POLITECNICO DI MILANO

Facoltà di Ingegneria dell'Informazione

Polo Regionale di Como

Laurea Specialistica in Ingegneria Informatica

Master thesis

Identification of Initial Test set for Incremental Diagnosis

Supervisor: Professor Fabio Salice

Assistant Supervisor: Luca Amati

Author: Saeid Ebrahimi Ghasemabadian

Student ID: 722947

Academic year: 2010

ConteFnts

Abstract

The MS Thesis *Identification of Initial Test set for Incremental Diagnosis* is about analysis and development of a methodology for automatic testing strategy for complex systems. In other words we solve the problem of finding the best test set for beginning the tests for finding the faulty component.

We formulate the problem which we are going to solve as a set covering problem where we need to cover all components of a system (a digital device) with proper tests, in order to identify the most likely component containing a failure, minimizing at the same time the number of tests.

By studying different methodologies which work in the field of set covering problem and analyzing them, we applied a method based on binary optimization.

We introduce first the methodology which have been used to perform test sets called incremental Automatic Functional Fault Detective, so by using the results obtained from applying this method we begin to perform our computations.

We present two methods, dubbed as Sum method and Logarithm method, based on different considerations about the a-posteriori probabilities of test outcomes (conditional probabilities with respect to component faulty status). The translation of the concept of *minimal test set* introduces different types of constraints, producing different results of optimization solution.

For further considerations we made robustness on both of these methods and run the computations, both on in the case of equivalent test cost, and in the case where a custom cost is attributed to each available test. Method efficiency and analysis on timing complete the results.

The Matlab Optimization framework was used to implement the algorithms. The results have been collected, we analyze the behavior of the charts draw from the results and also comparing the results of Sum method and Logarithm method, and analyzing the robustness on them we make the conclusion on the results which shows that the implemented methods returns acceptable results.

By running experiments based on the customization of both methods and collecting the results and comparing the behavior of the result with previous results obtained from methods, we have adequate results.

As conclusion we assume that the introduced methods and results of experiments return satisfying results for performing an automatic testing strategy for complex systems.

Abstract

La Tesi “Identificazione del Test set iniziale per Diagnosi Incrementale” si occupa dell’analisi e lo sviluppo di una metodologia per il controllo automatico di sistemi complessi. In altre parole affronta il problema di trovare il migliore test-set iniziale per garantire la stabilità e il corretto comportamento di una metodologia per la ricerca di componenti guasti in una scheda elettronica.

Il problema che ci accingiamo a risolvere è di tipo “set covering”: al fine di garantire che un qualsiasi guasto su ogni componente di un sistema venga rilevato; obiettivo è minimizzare il numero di test che compongono questo inseime test-set.

Attraverso lo studio e l’analisi di differenti metodologie riguardanti il problema del set-covering, ci si è basati sulla modellizzazione e implementazione di algoritmi risolutivi attraverso la programmazione binaria intera (Binary Integer Programming).

Per l'inizio degli esperimenti dobbiamo, prima di tutto, definire i test da esegure sulla scheda elettronica. La metodologia “incremental Automatic Functional Fault Detective” definisce una strategia ottimale per la scelta dei test da eseguire per minimizzare il numero totale di test. Il presente lavoro si occupa di identificare un sottoinsieme minimale di test la cui applicazione garantisca che l’esecuzione della metodologia iAFD sia in grado di rilevare ogni possibile guasto su un componente.

Due metodi sono stati introdotti, il “Sum method” e il “Logarithm method”. Questi si differenziano per la formulazione dei vincoli di ottimalità sulla copertura dei singoli componenti fornita dai test, basata su una differente interpretazione delle probabilità a posteriori dell’esito dei test stessi.

Oltre al calcolo delle soluzioni ottime su alcuni casi di test, alcune considerazioni sulla robustezza dei metodi sono state estratte. Inoltre, un costo ad ogni test; in questo modo possiamo avere, allo stesso tempo, risultati efficienti dal punto di vista dei costi, supponendo che il tempo medio sia stato misurato in tutti i casi.

Per eseguire gli esperimenti abbiamo utilizzato l'ambiente di ottimizzazione numerica di Matlab.

Dopo che sono stati effettuati tutti gli esperimenti con i metodi "Sum method" e "Logarithm method", il calcolo robusto è stato applicato ad essi e sono stati raccolti i risultati, di cui abbiamo analizzato il comportamento.

Abbiamo quindi confrontato i risultati dei due metodi "Sum method" e "Logarithm method" e analizzato l'effetto del calcolo robusto su di essi; abbiamo quindi tratto una conclusione riguardo i risultati, che dimostra che i metodi implementati forniscono risultati accettabili.

Eseguendo esperimenti basati sulla personalizzazione di entrambi i metodi, raccogliendo i risultati e confrontando il comportamento di essi con i precedenti risultati ottenuti con i due metodi, abbiamo ottenuto risultati simili.

In conclusione, possiamo affermare che i metodi introdotti forniscono risultati soddisfacenti per l'implementazione di una strategia di test automatico per sistemi complessi.

Chapter 1: Goals

In this thesis we work on the analysis and development of a methodology for automatic testing strategy for complex systems, and a specific application of the methodology on a case study based on electronic devices, as it is the case of a network routing device developed at Cisco.

