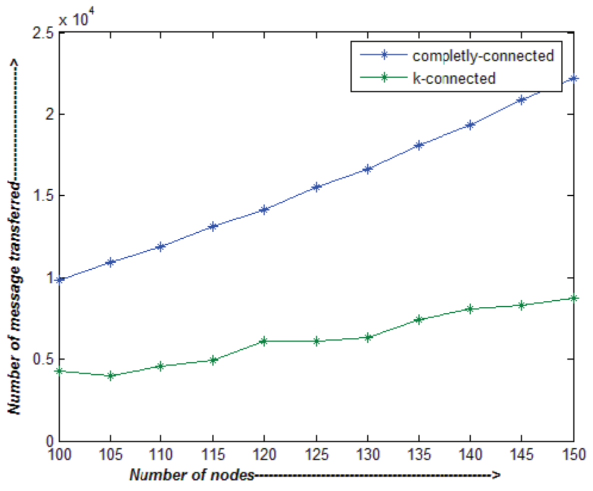


Figure 3.15: Graph showing the number of message transfered in completly-connected as well as K-connected system,where, (100 <= n <= 150)

that random mutation. Because in random mutation , each bit of chromosome have equal probability for improving their fitness. But supervised mutation restricted the solution space by allowing the reduced number of qualitatitive chromosome.

Comparison of Cpu time needed to diagnose faulty nodes between non-GA based algorithm(FIA) and proposed GA-based algorithm(GAPFDA)

Figure 3.18 represents the CPU time needed to diagnose the faulty nodes using GAPFDA is lesser than the CPU time needed to diagnose faulty nodes by using FIA[4].

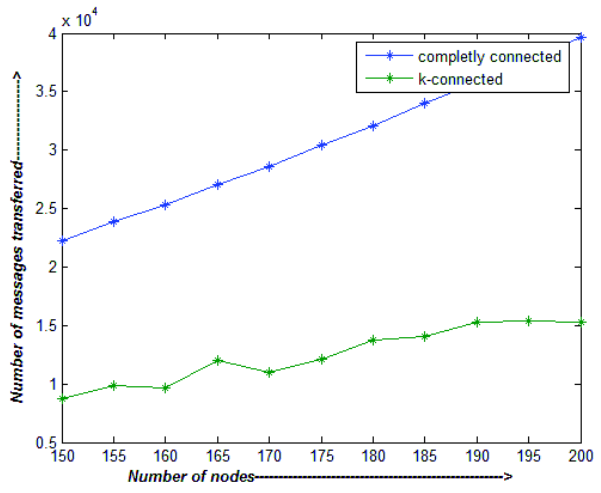


Figure 3.16: Graph showing the number of message transfered in completly-connected as well as K-connected system,where, (150 <= n <= 200)

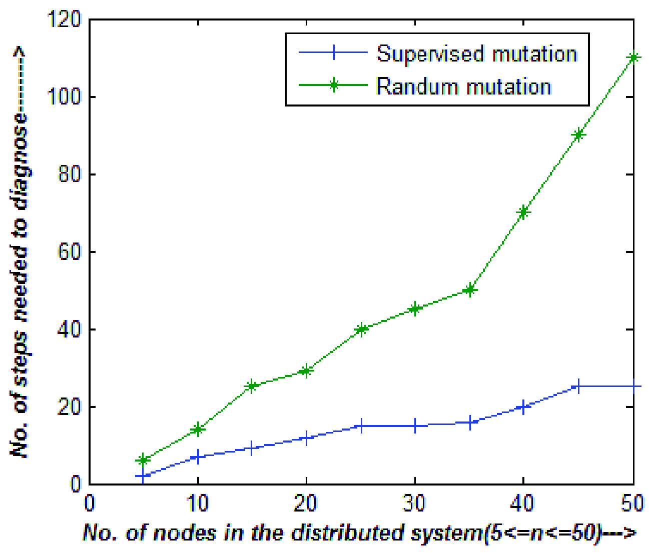


Figure 3.17: Graph showing the number of steps required by applying random mutation and supervised mutation in K-connected system,where, (5 <= n <= 50)

3.5 Conclusion

In this chapter, the requirment of permanent fault diagnosis in K-connected distributed system using genetic algorithms has been introduced. The proposed algorithm along with system, fault and diagnosis model are described in detail. The simulation result

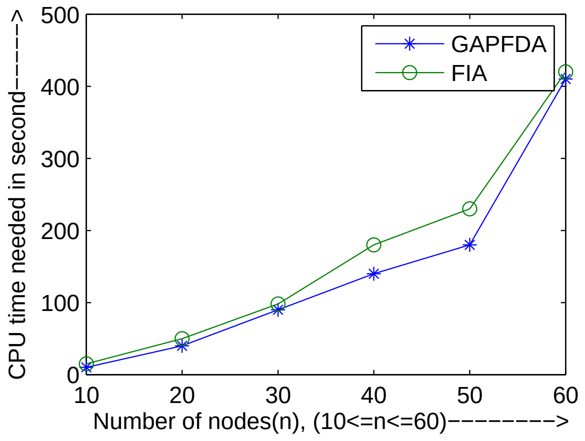


Figure 3.18: Graph showing the where Comparison of CPU time needed to diagnose faulty nodes between FIA and proposed GAPFDA, (10 <= n <= 60)

shows that the proposed algorithm is correct, efficient an feasible in terms of CPU time, number of steps needed for diagnosis and message overhead. An example illustrating the complete functioning of the proposed fault diagnosis algorithm is presented. The proposed diagnosis algorithm, its description and analysis have been discussed. The time complexity and message complexity of GA-based permanent fault diagnosis algorithm(GAPFDA) are O(n*P*K*ng) and O(n*K) respectively.

Chapter 4

GA-based Intermittent Fault Diagnosis In K-connected Distributed System

4.1 Introduction

In chapter 3, we presented a genetic algorithm based fault diagnosis approach to diagnose permanent faulty nodes in a K-connected distributed systems assuming that most of the emerging class of distributed systems are K-connected. It should be noted that many K-connected systems are also subjected to intermittent faults. The presence of intermittent faults in these systems not only can degrade the performance of the system but also reach in erroneous outcome of the tasks which are executed on these systems. This indicates that, these faults affect both quality of service as well as reliability of the systems. An intermittent faulty node in the K-connected distributed system behaves as a faulty node in one duration and behaves as a faultfree node in another consecutive duration. This process is usually repeated for a specific duration. Here the behavior of a node is defined as the response it gives when it receives a signal from the connected neighbourhood nodes when nodes test each other. So intermittently faulty nodes behave in an unpredictable manner. Motivated by the need to maintain the quality of service and reliability of a K-connected distributed system, a GA-based fault diagnosis algorithm to diagnose intermittent faulty nodes is described in this chapter.

The chapter is organised into different sections as follows. Section 4.1 introduces the knowledge regarding the nature of intermittent fault and describes how it is important to diagnose the intermittently faulty nodes in the system. Section 4.2 describes the system, fault and diagnosis model of the K-connected distributed system. Section 4.3 describes the fault diagnosis algorithm. Section 4.4 discusses the results obtained from the simulation followed by conclusion in section 4.5.

4.2 System , Fault and Diagnosis model

The system model here is same as that of system model described in chapter 3. However, we again describe the system model here for completeness of this chapter. The K-connected system is assumed to be synchronous. In a K-connected distributed system, the connectivity of nodes varies with size of the system. The connectivity of K-connected distributed system refers to the minimum degree of the network. The nodes in the application specific K-connected distributed system uses message passing mechanism to perform their work. The K-connected system is t-diagnosable that can allow a maximum number of t faults where t=floor((n-1)/2).

Occurrence of faults may be at different levels of the K-connected distributed system such as physical layer, hardware, system software, and middleware. In this chapter, we considered the fault introduced in the K-connected distributed system is intermittent fault. Faults assumed to occure at node or system level. The communication link connecting the nodes of a distributed system is assumed to be faultfree. A node becomes intermittently faulty, if the faulty nodes behaves faulty for a certain duration and behaves faultfree in the next consecutive duration. If this type of faulty nodes are not detected for a long time, then the nodes become permanently faulty. Hence diagnosing intermittent faulty nodes in the K-connected system are essential for smooth functioning of the system. Diagnosis of intermittent fault reduces the effect of permanent fault on the system because it reduces the chance of a node of leading to permanent fault. In this chapter, the test result is interpreted using PMC model. The table shown below, contains the information about the tester and tested nodes of the 2-connected distributed system shown in Fig 4.1. The test result is interpreted by

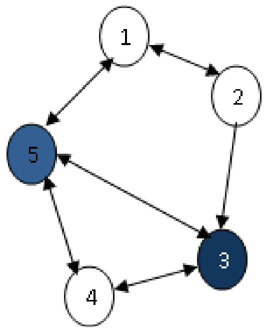


Figure 4.1: Intermittently faulty nodes in 2-connected Distributed System

Table 4.1: Tester nodes and tested nodes of 2-connected distributedsystem

Tester node	Tested node
1	2,5
2	1,3
3	4,5
4	3,5
5	1,3,4

using PMC model .