Testing strategies and methodologies are important as well as the development of the product itself. Unfortunately not all products will work once they have been manufactured and it is necessary to test the product before it is shipped in order to ensure that it is working properly on the best and efficient operational status desired from the product. Furthermore, in some cases it is necessary to analyze a systematic approach for the localization of a failure among the different components of a complex system, both to focus the attention of the test engineer on it (for replacement purposes) and to shorted repair time.

Present work is based on the proposed project from a collaboration between Cisco and Politecnico di Milano. Test strategy or testing process needed to be developed around the products of this company in line with the specific requirements, according to the most efficient methods and techniques.

The techniques which we are interested mainly have to be used in industry base products, so it have to be designed in the way that companies test engineers be able to use it in the test laboratories.

When creating a testing method for electronic circuits it is necessary to take account of many aspects including the complexity of the boards, the time required to perform the tests, cost of the tests, and many other factors.

It may be necessary to ensure that elements of the design are changed to enable testing to be accommodated more easily and in a more cost effective manner. As a result it is necessary to develop the test strategy from the earliest stages on the development of the product.

However, we can see that the issues of testing the products have been under attention from the time that vendors and industries started to produce goods in batch sizes, and the products became more complex and including several components. In fact electronic systems, and now a day electronic devices are widely used by all organizations and people, it has a big role in societies, in the case if we have fail in electronic system, it may cause several problems for peoples who are using that system.

In order to develop test method using intelligent techniques, it is necessary to start at the beginning of the project study available methods and techniques, then carry out the development and implementation forward. In order to ensure that the testing method is carried forward, it is wise to create a test strategy document.

As any other industries there are some standards which have to be taken under consideration when we are going to develop our project which has direct use in electronics industry. The widely used and most important standards for electronic and electrical devices are performed and issued by IEEE organization [1], so for in this work we try to follow these standards.

There have been developed several methods by researchers and scientists in field of testing electronic systems using different approaches but the focus of this work is to introduce specific method and techniques.

The Traditional Approaches, such as using Rule-based diagnostic systems represent the experience of skilled diagnosticians in the form of rules which generally take the form “IF symptom(s) THEN fault(s)” [2], trying to find efficient method for testing electronic systems.

There are several soft computing methods to find the best test strategy, the Fuzzy method and Neural networks [2] [3], are some of these approaches, as well combining these two approaches and creating Fuzzy Neural Networks [3] also seems to be efficient solution for creating a testing strategy.

What we are going to solve in generic term is the set covering problem, in general we are given several sets as input. But some sets may have some elements in common.

So we have to select a minimum number of these sets so that the sets which have been chosen contain all the elements that are contained in any of the sets in the input.

One of the most important aspect in testing electronic boards is that which set of elements be tested first, in order to get the most efficient test result according to the parameters such as time consuming, costs and the operational satisfactory. If we interpret the above statement to mathematical language we need to find the best initial set for testing by using intelligent techniques. Therefore we need a set of elements for starting our test which have the most coverage on the board and also the highest possibility that the guilty component be found by performing the minimum number of tests.

In order to go ahead in our project successfully and introduce a complete methodology for testing, by soft computing models we need to divide the work in three main phase or steps. Therefore the project has been divided in three following steps.

First of all we need to find the best and most efficient methods in order to be able to perform the computational experiments. There are several methods in literature that study the set covering problems in general cases as a theoretical problem, as well there are several researches and papers that are specified for a special problem concerning set covering methods.

The set covering methods are very useful in many other industries such as logistics which was almost the first field of applied using set covering issue, in order to find and optimize the traveling destinations, recently it is widely used in biomedical experiments and diagnosis, and also in financial models.

For achieving our goal we need to have a deep study on related methods and analyze them in order to configure the best possible methods for our specific project, for this aim several methods such as the classical set covering method using simplex set covering algorithms which consist of several approaches for solving different set covering problem and have been widely used between researchers to find the best covering sets for different purposes, Lagrangian and Lagrangian Relaxation methods,

and the genetic algorithms are some of the related methods which are used for computing and analyzing.

The second step, after the best methods are identified and configured for our purpose we have to do the computational analysis, which is one of the important parts of this work. Several tests will be performed using at least two methods. The reason that we have to perform the tests using more than one method is that we need to compare the results of the methods in order to assure the obtained results.

In order to do the soft computations we need to use computation applications or software, for this part we look over different computational software, there are several software such as GLPK (GNU Linear Programming Kit) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP) [4], IBM High-performance mathematical programming engine (IBM ILOG CPLEX) [5], and MATLAB [6] one of the powerful computing mathematical applications that is used widely by mathematicians and also have the best compatibility with windows operating machines [7].

Therefore the MATLAB [6], computing application will be used for our computational work in this research paper.

The third step, after all when we already configured the methods that we want to use and also we did the computational experiments with those methods, now in this step we have several results obtained from our computing. In order to give a valuable meaning to our result we need to analyze them, which is the goal of this part of work, as well as it is known for analyzing different results we need to use parameters and bench marks. Further we need to define our parameters and benchmarks for this work.

The parameters which we will use in this work are *scalability;* scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner or to be readily enlarged [9], *complexity*, it is one the important parameters usually used in computer science [10], and *time,* as much as the time consuming by test be less it is better

because in industry time has cost, so vendors are interested in less consuming time operations.

All the results obtained from previous steps of this project will be carefully analyzed in respect to the parameters and standards. This is our goal in this step of the project.

The result of the work is finally satisfying the need of the project which is the best initial set for testing electronic systems for fault diagnosis using intelligent techniques in respect to scalability, complexity, and time.

Chapter 2: Previous Works

2.1. Introduction

One of the most famous problem in optimization is the problem of traveling salesman, in order to find the shortest way to visit all the cities which he needs to travel between them to sale his goods, in this problem we are dealing with optimization of traveling according to mainly two criteria visiting cites and distance between the cities.

The Set Covering Problem have been widely studied, and a lot of different methods for solving the problems have been proposed and developed by researchers working in different fields of science and technology such as bioinformatics, and artificial intelligence.

By considering that solving very complex approximation algorithms need a very large set of computational operations to get proper results.

Nowadays intelligent algorithms by help of powerful computing machines are able to solve more complex approximation algorithms, and have been used by many researches for high level and very complex approximation problems such as Local Improvements, Randomized Rounding, Iterated Heuristic, Genetic Algorithm, and Lagrangian Heuristic [1][11][12].

Particularly the methods and algorithm which are developed to find the best initial test set or in other word the set covering problem is not limited to the methods named above, and all the methods will not be studied in this chapter. One can develop his own methodology in order to find the best method for his specific project by combining the exciting methods or at all introducing a new method.

Finding the best initial test set using intelligent techniques, identifying the best set, will lead the whole project further correctly to find the guilty component as soon as possible with the minimum number of tests.

The focuses of this chapter is to study the previous works related to the research of this thesis. Study the methods and algorithms which other researchers used to find the best initial test set in the electronic circuit industries or in other scientific or industrial fields. In fact we are only interested in methods and algorithms used for computations and the way that they prove that their method is doing well.

2.2. The Set Cover problem

In order to find the best initial test set and satisfy the goals of our work, we need first of all have a clear idea about what are optimization methods and in particular case optimization of set covering problem. In our work the optimized set cover is the best test set for beginning of the tests.

In [13] and [14] gives us very clear explanation, in details about the set covering problem, and the mathematical definition of the set cover problem is also provided and explained, further there are several different issues which are dealing with this problem.

For our work we need to implement the set covering problem according to our specific problem and the general definition cannot be directly used in order to find the initial test set for the electronic circuits. The electronic circuits are very complex and the model circuit influence the initial test set, the better method gives us more information about the potentially fault component.

In general set cover issue is divided in two main parts, scientifically called *unweighted* and *weighted* the difference between this two issue is explained in [8] and [14]. Regarding to the project which we are working on, we will mainly deal with the weighted set cover problem, in order to make our computations for finding the best initial test set.

The unweighted set covering is not useful in our case because if we perform a test which cover several components on the board, all the components will not be covered by same degree of the test coverage, means that we need to assign a value or

weight coverage to each component under the specific test. So we can assume that the weighted set cover problem is more under attention in this work. The methods which will be used have to be able to handle weighted set cover problem.

One of the first issues which go in mind by studying [15] is the minimum set cover; the paper worked on finding the minimum set cover as goal. In fact we also need to obtain the minimum set cover as result in our work as well. In other words means that the algorithms which will be used for computing the test sets, should return as output a set cover of minimum size respect to the coverage degree of each test performed on the components.

The weighted set covering problem, compute a sub-collection of the subsets with the minimized cost. The weight of a subset is the sum of the weights of the elements in the subset, in general a sub-collection of subsets is called a *cover* [8][16]. Further consideration on weighted set-cover shows that each subset has at most k elements, it means that it is also possible to calculate the weighted k-set cover problem.

The greedy algorithm is one of the approximation algorithm for the weighted k-set cover problem. In [17] we can see the explanation of the unbounded values of k and the approximation ratio of the greedy algorithm for the unweighted set cover problem. The definitions of greedy algorithm in [17], are fully explained. In addition in [18] and [19] several issues about approximation problems have been proved and the factors that the unweighated set cover cannot be approximated are explained, so we can see that not always we can have an approximation unweighted set cover.

By reviewing and considering different approaches and problems which are discussed in previous works and the results obtained by different methods, we can assume that greedy algorithm is one of the best possible approximation algorithm for the weighted set cover problem in the most cases, [17][18][19].

As our focus is on weighted set covering problem, so the greedy algorithms can be one of the potential solutions for our project. By sure we cannot assume in this step

of the work that for our specific project it will return the best and most efficient solution, so it is not possible to rely only on this method as it is widely used by other research projects and if they gain perfect result does not mean the results are the best possible according to the parameters which we have in our work. In general we are interested to use different methods and even more reliable ones for achieving the goals of our project.

In [20] the paper works on both, heuristic and exact approaches, taking in to consideration the linear programming techniques. It introduces different algorithms and methods based on lagrangian relaxation, and the computational experiments are done by CPLEX [6]. It compares the result of different algorithms tested by CPLEX [6] by the computing time of each algorithm.

If we consider that usage of an algorithm is better only by lower computing time, we cannot be sure that the results are the most effective ones. Especially in our case the test should cover the component with a acceptable degree of coverage, not only to cover.
But lagrangian relaxation methods and the algorithms which are used for the computations are in interest of our work also. By applying modifications on these algorithms it is possible to obtain proper solutions for our work as well; the algorithms of this model are more complex than other minimization algorithms.

Considering the problem of travelling salesman in this century means the fright forwards, we can see in [21] problem of airfreight forwarders have been solved by Lagrangian relaxation based heuristic, the paper shows that the algorithm implemented base on Lagrangian relaxation returns sophisticated results for set covering problem.

As it was mentioned before in this paper, our focus is on weighted set covering approaches. Weighted Boolean optimization is one of the powerful optimization methods that can be used to solve the set covering problem.