4.3 GA-based Intermittent Fault Diagnosis Algorithm

The GA-based Intermittent Fault Diagnosis(GAIFD) approach consists of two algorithms. The algorithm 1 is the main algorithm, in which faulty and faultfree nodes of the K-connected system is found. The algorithm 1 is dependent on algorithm 2. Algorithm 2 returns the fitness of all the chromosomes in every generation to the algorithm 1. The notations, those are used in the algorithm are described:
G(U,E)= Undirected testgraph represents the processors of the system in terms of vertex u_i and communication link among processors in terms of edge e_i.
TT^*=Binary vector, that contains the information of testoutcomes after the test is performed on each processor. It is also known as real test syndrome.
TT= Binary vector, that contains the information of testoutcomes from the testgraph, those are generated from chromosomes. It is also known as chromosome generated test syndrome.

F= Set of permanently faulty nodes generated as output from algorithm1.

FF= Set of faultfree nodes.

F^*= Fault set of real system.

FF^*= Faultfree set of real system.

C= Collection of binary bit strings of length 'n' , where n equals to the number of nodes in the system. Number of binary bit strings in C are fixed.

C_{init}= Initial Population(Initial collection of binary bit strings of length 'n', where n equals to the number of nodes in the system.

FT(C_i)= Fitness of chromosome, it is defined as the degree of correctness of an expected solution(C_i) in the solution space.

HighFT= An array that is assigned by the chromosome/expected solution(C_i) of highest fitness value.

FIT= An array that contains the fitness value of all the chromosomes in the population in every generation.

P= Population size at each generation.

T^*(node) = Collection of testoutcome, when the tester node tests its connected neighbourhood nodes(the nodes to be tested) in the real system.

(T*)´(node)= Collection of testoutcome, when the node is tested by its connected neighbourhood nodes in the real system.

T(node)=Set of testoutcome, when the node tests its neighbourhood nodes of the *chromosome*-generated testgraph.

T´(node)= Set of testoutcome , when the node is tested by its connected neighbourhood nodes of the *chromosome* generated testgraph.

Collect-SameOutcome1(node)= Total number of similar testoutcomes are collected by comparing testoutcome(T^*) of tests performed on real system and testoutcome(T) of the tests performed on chromosome generated testgraph.

Collect-SameOutcome2(node)= Total number of similar testoutcomes are calculated by matching T′(node) and $T^{*'}$(node).

d_{out}(node) and d_{in}(node)= Outdegree and Indegree of a node in the system.

$P_{C1}(node)$ = Probability of correctness of tests performed by tester node(the node,that tests other nodes).

$P_{C2}(node)$ = Probability of correctness of tests performed on a node(tested node).

P_C = The average of $P_{C1}(node)$ and $P_{C2}(node)$

c_i = The i^{th} bit of the bitstring(Chromosome)

C_i = i^{th} chromosome.

r= Number of rounds.

R_{fixed}= The number of rounds the testing phase and diagnosis phase will be executed(experimentally set by the user).

RESULT= Matrix to store the diagnosis outcome of each round.

4.3.1 Description of GAIFDA

In GAIFDA, the input given to the algorithm is the real test syndrome TT^* and the testgraph of the K-connected distributed system. Here the real test syndrome(TT^*) contains the information regarding the testoutcome generated after applying PMC model to the real system. The testgraph represents the connection link of comparator node with compared node, the weight of each edge in the graph is the outcome of the test, that have been taken place by tester node and tested node. In GAIFDA, the output is the faulty nodes and faultfree nodes of the K-connected system. The basic steps followed by GAIFDA is the most fundamental steps of genetic algorithm. These are (i)Selection(ii)Reproduction (iii)Supervised mutation.

The following example provides the brief explanation of the genetic algorithm based fault diagnosis algorithm. The 3-connected system shown in Figure 4.2 consists of 5 nodes numbered in the sequence 1 to 5. The node 1 is intermittently faulty. Suppose the test graph is generated for the round r=1 from the figure 4.2 is shown in figure 4.3. In the round r=1,suppose the node 1 is giving incorrect outcome. So the generated real test syndrome , TT^* is (111 100 000 1000 100) as per the node sequence 1 to 5. In order to get the test outcome of individual nodes(T^* and $T^{*'}$) in the system, we separate TT^* to T* and $T^{*'}$ as shown in Table 4.2. The first step of genetic algorithm based intermittent fault diagnosis is generating initial population of chromosomes i.e

Algorithm 3 GA-Based Intermittent Fault Diagnosis Algorithm for K-connected Distributed system

Input:G(U,E) and the real test syndrome TT^*
Output:Faulty(F) and Faultfree(FF) node set of the system.

```
 1: SET r=1 and RESULT=NULL.
 2: while ($r \neq R_{fixed}$) do
 3:     Perform testing at node level , collect the test syndrome.
 4:     Generate Initial population of chromosomes $C_{init}$.
 5:     Evaluate fitness of each chromosome of the population $C_{init}$ by calling the
        subfunction FT(chromosome).
 6:     C=$C_{init}$
 7:     HighFT←Chromosome of C having highest fitness.
 8:     while ($sum(FIT) \neq P$) do
 9:         Selection(C);
10:         Reproduction(C);
11:         Mutation(C);
12:         For all $C_i$ belongs to C,where i varied from 1 to P ,compute the fitness by
            calling FT($C_i$);
13:         HighFT←Chromosome $C_i$ having highest fitness.
14:         Collect the chromosomes of fitness having HighFT value generate new
            population(C).
15:         FIT←Fitness of all the chromosomes present in the population(C).
16:     end while
17:     The faulty nodes(Bit value 1) are identified from the chromosomes and stored
        in the set F. The nodes having bit value 0 are faulfree nodes of the system and
        stored in the set FF.
18:     Set $RESULT[r]$=diagnosed faulty node set F nodes at each round r .
19:     Increment r.
20: end while
21: Identify the faulty nodes from RESULT.
22: End
```

Algorithm 4 Algorithm for finding Fitness of each chromosome(FT(chromosome)

```
1: T*(node) and $T^{*'}$(node) are derived from real test syndrome $TT^{*}$.
2: T($c_i$) and T′($c_i$) are derived from the testgraph $G_c(U, E)$.
3: Collect-SameOutcome1←$|T^{*}(node) \bigcap T(c_i)|$
4: Collect-SameOutcome2←$|T^{*'}(node) \bigcap T'(c_i)|$
5: DOUT←$|d_{out}(node)|$
6: DIN←$|d_{in}(node)|$
7: $P_{C1}(c_i)$←Collect-SameOutcome1/DOUT
8: $P_{C2}(c_i)$←Collect-SameOutcome2/DIN
9: $P_C(c_i)$←$(P_{C1}(c_i)$+$P_{C2}(c_i))$/2
10: Fitness of Chromosome←$P_C(c_1)$+$P_C(c_2)$+......+$P_C(c_n)$)/n ,where $c_1, c_2, c_3, c_4, \ldots, c_n$ of
    a chromosome are expected node status(faulty or faultfree) of nodes of real system.
11: return(Fitness of Chromosome)
12: End
```

Table 4.2: Test outcome obtained individually by each node of 3-connected distributedsystem

Nodes	T*	$T^{*'}$
1	(1,1,1)	(1,1,1)
2	(0,1,0)	(1,0,0)
3	(0,0,0)	(0,0,0)
4	(0,0,1,0)	(0,1,0,0)
5	(0,0,1)	(0,0,1)

a set of possible solutions. Here the solutions are defined in the form of binary bit strings. In the above example, the initial population is represented in figure 4.4. The size of initial population is taken as 4. In the initial population, each chromosome is of length n, where n is the number of nodes in the system. In a chromosome, 1 represents a faulty node and 0 represents a faultfree node. So one possible solution(chromosome) contains the information regarding the status of each node in the system. Chromosomes in the population are randomly generated bitstrings. But each chromosome contains atmost floor((n-1)/2) number of 1. Chromosome having all the zero bits is not an expected solution, because we assume the presence of at least one fault in the system. So finally chromosomes having all zeros and chromosomes having more than floor((n-1)/2) number of 1s are excluded from the population.