In [22] the Maximum Satisfiability problem have been studied, the paper propose a new algorithm for Weighted Maximum Satisfiability problem which is one of the optimization extensions of the Boolean Satisfiability problem. Further the paper shows that a general algorithm for Weighted Boolean optimization can be as efficient as other dedicated algorithms.

The Weighted Boolean optimization was one of the initial methods which we were trying to work on it, by considering a matrix that tests are columns and components are rows. First of all assign a degree of coverage to each component under test, further by introducing some composition rules, we would obtain some test sets, we cannot prove in this paper that this method gives us the most affective result comparing with other approaches.

To perform fault diagnoses for dynamic systems [3], work on the idea of combining of the Neural Network and the Fuzzy logic together and introducing the Fuzzy Neural Network for fault diagnoses. This method is heavily based on prior knowledge of the system and training data, which is not suitable in our project since that we are working to find the best initial test set for fault diagnosis and we do not work in the designing phase of the electronic boards.

However soft computing algorithms based on linear programming such as lagrangian relaxation methods, and also Weighted Boolean optimization methods such as Pesudo-Boolean optimization, have been used by many solvers, to solve different kind of complex optimization problems.

Well known that the soft computing intelligent techniques are not limited to the methods studied above, there are other intelligent technique such as Genetic Algorithm and Binary Integer Programming.

Genetic algorithms are recently used to solve large kind of problems such as pattern recognition, system classification, control systems, and combinatorial optimization. There are two well known genetic algorithms Sequential or standard genetic algorithms (SGAs) and Parallel genetic algorithms (PGAs) [23]. In fact the sequential genetic algorithm is the modified version of the standard genetic

algorithm, in case each solver according to the kind of the problem for solving choose one of the methods.

A genetic algorithm mainly reproduces a solution through a simple representation, look like a *bit string*, on which transformations are performed in order to find acceptable solutions and the solutions be efficient for computing time or proximity [24]. The main reason that Parallel genetic algorithms have been implemented is to reduce the processing time required in order to achieve an acceptable solution to explore a solution space. In hard problems, large populations impact seriously on the processing time [25].

An Indirect genetic algorithm is proposed in [26], it uses external decoder function to find the solutions, further it shows that the indirect method comparing with direct methods gives better result according to solution quality, speed, and adaptability to new types of problems. The advantage of this method that it can be re-used for other problems unchanged because the genetic algorithm component is almost independent from the problem specific decoder component.

There are several genetic algorithm models for optimization proposed by different researchers. For numerical optimization we can see a sort of methods explained and examined in [27], and computational results have been compared with each other in order to draw conclusion through computational experiments. The numeric results which are obtained, and there comparisons which are performed, are not directly related to our work and we are not trying to prove which method of genetic algorithm is returning the best solution.

The Binary Integer programming method have been introduced in [29], the method looks simple but in fact the algorithm behind the method is very power full and can solve very complex methods. In general we can convert almost any minimization problem to the Binary Integer programming method. The paper solve some examples and compute them. The method seems to return reasonable and well minimized solutions for different examples.

The Binary Integer programming as it is possible to gain from the name of the method will only return the binary solutions of 0 and 1. In the case of our problem it is possible to use this method, because we can convert the inputs of our problem to the standard inputs which Binary Integer programming method use for its computations and after computations we will get a set of binary results, so we can use them for performing the analyzes.

By studying the different papers we can see that mainly the researches use the MATLAB[6], for running the computations and the results obtained from it is highly trustable although that is one the most powerful computing software.

For making the computations for the methods in [28] provide a sort of information on how to use the genetic algorithm tool box in MATLAB [6], and how we can find the optimization by using MATLAB [6]. This application have been used recently by many researchers for solving complex mathematical problems.

The most strong part of the MATLAB[6] software which make it very friendly for using is several ready functions and tools, for example for minimization there is several predefined tools for example for genetic algorithm there is (GA) tool and for Binary Integer programming we can use the (bintprog) tool.

For the computational part of our project we will use the Binary Integer programming (bintprog) tool, according to the papers which we studied in this step of the work the algorithm which is used in Binary Integer programming can satisfy the requirements of the computing phases of this project.

In order to be able to use this method first of all we need to model our problem in the way that it fit the inputs of the Binary Integer programming. As it is well known the results will be binary. Considering this fact we need to have a strong policy to analyze the obtained binary results.

Using this predefined tools speed up the computation parts of the project and also minimize the errors which can appear in the phase of the designing algorithms and

writing codes because the algorithms are tested before and correct results are guaranteed to be obtain if the inputs be well defined. We can also use it for our computations, so we can compute the test sets and therefore analyze the results obtained from the same computing application but using different methods.

2.3. Conclusion

We realize that for our specific problem the weighted set cover methods are more reasonable, because of this fact mainly the papers which work on solving the weighted set cover problem have been more under attention and studied.

When having more precise look in the previous works we can realize that for our purpose we need a very powerful computing method almost more efficient than all of the previous works. Our computations are suppose to cover a large amount of entry data, the reason for large number of entered data is that nowadays the electronic systems are assembled with a lot of components and they are very complex in scheme of integration.

In order to be sure that we obtain reasonable results from our work we need to compare our results. So according to the experience from previous works we will compute our data parallel by using Binary Integer programming tool but with different methods of input data.

In order to perform the computations by using computing machine, we planned to use MATLAB computing application. According to the MATLAB libraries [6], the bintprog tool is the predefined tool for using the Binary Integer programming algorithms so we will use this tool mainly for our computations the complete explanations of how to we use this tool will be given in the next chapter.

Chapter 3: Methodology

3.1. Introduction

In our paper we will describe the methodologies which have been used for researching and performing computations, methods and algorithms for this work. As all scientific works, we started our work with defining the goals of the given project, clarify the goals and targets in order to have better understanding of the project.

The main goal of this research is to find the best initial test set for electronic circuit using intelligent techniques. The goal seems to be very general therefore we break it in three sub goals, in chapter two we studied the previous researches and papers to get idea about the methods which other researchers used, to get clear idea about the different possible methods to perform tests, compute and how to analyze the results obtained from different methods for applying the solutions in different scientific or industrials fields. We can use the observed ideas from previous papers to introduce the methods which we are using in our research work for computing results and finding proper solutions to our specific needs of the project.

Further in this chapter we will see that how the component test matrix (CTM) are generated, and two computation methods which we will use to perform the computations on the test sets obtained from the electronic circuits, with Matlab simulating and computing software. The base of the tool which we are using is the tool for solving binary integer problems, which is proper to our work according to the component test matrix design.

After the (CTM)s are generated we apply the first computing method on them, the first method we directly apply is the binary integer problem solver (bintprog) tools for computations in Matlab, by running several tests on the initial test sets and gathering the data results.

The second method which we use for the computation is again based on same tool in matlab the binary integer problem solver (bintprog) but with modified inputs in this case we do apply the log to our inputs.

Therefore we apply the robust on the both methods, by removing the first solution in any iteration in the initial test set and gathering the results by this way we can have better idea that the test sets are well designed or not.

After all by gathering the results from both methods and the results from robustness applied on the both method, in the next chapter of this paper we work on the analyzing the results in order to show that the according to the obtained results are the methods which we chose for our work are well defined and in the last chapter we will write the conclusions observed from this work.

3.2. The Problem

At the beginning we formulate the problem of finding the best initial test set by using the generated component test matrix (CTM) according to statistical estimates of the coverage degree of each component for each test which have been performed. To do so, we assume that by the following terminology t= $\{t1, \ldots, tn\}$ are the set of tests which are performed to cover the components. Components are sets of parameters that the test specify for each of them a degree of converge on them, **c** = $\{c1, \ldots, cm\}$ the sets for which statistical coverage exist.

An |**c**|×|**t**| matrix, where the (i, j) are entries, and Pij , holds the probability of coverage of a test tj with a test generated for using the set Ci. We also assume that Pij are statistically independent for simplification of the problem. Furthermore, at end by taking in to consideration that these statistical estimates are reliable. The Initial test set is represented by the vector **v** = $\{v1, \ldots, vm\}$, which specifies an activation policy, considering that $v = \sum v_i$ the total number of tests derived by the policy **v**. In fact we need to solve the equation $CTM * x \geq b$, x is the solution matrix which are looking for it. There for by finding the x we have a binary matrix as a result for each of our CTM, the important part is to analyze the results obtained.

After identifying the problem further we need to explain how the component test matrix (CTM)s have been generated. Since the identification of the CTM is a very important part of electronic circuits tests.

As better component test matrix (CTM) be defined the more better result can be obtained in further computations because it has direct relation to the physical layer of the circuit and the affect of tests on the components.

To solve the main problem which is minimization of the initial test sets, in the computational part we need to introduce and explain method which we are going to apply for solving the problem, as it was mentioned before the method which we are planning to use in this work is Binary Integer Programming. For the computations we use the Binary integer programming tool (bintprog) in Matlab minimization tools.

3.3. Component Test Matrix (CTM)

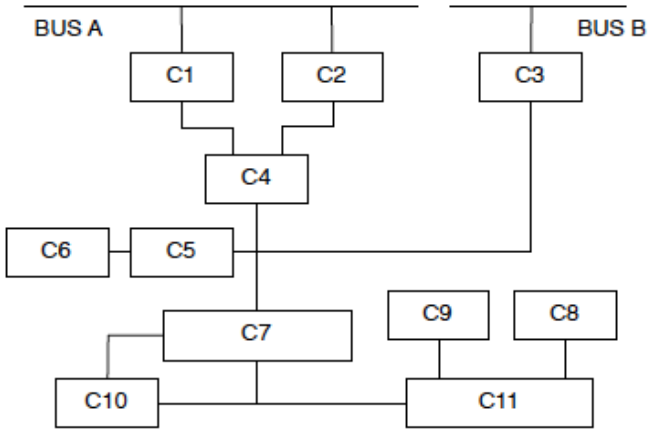
The methods of generating the component text matrix (CTM) is not implemented as a part of this paper, moreover it is not included in the goals of this project, but we will not be able to find the best initial test set without having the results of the affects of each test on the components. The component test matrix (CTM) used in this paper were developed by the previous phase of the project proposed by the CISCO company therefore the researchers from Politecnico di Milano incorporate of CISCO Photonics engineers, introduced a strong algorithm based on Bayesian Belief Network. We will just make a brief introduction about the method and we directly use the components test matrix (CTM)s generated by this algorithm in this phase of the work which is finding the best initial test sets according to the test sets performed by (CTM). As mentioned before the more efficient test set for beginning the tests leads the tester to obtain and more accurate results in shorter time, which for complex systems is an essential point for testing. The complete information can be find in [30].