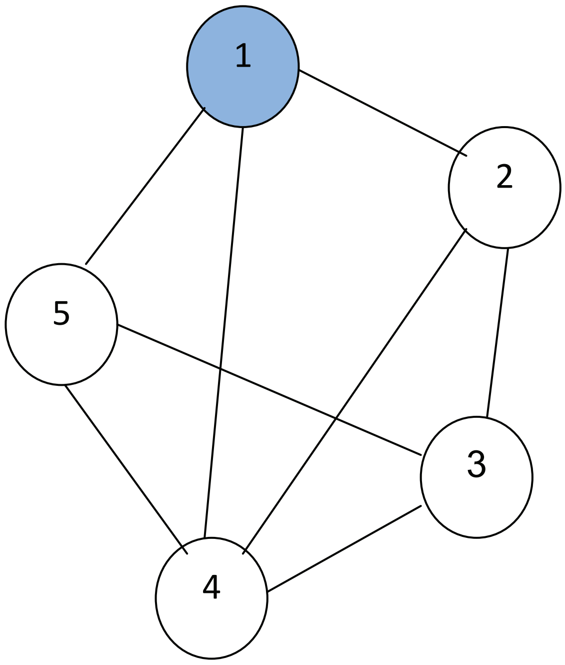


Figure 4.2: A 3-connected Distributed System

The second step of genetic algorithm based intermittent fault diagnosis(GAIFD) is, finding the fitness of every chromosome in the population. Here fitness of a chromosome is the average of the fitness of every bit of chromosome. If a test graph is retrived from the chromosome and testing is applied on it, fitness of every bit in the chromosome is the degree of correctness of the test performed by that bit(node). In another way, fitness of chromosome can be defined as how much it is nearer to the actual solution of the diagnosis problem. Fitness of chromosome is denoted by FT, where chromosome is denoted by C and c_i is the i^{th} bit of the chromosome.

Table 4.3: Fitness of all the chromosomes in the initial population

Chromosomes	Fitness of each bit	Fitness of chromosome
10010	(1 ,.66 ,.66 ,.25 ,.66)	.6500
10100	(.83,.66 , 0 ,.75 ,.66)	.5800
11000	(1, .33 ,.66 ,.75 , 1)	.7500
00001	(.33,.66 ,.66 , 1 ,.33)	.6000

Step 3 of GAIFDA is the selection of the chromosome having highest fitness. Here (11000) is the chromosome, that has the highest fitness value i.e .7500.

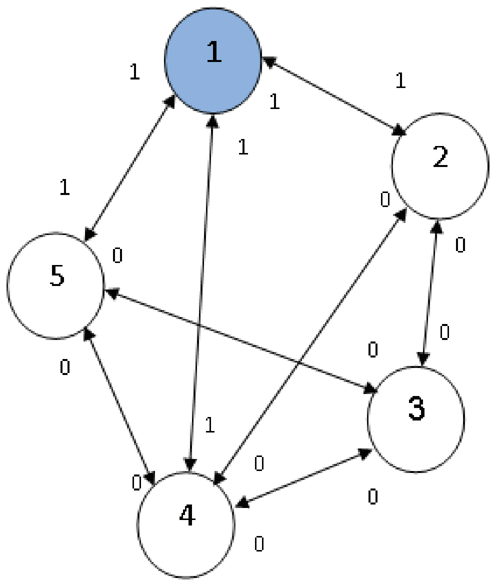


Figure 4.3: Test graph of 3-connected Distributed System

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure 4.4: Representation of Initial population of chromosomes

Step 4 of GAIFDA is reproduction. In the process of reproduction, a new generation is created from old poulation, they have better fitness value than the previous generation. But the population size remains unchanged. In the given example, the chromosomes in the new generation is shown in figure 4.5. Then supervised mutation is applied on chromosomes with probability of mutation p_m. p_m always holds the minimum fitness value from all the fitness values of every bit in the chromosome. According to the given example, for chromosome (11000), the individual fitness of the bits are $(1, .33, .66, .75, 1)$. So the value of p_m is taken as .33 .

As fitness of individual bits say about their degree of correctness as a testernode and tested node, the minimum fitness has the least probability of being correctly diagnosed as well as performing the correct diagnosis. Thats why if the bit having min-

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Figure 4.5: Representation of chromosomes of the new population

imum fitness is mutated then there is a probability of increase in degree of correctness for that particular bit. Thats why supervised mutation is applied on the chromosomes and the value of p_m is taken as the minimum fitness value among all the fitness values of bits. Another reason of performing supervised mutation is by mutating a bit in a chromosome changes the total fitness of chromosome as well as the types of solution increases in the solution space.

In the step 5 of GAIFDA, again the fitness of chromosomes of new population is evaluated. Step 3, Step 4, Step 5 will be repeated until the fitness value of each chromosome becomes 1 in a particular generation. The individual fitness of chromosomes in the new poulation is shown in the Table4.4. In step 4, by applying probability

Table 4.4: Fitness of all the chromosomes in the initial population

Chromosomes	Fitness of each bit	Fitness of chromosome
11000	(1.0 ,.33 ,.66 ,.75 ,1.0)	.75
11000	(1.0, .33 ,.66 ,.75 ,1.0)	.75
11000	(1.0, .33 ,.66 ,.75 ,1.0)	.75
11000	(1.0 ,.33 ,.66 ,.75 ,1.0)	.75

of mutation to the new poulation, the 2^{nd} bit of each individual chromosome will be mutated from 1 to 0. According to step 5, the fitness of the chromosomes of the new population is again calculated and shown in Table 4.5. The GAIFDA algorithm will be executed for r rounds, in each round it will get fault set. Because at every round the intermittently faulty nodes will give correct outcome also. In the given example if the value of r is set to 1 and 5 rounds of execution will be performed, in the first

Table 4.5: Fitness of all the chromosomes after mutation

Chromosomes	Fitness of each bit	Fitness of chromosome
10000	(1.0 ,1.0 ,1.0 ,1.0 ,1.0)	1.0
10000	(1.0, 1.0 ,1.0 ,1.0 ,1.0)	1.0
10000	(1.0, 1.0 ,1.0 ,1.0 ,1.0)	1.0
10000	(1.0 ,1.0 ,1.0 ,1.0 ,1.0)	1.0

round the outcome of the algorithm may be ″node 1 is faulty″. In the second round of testing , the outcome may be ″node 1 is faultfree″ i.e no node is faulty, In the 3rd round the outcome may be ″node 1 is faulty″, followed by the outcome ″node 1 is faulty″ in the final round. The number of times the system will be checked, that improves the confidence of getting the complete and correct fault set of the system. During each round of execution the diagnosis algorithm will get different test syndrome depending upon the number of identified intermittently faulty nodes in the system.

4.3.2 Analysis of GAIFDA

In this section, the correctness, efficiency and message overhead of the proposed algorithm GAIFDA are presented. A fault diagnosis algorithm is said to be correct if all the intermittently faulty nodes are diagnosed as intermittently faulty and all the fault free nodes are diagnosed as fault free. The message overhead of the diagnosis algorithm refers to the message complexity or the total number of messages exchanged during the process of diagnosis . The time complexity of any fault diagnosis algorithm is the total time needed for achieving the diagnosis. Now the correctness, message complexity and the time complexity of the proposed algorithm is proved by using the following lemmas and theorems.

Lemma 4.1: The proposed GA-Based intermittent fault diagnosis algorithm is correct.

Proof:

To prove the correctness of the proposed algorithm, we consider a t-diagnosable system, where at most t number of intermittent faults can be diagnosed. The proposed algorithm is implemented on a K-connected system, where K is the connectivity of the distributed network. Let U is assumed as the set of all nodes (identified by their node number) of the K-connected system. Let F^* is the real faulty node set and FF^* is the real fault free node set of the system. The outcome(fault set generated) by the algorithm is F. If F^*=F and FF^*=U-F, then the algorithm is correct.

If $F^* \neq F$ and $FF^* \neq U - F$ in the 1st generation of all the rounds, we need to generate another population by selecting the highest fitness valued chromosomes from the previous generation. That leads to $TT^* \neq TT$, where TT^* is the real test syndrome and TT is the chromosome generated test syndrome. Since P_{C1} and P_{C2} of each bit c_i of chromosome C_i depends TT and TT^*, their values will not be equal to 1. To improve the value of P_{C1} and P_{C2} towards 1, reproduction and supervised mutation are applied. When we apply reproduction and supervised mutation repeatedly over the intermediate generation, the fitness of each bit of the chromosome is 1. When P_{C1}=1 and P_{C2}=1, this shows the proposed algorithm achieves the diagnosis within a bounded number of steps by diagnosing each intermittently faulty node in the K-connected system as faulty and each fault free nodes in the K-connected system as faultfree.Thus F^*=F and FF^*=U-F for each round.

However an intermittent fault can be captured when the status of the node is in F and considered to be faulty in a particular round and the status of the same node is in FF i.e the node is faultfree in some other round. This behavior of the intermittently faulty nodes depend on the comparison outcome between real test syndrome and chromosome generated test syndrome. Though F^*=F and FF^*=U-F due to the assumption of 100 percent test coverage, for all the rounds remains true, the number of faulty nodes and faultfree nodes along with their identification numbers will change from one round to another round in the occurence of intermittent faults. This shows that the intermittent faulty nodes are captured correctly within bounded time.

Lemma 4.2:The message complexity of GAIFDA is O(r*K*n), where ′r′ is total number of rounds, ′K′ is the connectivity of the system and ′n′ is the number of nodes in the system.
Proof: The message complexity of GAIFDA, which assumes a K-connected system depends upon the number of rounds, the algorithm gets executed to confirm the faulty nodes are truely intermittently faulty. We prove this lemma by computing the message complexity of the algorithm. The total number of messages exchanged depends on the connectivity K of the K-connected system in the algorithm GAIFDA. Since the total number of nodes are n and each node tests at least K neighborhood nodes, the total number of messages generated for GAIFDA is 2*n*K. We multiply 2 to the term n*K due to two messages exchanged for each test i.e we are sending a test signal and receiving an acknowledgement from the tested node. The value of 2*n*K is reduced to n*K. The number of rounds, the diagnosis phase will continue, is ′r′. Hence the message complexity of GAIFDA is O(r*n*K).