As mentioned before, the model of the circuit plays a major role in the whole test and diagnosis process. The better is the model, the more reliable are the information about potentially faulty components.

The model is mainly composed by two sections: a first one regarding the physical structure of the circuit under test (in terms of components or/and functionalities) and a second one representing the set of relationships between components and tests. While the first part is objective and available using the design tool which brings from an high level description to the electrical schema, the second one is not so trivial.

An intervention of the test engineers team is needed in order to identify the tests they consider representative to verify the board functionality and to define tests/components relationship.

The output of the test engineers team activity is a table, called Component Test Matrix (CTM) where each entry represent the coverage level a test provides to a component, using a qualitative simplified scale (High, Medium, Low). This simplification has not an huge impact on the whole process, because the probabilistic engine tolerates imperfectness: moreover, the computation of an accurate value would have been really difficult.



	T01
C1	MED
C2	
C3	
C4	MED
C5	
C6	
C7	HIGH
C8	
C9	HIGH
C10	LOW
C11	MED

Illustration 1. An example of (CTM)

3.3.1 iAF2D methodology

A new approach is needed to fill the limitations of existing diagnosis approaches described before. The aim of this new approach is to proceed with an incremental strategy, performing only a subset of the available tests, and, based on the resulting partial syndrome, to select the next test (or tests) to be executed in order to refine the search of the candidate faulty component.

The goal is to limit the number of executed tests, using only those that would actually add information for the diagnosis, saving effort and time. This approach has been called iAF2D: incremental Auto Functional Fault Detective.

The framework which implements iAF2D methodology takes as inputs the model (with CTM) an a partial syndrome (the results of an initial set of tests, designed by the test engineers team). Results are the indication of the next test to be executed and/or the ranked list of the potentially faulty components. More precisely, iAF2D uses a Bayesian Belief Network (BBN) as the probabilistic reasoning engine.

A BBN is a DAG (Direct Acyclic Graph) where nodes represents variables and arcs represents conditional independencies between variables: in general, a BBN models the conditional dependencies of a set of variables.

By adding semantics to this model it is possible to obtain a casual BBN, where each node specifies an event which may happen or not, based on the occurrence of its parents through a conditional probability. In this context, variables are associated to components, which may be faulty or fault-free. Events are test outcomes, which may be PASS or FAIL, as depicted in the following picture.

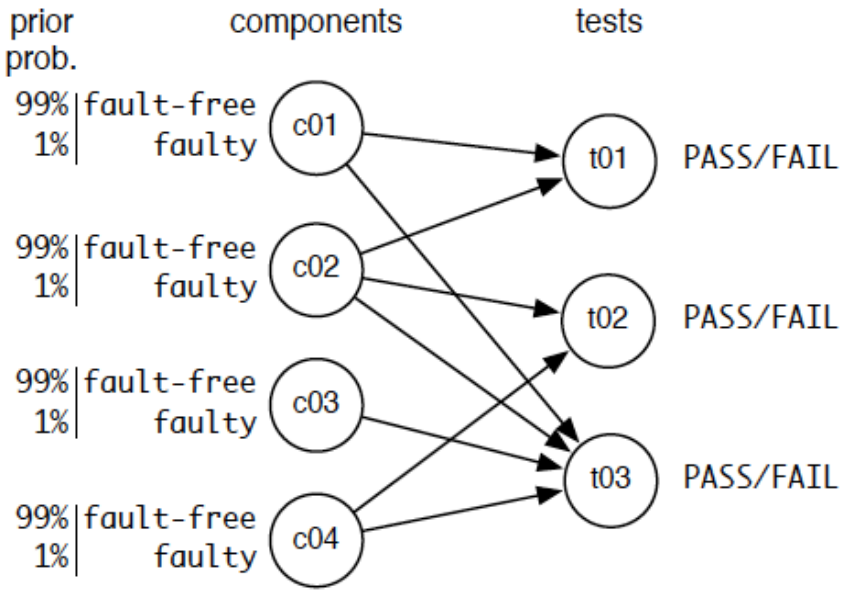


Illustration 2: The BBN model for the iAF2D diagnosis methodology

This translation from CTM to BBN model is part of the first step of iAF2D, which implies the identification of the initial set of tests, too. Results of these initial tests are collected and analyzed in the second step. Further tests are executed if no tests have failed: in fact, in this case, no information can be used to identify a set of faulty candidates.

It is necessary to improve the coverage of the board adding one more test at time. When at least one test fails, the core of iAF2D starts. In this third step the effects of the execution of each remaining test are simulated using the BBN engine. For each remaining test both PASS and FAIL outcomes are considered, calculating all the components' probability to be faulty and the probability for the test to give such an outcome.

All the components that appears in one of the FAILED tests constitutes the Possible Faulty Candidates set (PFC set), used together with the calculated probabilities to compute the indication of the next test to be executed.

A scalar cost function has been defined to take into consideration all these factor, giving a final value for each not yet executed test. These flow goes on until the whole

test set has been used (no more available test to perform) or a stop condition is reached. This condition is when an additional test (the next test suggestion) would not modify the order of the ranked PFC set.