Lemma 4.3: Time complexity of GAIFDA is O(r*K*n*P*ngi), where ′r′ is number of rounds and ′K′ is the connectivity of the network, ′n′ is the total number of nodes, ′P′ is the population size of the GA, ′ngi′ is the number of generations of the GA.
Proof:
Time needed to diagnose the faulty nodes by GAIFDA depends upon (i)time needed to create chromosome generated test syndrome from the chromosomes of the population i.e O(K*n*P), where P is the number of chromosomes of the population, K is the connectivity of the network and n is the number of nodes(length of chromosome), (ii)time needed to get the optimum solution by (1)finding fitness of chromosomes, (2)reproduction and supervised mutation, i.e O(ngi),ngi is bounded for a distributed system of a specific network size due to the application of supervised mutation. The number of rounds the algorithm gets executed is r. Hence the time complexity of GAIFDA is O(r*K*n*P*ngi).

Theorem:The proposed GAIFDA is correct and efficient in terms of message complexity

and time complexity.
Proof: The correctness of the algorithm is trivial as all the theorem follows the Lemma 4.1, Lemma 4.2, Lemma 4.3. According to Lemma 4.1, the proposed algorithm GAIFDA is correct and efficient. According to Lemma 4.2, the proposed algorithm is efficient in terms of message complexity. According to Lemma 4.3, the proposed algorithm is efficient in terms of time needed to diagnose the faulty nodes in the K-connected system. The proposed diagnosis algorithm GAIFDA is correct and efficient in terms of message complexity and time complexity.

4.4 Simulation Results

The GAIFDA is simulated using MATLAB. The algorithm is evaluated in terms of following parameters.

- Number of nodes in K-connected distributed system verses CPU time.
- Number of steps required to diagnose verses Number of nodes.
- Number of messages transferred in GAPFDA and GAIFDA verses Number of nodes.

Number of nodes verses CPU time needed to diagnose

The total CPU time required by the ultra reliable node to diagnose the K-connected system varies with the number of nodes in the system. As per the graph shown in Figure 4.6 and figure 4.7, the number of rounds the algorithm will be executed is taken as 5. With the increase in number of nodes from 10 to 140, the CPU time needed to diagnose the intermittent faulty nodes increases. But along with the number of nodes, the second factor on which the CPU time depends, is the number of rounds the algorithm gets executed to get the complete set of intermittently faulty nodes. If we vary the number of rounds from 5 to 10 or 15 or 20 accordingly the diagnosis time increases. The number of rounds is set to 5 in all the experiments.

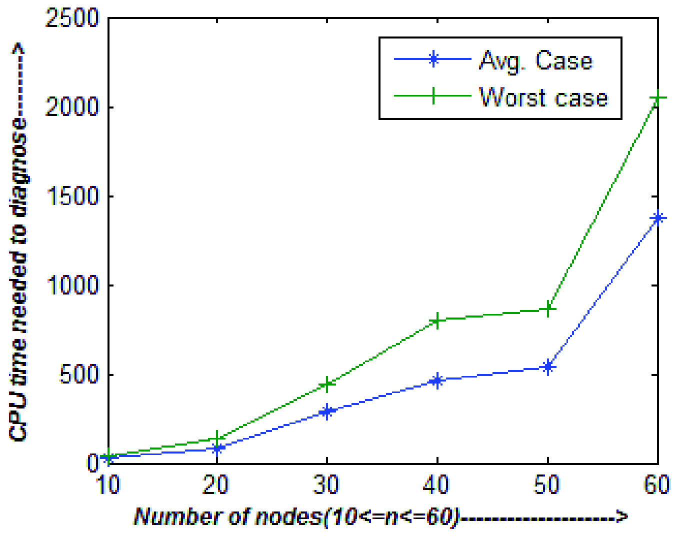


Figure 4.6: Graph showing the change in CPU time with increase in number of nodes where ,($10 <= n <= 60$)

Number of steps required to diagnose verses Number of nodes

The number of steps is defined as the number of generation needed to find out the optimal solution. In the fault diagnosis algorithm, the evaluation of new generation include the steps of supervised mutation and reproduction. So the time required for each step depends upon the number of mutated bits. In the proposed work, the probability of mutation is set to the minimum fitness value(i.e the bits having lowest fitness are to be mutated or flipped). The more the number of bits to be flipped , the more time is needed by the respective step.Here the population size and connectivity K remain fixed and the number of nodes is increased.When the number of nodes varies in the range 10 in the system, the number of steps are almost equal. The average case as well as worst case outcome shown in Figure 4.9 concludes that, the number of steps required to find the faulty nodes in the system increase with the size of network, where the number of rounds the GAIFDA gets executed is constant.

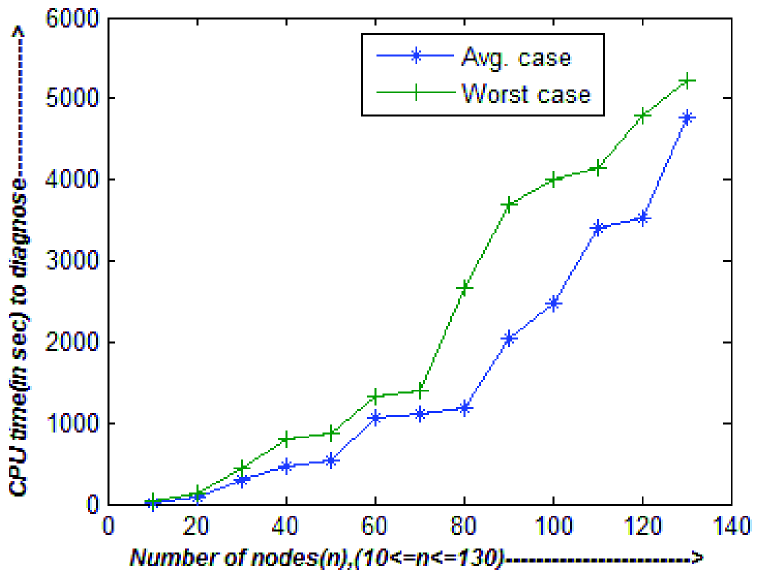


Figure 4.7: Graph showing the change in CPU time with increase in number of nodes where,($10 <= n <= 130$)

Number of messages transferred in GAPFDA and GAIFDA with increase in number of nodes

In figure 4.9, the network size is increased upto 100 nodes and the total number of message transferred is calculated. The number of messages transferred in between the nodes of a K-connected distributed system for diagnosing intermittent fault is more than the message transferred in diagnosing permanent fault of a system. In intermittent fault diagnosis, the number of messages transferred depends upon the number of rounds the algorithm gets executed. In the simulation model the number of rounds is taken as 5. The total number of messages transferred is 5*(the number of test messages transferred in 1 round).

The proposed GAIFDA is the extension of the work in chapter 3,hence the comparison has been only made with the GAPFDA of chapter 3.

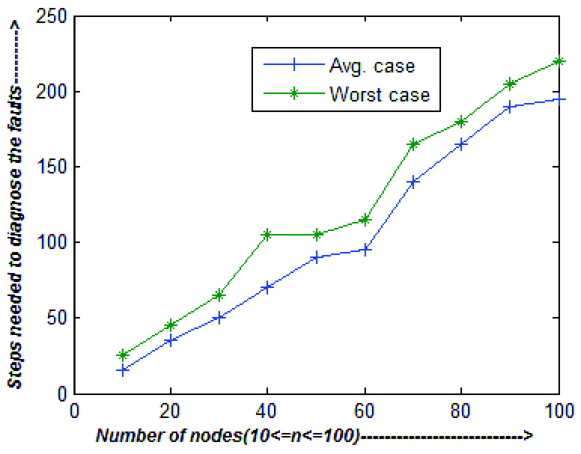


Figure 4.8: Graph showing the number of steps needed to diagnose intermittently faulty nodes with increase in number of nodes where,(10 <= n <= 100)

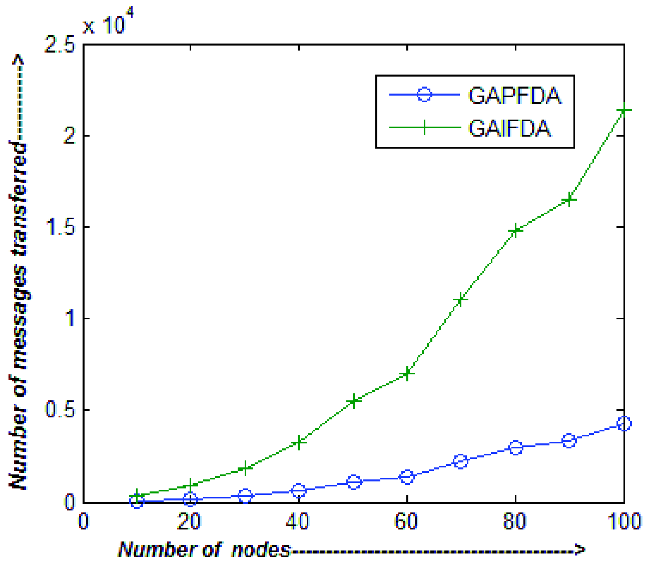


Figure 4.9: Graph showing the number of messages transferred in GAPFDA and GAIFDA with increase in number of nodes where,(10 <= n <= 100)

4.5 Conclusion

In this chapter, the requirment of intermittent fault diagnosis in K-connected distributed system using GA has been introduced. The proposed algorithm along with system, fault and diagnosis model are described in detail. The simulation result is shown for different network sizes. The result shows that, as we increase the networksize, CPU time needed to diagnose nodes, no. of steps required to diagnose the nodes increases. The proposed diagnosis algorithm, its description and analysis have been discussed by suitable example. The time complexity and message complexity of GA-based intermittent fault diagnosis algorithm(GAIFDA) are O(r*n*P*K*ngi) and O(r*n*K) respectively.

Chapter 5

Fault Diagnosis in Multi Rate Fly-By-Wire system

5.1 Introduction

Distributed Embedded Systems are being increasingly used in various application domains like automotives, weather forecasting and defence field. This is due to reduction in processing element cost and development in network technology [7]. Automotives like Fly-By-Wire system, Steer-By-Wire system, Break-By-Wire system are distributed embedded system comprises of smart sensors, actuators, microcontrollers connected by a bus having limited bandwidth. In fly-by-wire system, the traditional aircraft system is replaced by a microprocessor-controlled and networked electromechanical one without any mechanical backup. Wires from the control stick in the cockpit to the control surfaces on the wings and tail surfaces replaced the entire mechanical flight control system.

The key advantage of Fly-by-wire system over fly-by-hydraulic lines is given below [39].

- Less vulnerability to battle damage
- Maneuverability
- Smoothness
- Lighter

- Requires less maintenance
- Reduces fuel consumption

For electric steering control sensors at hand-wheel and road-wheel position sense the hand wheel angle and road wheel force respectively. Embedded microcontrollers compute the desired actuation command for the actuators in steering rack motor. Due to electromechanical failure, an actuator delivers erroneous output. This may result in undesirable system level behavior. Faults in actuators in the Fly-By-Wire system, may occur repeatedly due to electromechanical failure, due to undesired input from the microcontrollers. After certain interval, the fault becomes permanent and causes catastrophic failure if it is not diagnosed beforehand. Diagnosing the fault at system level and making the system reliable inspite of occurrence of fault is essential for the safety-critical fly-by-wire systems.

The System level fault diagnosis overcomes the examination of complex circuitry at chip level. It concerns with the behavior of faulty nodes in a Fly-by-wire system. Another way of making the system more efficient depends upon efficient communication among sensors, actuators and microcontrollers. The lesser the message transmission among the nodes, lesser will be the communication overhead, lesser will be the communication cost. The transmission protocols used for these type of real time distributed systems are TTP [25], FlexRay [15], CAN. The author of [24] has proposed a software-based approach, where diagnosis of failed components is done within designer-specified deadline while meeting the performance goals of a single-rate control application. The factors that motivated us to do this work are (i) the Fly-by-wire systems are safety-critical. The diagnosis is indispensable for such systems. (ii) the comparison based diagnosis [32] is the most practical diagnosis methods and is suitable for these systems. (iii)as in case of self validating actuators, diagnostic checks can be built into a processor, the computing power needed to execute sophisticated behavioural models is generally not available, external diagnosis complements the local check done by self diagnosis of the processors. (iv) Model based redundancy is used for fault diagnosis, where software based diagnosis tasks compare the difference in-between actual component behaviour with the one predicted by a mathematical model and reconfiguration takes place before the system becomes unsafe. The existing approach is applicable in multi-rate system, where the command to actuators is given at a varied time interval.

5.2 System , Fault and Diagnosis Model of Fly-By-Wire System

Fig. 5.1 shows the distributed architecture of Fly-By-Wire system, in which smart sensors, actuators, micro-controllers are connected through controller area network. Micro-controllers, actuators, sensors in the network suffer from different types of failures, which are assumed to be bounded in numbers. Fig. 5.2 shows the integrated

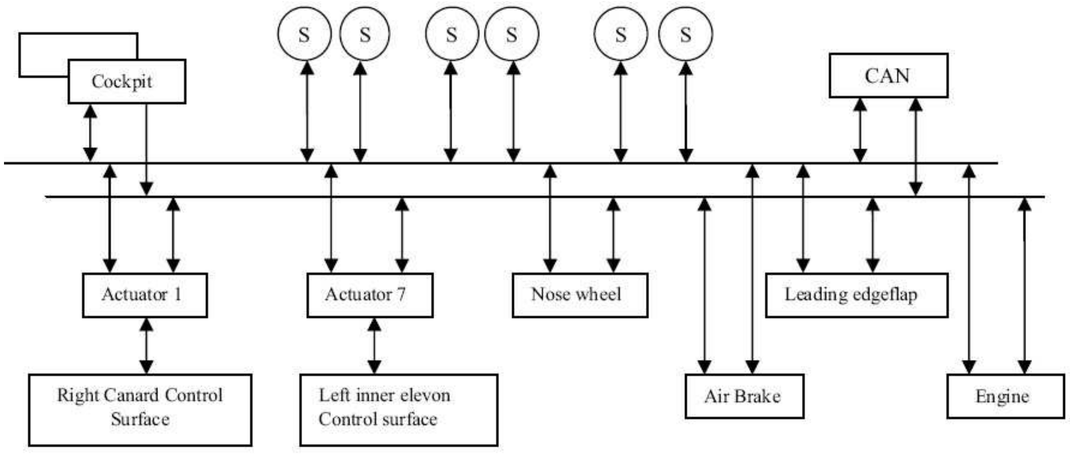


Figure 5.1: Distributed architecture of fly-by-wire system

approach to fault tolerant control and distributed diagnosis for FBW system. Actuator control portion defines all the control tasks providing command to the actuators. In a multi-rate system like fly-by-wire system, 2k+1 modular redundancy technique is followed to control the actuator and to make the system k-fault-tolerant. Actuator diagnosis portion of the system model defines all the tasks those are used to diagnose the faulty actuator. System model signifying the control and diagnosis portion is assumed to perform the desired function in safety critical environment.

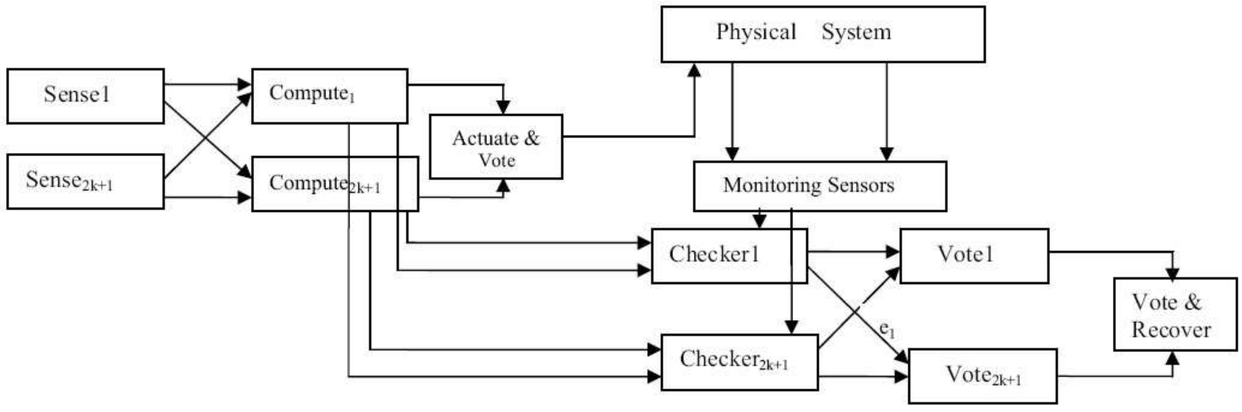


Figure 5.2: Integrated approach to fault tolerant control and distributed actuator diagnosis(adopted from [21])

5.2.1 System Model

Control architecture comprises of redundant sensors, actuators, micro-controllers of different types connected by controller area network via a network interface, which guarantees deterministic transmission latency. Message transmission between the components of the control architecture follows CAN protocol, which assumes a fault - free communication. Smart sensors of the control architecture sense the environmental parameters, the micro-controllers connected in the network compute the desired actuator command and the voted actuation command is given to the actuator. All the sense, compute, actuate tasks of Fig. 5.1 are mapped to the Fig. 5.3 .