An algorithm and a framework has been designed to implement the described iAF2D methodology. A set of experiments has been performed to check the improvements w.r.t. the classical diagnosis methodologies mentioned before. Results are encouraging, confirming that a solution (the faulty component identification) is reachable in a finite number of steps. The complete algorithm is shown in figure 1.

```
Input:  System Model CTM (Compcnents-Tests Matrix), T (Test Set).
Output:  PFC (Possible Faulty Candidates).
Procedure FindCandidates(CTM,T)
{
 1  initTS = Select_Initial_Tests(T)
 2  T = T - initTS
 3  PartialSyndrome - Execute_Tests(initTS)
 4  PFC = Compute_Possible_Faulty_Candidates(partialSyndrome)
 5  while (PFC = ∅))
 6     T0 = Select_Add_Initial_Tests(T)
 7     initTS - initTS + T0
 8     T = T - T0
 9     PartialSyndrome = Execute_Tests(T0)
 9     PartialSyndrome = Execute_Tests(initTS)
10     PFC - Compute_Possible_Faulty_Candidates(PartialSyndrome)
11  end while
12  while ((T ≠ ∅) and (not STOP )))
13     Matrix = Simulate_Tests(T,PartialSyndrome)
14     Ti - Select_Next_Test(Matrix)
15     STOP = Analize_STOP_Condition(Matrix)
16     if(not STOP)
17        T = T - Ti
18        PartialSyndrome - PartialSyndrome + Execute_Tests(Ti)
19        PFC = Compute_Possible_Faulty_Candidates(PartialSyndrome)
20     end if
21  end while
22  output PFC
}
```

Figure1

Moreover, the incremental approach seems not to reduce the accuracy of the diagnosis, which is affected mostly by the model (including the CTM) and the test

design and selection activity. This means that having a consistent CTM is one of the critical points: that's why the attention is now focused on the identification of an intermediate model to be used by test engineers team when defining CTM.

3.4. Binary Integer Programming (BIP)

3.4.1 Definition

Binary integer programming as it is possible to drive from the name of it each can take only the value of 0 or 1. This my refer to the selection or rejection of an option which we assign the value to it, in general 1 is mentioned as selected and 0 rejected, but not limited to this options and several other options can be used.

We will study the binary integer programming problem of finding a binary vector x that minimizes a linear function $f^t x$ subject to linear constraints:

$$min f^t x \; such \; that \begin{cases} A * x \leq b, \\ x \; binary \end{cases} \qquad (1)$$

Where the *f*, *b*, are vectors in this equation, *A* is matrices, and the expected solution of x a binary integer vector, so its entries can only take on the values 0 or 1.

3.4.2 bintprog Algorithm

bintprog uses a linear programming (LP)-based branch-and-bound algorithm to solve binary integer programming problems. The algorithm searches for an optimal solution to the binary integer programming problem by solving a series of LP-relaxation problems, in which the binary integer requirement on the variables is replaced by the weaker constraint $0 \leq x \leq 1$. The algorithm

- Searches for a binary integer feasible solution
- Updates the best binary integer feasible point found so far as the search tree grows

- Verifies that no better integer feasible solution is possible by solving a series of linear programming problems
- The following sections describe the branch-and-bound method in greater detail.

3.4.3 Branching

The algorithm creates a search tree by repeatedly adding constraints to the problem, that is, "branching." At a branching step, the algorithm chooses a variable xj whose current value is not an integer and adds the constraint xj = 0 to form one branch and the constraint xj = 1 to form the other branch. This process can be represented by a binary tree, in which the nodes represent the added constraints. The following picture illustrate (3), a complete binary tree for a problem that has three variables, x1, x2, and x3. Note that, in general, the order of the variables going down the levels in the tree is not the usual order of their subscripts

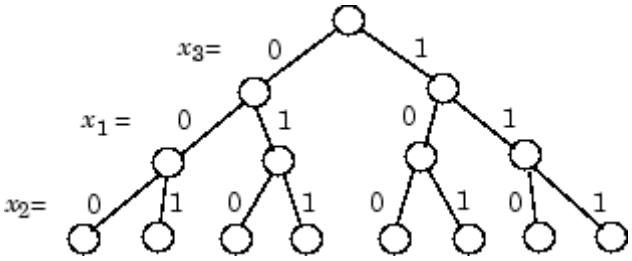


Illustration 3: Binary tree with three variables

3.4.3.1 Deciding whether to Branch

At each node, the algorithm solves an LP-relaxation problem using the constraints at that node and decides whether to branch or to move to another node depending on the outcome. There are three possibilities:

If the LP-relaxation problem at the current node is infeasible or its optimal value is greater than that of the best integer point, the algorithm removes the node from the tree, after which it does not search any branches below that node. The algorithm then moves to a new node according to the method you specify in Node Search Strategy option.

If the algorithm finds a new feasible integer point with lower objective value than that of the best integer point, it updates the current best integer point and moves to the next node.

If the LP-relaxation problem is optimal but not integer and the optimal objective value of the LP relaxation problem is less than the best integer point, the algorithm branches according to the method you specify in the Branch Strategy option.

3.4.4 Bounds

The solution to the LP-relaxation problem provides a lower bound for the binary integer programming problem. If the solution to the LP-relaxation problem is already a binary integer vector, it provides an upper bound for the binary integer programming problem.

As the search tree grows more nodes, the algorithm updates the lower and upper bounds on the objective function, using the bounds obtained in the bounding step. The bound on the objective value serves as the threshold to cut off unnecessary branches.