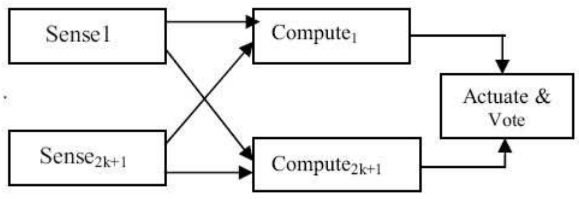


Figure 5.3: Control Architecture

5.2.2 Fault and Diagnosis Model

Actuator diagnosis architecture consists of two-phase approach. In the first phase each microcontroller independently evaluates the behavior of actuator A_i. The local decisions are then consolidated via a suitable agreement algorithm during phase 2 to obtain a global view of A_i's status. Fig. 5.4 describes the actuator diagnosis architecture. By following the rule of analytical redundancy , dedicated microcontrollers of the k-FT system checks the actuator's status individually within their local execution time. After evaluation of actuator status,the status of the actuator is determined by the help of a mathematical model running within individual microcontrollers. Here status is nothing but the difference in between the actual actuator command with the estimated command derived from the mathematical model locally, the input to the mathematical model is the information obtained from monitoring the environment, after the recent actuation command is given to the actuator.

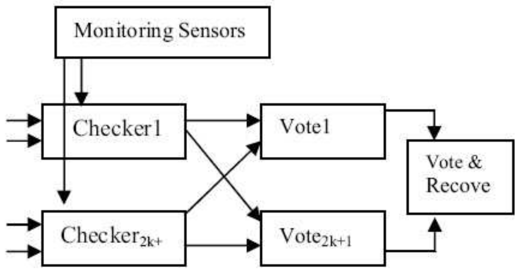


Figure 5.4: Diagnosis Architecture

5.3 Fault Diagnosis Algorithm

The notations used in the proposed algorithm is as follows.
r= Number of rounds.
E= Set of residues, where residue=(Estimated actuation command)-(Calculated actuation command).

5.3.1 Description of the Fault Diagnosis Algorithm

The fault diagnosis algorithm is described as follows. The behavior of the actuators in the FBW system is studied by monitoring sensors. Then the microcontrollers study the difference in between the actual actuator command value and the value obtained

Algorithm 5 Fault Diagnosis Algorithm

```
 1: Number of rounds $r = 1$
 2: while ($r < n$) do
 3:     Monitoring sensors read the actuator behavior.
 4:     while ($|E| \neq 1$) do
 5:         Microcontrollers performing checker tasks gets the input from the moni-
            toring sensors and compute the actuation command.
 6:         The estimated value of actuation command is determined by the checker
            tasks.
 7:         Residues in each microcontroller is obtained by comparing the estimated
            actuation command with previously applied actuation command.
 8:         $E = (e_1, e_2, e_3, ...e_{2k+1})$,where $e_1$,$e_2$,...,$e_{2k+1}$ set of residues obtained from the
            checker tasks to each processors.
 9:         Remove that pair from the set E.
10:     end while
11:     The last element remained in the set is the obtained.
12:     if ($obtainedvalue >= thresholdvalue$) then
13:         Faulty actuator is diagnosed as transient faulty
14:     end if
15: end while
16: Actuator is diagnosed as permanently faulty and recovery task is performed on
    actuator
17: End
```

from the manipulation of the outcome, which is obtained from the monitoring sensors. Status obtained in term of residues from redundant microcontrollers performing checking tasks again fed as input to microcontrollers set having cardinality $2k+1$, Each microcontroller in the set, applied certain voting mechanism [41] to the obtained set of input and evaluates them against a priori defined threshold to diagnose actuator. Step 4 to step 8 in the algorithm obtain the median value of obtained residues in different microcontrollers. In step 9 , the median value obtained from the set of residues is compared with threshold value, which is system specific. Once the obtained value is greater than equal to threshold value, the actuator is detected as faulty for the first time, this fault is known as transient fault. If the transient fault remains in the actuator for a certain period of time, that becomes permanently faulty. If the same residue is obtained for r number of times, then transiently faulty actuator is diagnosed as permanently faulty and recovery action is taken place.

5.3.2 Analysis of Algorithm

Monitoring sensors reads the actuator behavior in O(1) time. Step 2 to 7 takes O(2k+1) time that is O(k). Step 1 to 10 in the algorithm takes O(r) times. So the time complexity of the proposed algorithm is O(r*k) for r rounds in a k-fault tolerant system like FBW.

5.4 Graph Model of FBW System

The proposed system model is mapped to the directed acyclic task graphs G_1, G_2, G_3. As Fly-By-Wire system is a multi-rate system, control portion of G_1, G_2, G_3 are different from one another. But diagnosis portion of the graphs G_1,G_2,G_3 are taken similar to the directed sub-graph of the steer-by-wire system proposed by author Nagrajan Kandaswamy in [24]. The task graph representing the diagnosis portion of Fly-By-Wire system is shown in Fig. 5.8 . Task-graph G_i consisting of vertices and edges representing tasks and precedence constraints respectively. The weight associated with the edges in between tasks is represented by intercommunication cost. Sometimes the weight associated with the edges represent sampling delay. Here sampling delay is the time interval after which monitoring sensors of the diagnosis portion sense the physical system.

In Fly-By-Wire system, different functionalities like electric steering control, traction control, cruise control generate command for the actuators repeatedly at different time intervals, task graph model differs according to the different functionalities during takeoff, cruise control, landing of flight. The task-graphs representing steering control, traction control and cruise control are shown in Fig. 5.5, Fig. 5.6, Fig. 5.7 respectively.

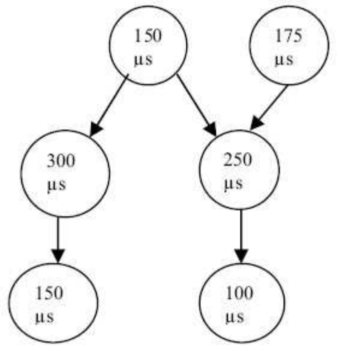


Figure 5.5: Electric steering control task graph

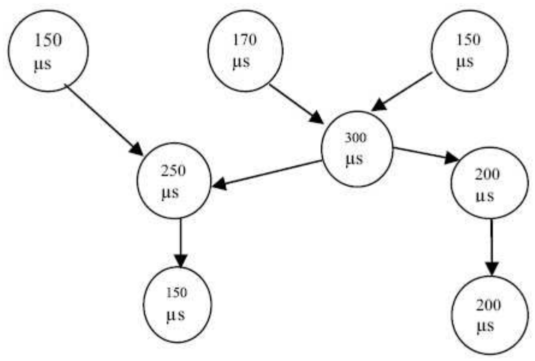


Figure 5.6: Cruise control task graph

5.4.1 Electric Steering Control

Electric steering work performed by the fly-by-wire system during landing is partitioned into a set of tasks acquiring some execution time and the interdependent tasks are represented by a task graph $G_1(V,E)$,where V represents the tasks and E represents the interdependency cost, E is taken as directed due to precedence constraint

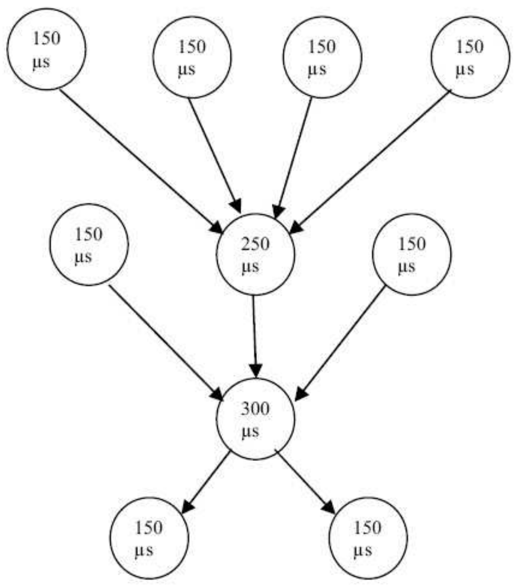


Figure 5.7: Traction control task graph

,i.e.T_4(Desired hand-wheel effort) task, time period for the directed acyclic task graph is taken as critical path will be performed after the task T_1(determination Hand-wheel position) and T_2(Determination of road-wheel force).

5.4.2 Cruise Control

Cruise control work is accomplished by the fly-by-wire system during landing is also partitioned in to different types of task. First the distance of the object near which the flight will stay in ground is determined, as well as the current speed of the flight is determined then desired speed is maintained for proper and smooth landing of the flight. In second step current throttle position is determined, inwhich a separate task is done simultaneously with the tasks associated with the first step. Depending on the required speed and current throttle position, breaking force (a task) is applied and command is given to the actuator present at the break and throttle. In this directed acyclic task graph, the output of one task is feed as the input to another task. No backward movement occurs in the task graph.

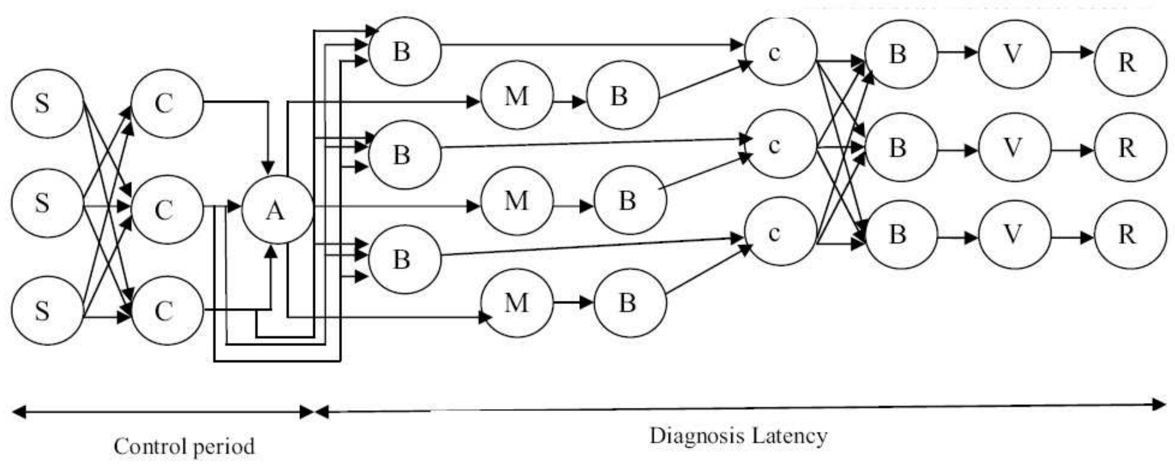


Figure 5.8: Actuator Diagnosis Task Graph integrated with Actuator Control Task graph

5.4.3 Traction Control

During landing, traction control is done by determining the wheel speed of all 4 wheels (2 front wheels and 2 rear wheels) so that the yaw rate can be calculated , simultaneously hand wheel position and lateral acceleration is calculated as two separate tasks. Then desired breaking force is determined to actuate the throttle and break.

5.5 Task Graph Scheduling Algorithm

The efficiency in performance of fault diagnosis in the fly-by- wire system depends on two factors. One factor is the time difference in between reading the sensor and commanding the actuator. Another factor is the time needed for diagnosing the faulty actuator by the fault free microcontrollers.

5.5.1 Control Period

Distributed system comprises of smart sensors, microcontrollers, actuators. Sensors sense, microcontrollers compute actuation command, Computed commands are voted and actuator works upon the voted command. Hence sensors, microcontrollers control the actuators in repeated time interval. In a multi-rate system, different task graphs have different control period, as the functionalities are different.

5.5.2 Diagnosis Latency

The time duration within which, the faulty actuator is diagnosed after command is given to the actuator and the system become safe is termed as diagnosis latency. It is denoted by t_d. Diagnosis Latency can be defined in another way, the total time needed by the checker tasks to generate the residues and the voter tasks needed to compare the residues along with the communication time needed for message passing through communication channel.
Suppose

- T_{unsafe}= Time within which the physical system becomes unsafe.
- T_{evalc}= Time needed for comparing the actual actuation command with the actuation command generated from the mathematical model.
- T_{evalr} = time for execution of the voting algorithm in each processor during each step of actuator control
- t_c = time taken for passing message among microcontrollers.

Diagnosis latency t_d within which actuator is to be diagnosed should satisfy the inequality given below:

$$t_d \leq (T_{unsafe} - T_{evalc} + T_{evalr} + t_c) \tag{5.1}$$

5.5.3 Tasks in Embedded Task graph Of Actuator Control and Diagnosis(adopted from [21])

The execution time range of different types of tasks in the above table is taken from literature survey. Execution time of different tasks are obtained randomly during simulation.

5.6 Description of Graph Scheduling Algorithm

In the fly-by-wire system, the execution period of each task graph is its critical path. As the system is multi-rate in nature the control period of each task graph is different

Table 5.1: Task type, Task name, Task execution time

Task Type	Task name	Execution Time Range(µsecond)
S	Sense	150-200
C	Compute	400-500
A	Actuate	250-300
B	Buffer	50-50
C	Check	400-500
V	Vote	350-400
M	Monitor	400-500
R	Recover	350-400

but smaller than the execution period. So successive iterations of different task graphs are overlapped to get a feasible schedule. As all the graphs are sequential and directed, all the task graphs are pipelined to achieve both the desired control performance and diagnosis latency. The concept of pipelining or overlapping of task graphs is taken from the concept of software pipelining.

The different tasks in the task graph are assigned priority according to their appearance order in the task graph [1]. The tasks of the task graphs are sorted according to their priority. Task having the tightest schedule is selected and assigned to the microcontroller if release time and deadline of the task are less than microcontroller execution time then that task is allocated to that particular microcontroller. If release time and deadline are greater than the microcontroller execution time then new microcontroller is allocated for that task. If release time is less than microcontroller execution time and deadline is greater than the processor execution time the task has splitted in to two subtasks. One subtask is assigned to the time slot of the existing microcontroller and another subtask is assigned to a new microcontroller. The same process is repeated until all the tasks are scheduled and allocated to the microcontroller.

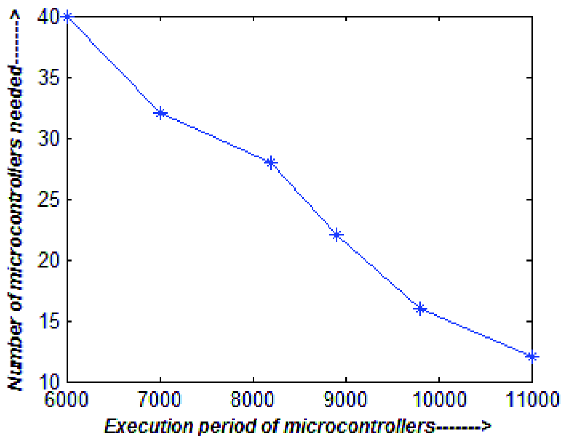


Figure 5.9: Increase in number of microcontrollers with increase with microcontroller execution time

5.7 Simulation Result

Matlab is used as a simulation tool. The simulation results shown below in figure 5.9, gives the number of microcontrollers required at different control periods and different diagnosis latency. Both the control period and diagnosis latency are varied with slack values. Deadlines are categorized as

- Tight if (0 <= *slack* <= 0.3)
- Medium if (0.3 <= *slack* <= 0.7)
- Loose if (0.7 <= *slack* <= 1.0)

As the deadline of the task graph becomes tight with slack, the number of processors required is more. When the deadline of the task graph becomes medium, the number of microcontrollers required is less than the number of microcontrollers required in tight deadline. When the slack value is within 0.7 to 1.0 that means in loose deadline the number of microcontrollers required is the minimum.

5.8 Conclusion

An algorithm for fault diagnosis of a multi-rate fly-by-wire system has been proposed. In this work the comparison based approach is used for diagnosing the faulty actuators. The proposed approach has been evaluated using the parameters such as number of microcontrollers required for accomplishing both control and diagnosis tasks. The result shows implementation of a FBW system is feasible and is proved to be the best among other diagnosis approaches. The time complexity of the fault diagnosis algorithm is O(r*k) for r rounds in the k-fault tolerant system like Fly-By-Wire(FBW) system. The GA-based approach can very well be applicable to Fly-By-Wire system being a small scale distributed newtwork. In fact, this is our future work.

Chapter 6

Conclusion

6.1 Conclusion

The fault diagnosis problem is an important problem in distributed systems. A vast collection of diagnosis algorithms provide a guiding framework and motivation to carry forward the research in this direction. In the thesis, chapter 1 and 2 give a brief introduction of system level fault diagnosis and the work already proposed in this field, where total time taken by the algorithms are mostly dependent upon (i)the time of test messages transferred by all nodes to their neignbourhood nodes (ii) the time needed for performing computation(comparison) in individual processor nodes (iii)the time needed to transfer the test outcome message to the neighbourhood nodes to collect the actual outcome from the local diagnosis outcomes.

In chapter 3 and 4, GA-based fault diagnosis algorithm for diagnosing permanent and intermittent faults is described. Once the test sysndrome is collected, the algorithm executed in the ultra reliable node diagnose the status of each node in the system. Hence the communication paths in between the nodes become free from the task of transferring message obtained from local diagnosis to complete the diagnosis task. So other application specific tasks those need the communication link can use the path more efficiently. The correctness of the proposed permanent and intermittent fault diagnosis algorithms have been proved. The time complexity and message complexity of GA-based permanent fault diagnosis algorithm(GAPFDA) are

O(n*P*K*ng) and O(n*K) respectively, where 'n' is the number of nodes, 'P' is the population size, 'K' is the connectivity of the network, 'ng' is the number of generations(steps). The time complexity and message complexity of GA-based intermittent fault diagnosis algorithm(GAIFDA) are O(r*n*P*K*ng) and O(r*n*K), where 'r' is the number of rounds the GAIFDA gets executed to get the complete fault set. The notations 'n', 'P', 'K', 'ng' used in O(r*n*P*K*ng) have the same meaning as the notation used in O(n*P*K*ng) and O(n*K) of GAPFDA. Now-a-days automotives are designed to run by wire rather than by traditional system. In Chapter 5, we have presented the fault diagnosis algorithm for diagnosising actuator fault of a Fly-by-wire(FBW) system with in a designer-specified deadline. The time complexity of the proposed algorithm is O(r*k) for r rounds in a k-fault tolerant system like FBW. Along with the faultdiagnosis algorithm, one efficient scheduling algorithm is evaluated, that schedules taskgraphs of different time period to various microcontrollers of the FBW system.

Our future work is to design and evaluate fault diagnosis framework that includes all kinds of faults for distributed network such as industrial communication network. We will also see the result after applying the fault models other than PMC model.

Bibliography

[1] Adam, T., Chandy, K., and Dickson, J., "A comparison of list schedules for parallel processing systems," *Comm. ACM*, vol. 17, pp. 685–690, December 1974.

[2] A.Townsend, "Genetic algorithms-a tutorial," tech. rep.

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

[4] B.Ayeb, "Fault identification algorithmc:a new formal approach," *In Proc. 29th Int. Symp. Fault-Tolerant Comput.*, 2001.

[5] Bianchini, P. and Buskins, R., "An adaptive distributed system level diagnosis algorithm and its implementation," in *in proc. FTCS-21*, pp. 222–229, 1991.

[6] Blough, D. and Brown, H., "The broadcast comparison model for on-line fault diagnosis in multicomputer systems:theory and implementation," *IEEE Transactions on Computers*, vol. 48, pp. 470–493, 1999.

[7] C.M.Krishna and Shin, K. G., *Real-Time Systems*. McGraw-Hill Higher Education, 1996.

[8] Couloris, G., Dollimore, J., and Kinberg, T., *Distributed Systems - Concepts and Design*. UK: Addison-Wesley, Pearson Education, 4th ed., 2001.

[9] Dias, F. M. and Antunes, A., "Fault tolerance of artificial neural networks:an open discussion for a global model," *INTERNATIONAL JOURNAL OF CIRCUITS, SYSTEMS AND SIGNAL PROCESSING*, vol. 4, 2010.

[10] D.Wang, “Diagnosability of hypercubes and enhanced hypercubes under the comparison diagnosis model,” *IEEE Trans. On Computers*, vol. 48, no. 12, pp. 1369–1374, 1999.

[11] Elhadef, M. and Ayeb, B., “An evolutionary algorithm for identifying faults in t-diagnosable systems,” pp. 74–83, 2000.

[12] Elhadef, M. and Ayeb, B., “Efficient comparison-based fault diagnosis of multiprocessor systems using an evolutionaryapproach,” in *15th Int. Parallel and Distributed Processing Symposium (IPDPS-2001)*, (San Francisco,CA, USA), April 2001.

[13] Elhadef, M., “A perceptron neural network for asymmetric comparison-based system level fault diagnosis,” in *International Conference on Availability,Reliability and Security*, 2009.

[14] Elhadef, M., Das, S., and Nayak, A., “System-level fault diagnosis using comparison models: An artificial-immune-systems-based approach,” *JOURNAL OF NETWORKS*, vol. 1, pp. 43–53, September/October 2006.

[15] et al, J. B., “Flexray-the communication system for advanced automotive control systems,” in *SAE World Congress*, 2001.

[16] F., B. and P., G. F. M., “A theory of diagnosability of digital systems,” *IEEE Transactions on Computers*, vol. C-25, pp. 585–593, June 1976.

[17] F, H., *Graph Theory*, ch. Addison-Wesley Series in Mathematics. 1972.

[18] J., P., “Random networks,” 2002. http : //www - f1.ijs.si/ rudi/sola/RandomNetworks.pdf.

[19] Jalote, P., *Fault Tolerance in DistributedSystems*. PTR Prentice Hall, 1994.

[20] J.Fan, “Diagnosability of crossed cubes,” *IEEE Trans. On Computers*, vol. 13, pp. 1099–1104, Oct 2002.

[21] J.G., K. and S.M., R., "Fault-diagnosis in fully distributed systems," in *In Proc. of 11th Inter. Symp. on Fault-Tolerant Comp.*, pp. 100–105, IEEE Computer Society Pub., June 1981.

[22] J.Maeng and M.malek, "A comparison connection assignment for self-diagnosis of multiprocessor systems," in *In Proc. of 11th Intl Symp. Fault-Tolerant Computing*, pp. 173–175, 1981 1981.

[23] K., S. A., K., A. V., and D., A., "A generalized theory for system level diagnosis," *IEEE Transactions on Computer*, vol. C-36, pp. 538–546, May 1987.

[24] Kandasamy, N. and Hayes, J. P., "Time constrained failure diagnosis in distributed embedded system: Application to actuator diagnosis," *IEEE Transaction On Parallel And Distributed Systems*, vol. 16, pp. 258–270, March 2005.

[25] Koptez, H. and Gruensteidl, G., "Ttp-a time-triggered protocol for fault-tolerant real-time systems," in *Proc. IEEE Fault-Tolerant Computing Symp*, pp. 524–532, 1993.

[26] M.Elhadef and et. al, "A distributed fault identification protocol for wireless and mobile adhoc networks," *Journal of Parallel and Distributed Computing*, vol. 68, pp. 321–335, March 2008.

[27] M.Malek, "A comparison connection assignment for diagnosis of multiprocessor systems," in *In Proc. of 7th Intl Symp. Computer Architecture*, pp. 31–35, 1980.

[28] M.Stahl and et. al., "On line diagnosis in general topology networks," in *in proc. of Workshop Fault-Tolerant Parallel and Distributed Systems*, July 1992.

[29] Nassu, B. T., Jr., E. P. D., and Pozo, A. T. R., "A comparison of evolutionary algorithms for system-level diagnosis," in *GECCO'05*, (Washington, DC, USA), pp. 2053–2060, June 2005.

[30] of the 4th International Symposium on Parallel, P. and (ISPDC05), D. C., eds., *An Artificial Immune System for Efficient Comparison-Based Diagnosis of Multipro-*

cessor Systems, (School of Information Technology and Engineering University of Ottawa, Ottawa, Canada), 2005.

[31] P., P. F., G., M., and T., C. R., "On the connection assignment problem of diagnosable systems," *IEEE Transactions on Electronic Computers*, vol. EC- 16, pp. 848–853, December 1967.

[32] Pelc, A., "Optimal fault diagnosis in comparison models," *IEEE Trans. Computers*, vol. 41, pp. 779–786, June 1992.

[33] P.M.Khilar and S.Mahapatra, "A distributed diagnosis approach to fault tolerant multi- rate real-time embedded systems," *In proc. of 10th Intel Conf. on Information Technology,*, pp. 167–172, December 18-20 2007. Rourkela,India.

[34] P.M.Khilar and S.Mahapatra, "A hierarchical approach to fault diagnosis in large-scale self-diagnosable wireless adhoc systems," *International Journal of Theoretical and Applied Information Technology*, vol. 3, pp. 25–44, October-December 2007.

[35] P.M.Khilar and S.Mahapatra, "Intermittent fault diagnosis in wireless sensor network," in *In proc. of 10th Intl Conf. on Information Technology, ICIT 2007*, (Rourkela, India), pp. 145–147, NIT, Dec 18-20 2007.

[36] S., C., "A framework for fault diagnosis of networks," thesis, Swiss Federal Institute of Technology, Lausanne,Switzerland, 1995.

[37] S.LATHA and DR.S.K.SRIVATSA, "Topological design of a k-connected communication network," in *Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications*, (Corfu Island, Greece), pp. 6–11, February 2007.

[38] Somani, A. K., "System level diagnosis: A review," tech. rep., 1997.

[39] Stanton, N. and P.Marsden, "From fly-by-wire to drive-by-wire: Safety implications of automation in vehicles.," *International Journal of Human-Computer Interaction*, vol. 24, no. 1, pp. 35–49, 1996.

[40] Subbiah, A. and Blough, D., "Distributed diagnosis in dynamic fault environments," *IEEE trans. on PDS*, vol. 15, pp. 453–467, May 2004.

[41] Subramanian, C. and Subramanian, D. K., "Performance analysis of voting strategies for a fly-by-wire system of a fighter aircraft," *IEEE Transactions On Automatic Control*, vol. 34, pp. 273–279, September 1989.

[42] Tanenbaum, A. and Steen, M. V., *Distributed Systems: Principles and Paradigms*. USA: Prentice Hall, Pearson Education, 2002.

[43] T.Araki and Y.Shibata, "Diagnosability of butterfly networks under the comparison approach," *IEICE Trans. Fundamental*, vol. E85-A, pp. 1152–1160, May 2002.

[44] X.Yang and Y.Y.Tang, "Efficient fault identification of diagnosable systems under the comparison model," *IEEE Trans. on Comp.*, vol. 56, pp. 1612–1618, Dec 2007.

[45] Yadav, N. and Khilar, P., "Hierarchically adaptive distributed fault diagnosis in manet," (Bangalore,India), International conference on intelligent computing, August 2010.

[46] Yang, H., Elhadef, M., Nayak, A., and Yang, X., "Network fault diagnosis: An artificial immune system approach," in *14th IEEE International Conference on Parallel and Distributed Systems*, pp. 463–469, 2008.