3.4.4.1 Limits for the Algorithm

The algorithm for bintprog could potentially search all 2n binary integer vectors, where n is the number of variables. As a complete search might take a very long time, you can limit the search using the following options

- MaxNodes — Maximum number of nodes the algorithm searches
- MaxRLPIter — Maximum number of iterations the LP-solver performs at any node
- MaxTime — Maximum amount of time in seconds the algorithm runs

3.5. Methodology Implementation

3.5.1 Introduction

In this section we will implement our problem in order to solve it by using the bintprog logarithm, explained above. As we mentioned before in the bintprog equation (1), and according to the bintprog solver, we need first of all to define the f, b, x vectors and A matrices.

The f is vector containing the coefficients of the linear objective function; in our case we use the f as vector of ones. After initial implementations of the method and obtaining the results of the computation for the customizing part of the methods we will define f in other way which will be explained well further. General speaking the f is the cost function in this work f corresponds to the cost of each test.

The b is vector corresponding to the right-hand side of the linear inequality constraints; in our methods we will assign different values to b vector according to the method which will be used. For the sum method we will use the b vector starting from 0.5 and increasing with 0.1 up to 1.5, for any iteration of the algorithm. For the logarithm method the b starts again from 0.5 but instead of increasing it will decrease by 0.1 up to 0.1, for any iteration.

The A is a matrix containing the coefficients of the linear inequality constraints; we define the A as the component test matrix (CTM). The component test matrix was explained above and the algorithm for generating them. As we mentioned before the generating of the component test matrix is not the goal of this work we use the generated component test matrixes for our computations.

The x is the binary results obtained from the solver, the 1 shows the cover on the corresponding component and the 0 means no cover or not sufficient cover on the component.

Further we will explain the details of each method we used for the computations, we used two main method Sum method, and Logarithm method, moreover the robust and customization have been applied on both methods, and the time have been measured for all the computations.

3.5.2 Sum Methodology

For applying the Sum methodology first of all we need to define the matrix A according the equation (1), the matrix A as we mentioned above is equivalent to the component test matrix (CTM). The component test matrixes in our project have three coverage degrees high, medium, and low.

From the numerical point of view A which is (CTM), is a matrix with the coverage between 0 to 1, assume that if the coverage is equal to 0.1 means low coverage, 0.5 medium coverage, and 0.9 is the high coverage. By this way the highest level of coverage on a single component can be 0.9 multiply by number of tests, means all tests cover that specific component by degree of 0.9.

The f, cost function at this phase of computations is equal to vector ones. And b starts from 0.5 summing by 0.1 up to 1.5,

The result x are the output of the equation (1), matlab solve this equation by using bintprog(f,A,b), but for our work the problem is that $A * x \leq b$ is not the condition which we need for our particular work, we need to solve $A * x \geq b$.

So the equation which we will use is shown as follow:

$$minf^t x \; such \; that \begin{cases} A * x \geq b, \\ x \; binary \end{cases} \qquad (2)$$

Where the variables are same as it was explained above, to solve the problem with the new condition we simply use the bintprog(f,-A,-b) for our computations. As it has been proved in operation research, the feasibility space is not changing through multiplying both sides of all constraints by minus one and changing the equation sign.

The result x which we obtained are binary as we were expecting it, the x is the selected solution from the component test matrixes (CTM) in our computations.

In fact by solving the equation (2) we are summing up the probabilities of covers, and we define a threshold which the sum of our probability should be greater than that, in our case the vector b.

If we have the following matrix(1), as component test matrix (CTM), with probabilities:

$$A = \begin{bmatrix} p(c1|T1) & \cdots & p(c1|Tn) \\ \vdots & \ddots & \vdots \\ p(cm|T1) & \cdots & p(cm|Tn) \end{bmatrix}$$

Matrix (1),

and the vector b:

$$b = \begin{bmatrix} b1 \\ b2 \\ . \\ . \\ . \\ bk \end{bmatrix}$$

Assume that n=k therefore we are solving a set of equation(3) as follow:

$$\begin{cases} p(c1|T1) + p(c2|T1) + \cdots p(cm|T1) \geq b \\ p(c1|T2) + p(c2|T2) + \cdots p(cm|T2) \geq b \\ \\ \\ p(c1|Tn) + p(c2|T2) + \cdots p(cm|Tn) \geq b \end{cases}$$

Equation (3)

By considering the rules of probabilities we know that *p(A and B) = p(A)p(B)*, if A and B be mutually exclusive then *p(A or B) = p(A)+p(B),* we can see the equation (3) cannot return the exact solution to our problem, so we will modify our method to have more exact and precise solution. The Logarithm method which we introduce further will solve this problem.

3.5.3 Logarithm Methodology

In this method same as the Sum method we use the bintprog solver to solve the problem. But there are several differences between these two methods.

In this method we need to define the variables of the equation (1), we begin from A which is the component test matrix (CTM). The component test matrixes in this case also have three degree of coverage high, medium, and low. But in this method we

replace the coverage on the contrary numeric notation, means that high refer to 0.1 and low refer to 0.9 and the medium will keep its value 0.5.

The f or cost function in this phase of method is equivalent to the vector of ones. The b which is the coefficient vector begins from 0.5 by taking away 0.1 up to 0.1.

Now we will configure the equation(1) for this method, as we can see from the name of the method we are going to use the logarithm for this method, we apply the logarithm operator to component test matrixes (CTM) and to the b vector, so we will get the equation(4),

$$minf^t x \; such \; that \begin{cases} logA * x \leq logb, \\ x \; binary \end{cases} \qquad (4)$$

By applying the bintprog(f,A,b), we will obtain the binary results x which were expecting, x is the selected solution from the component test matrix same as sum method.

The reason we apply logarithm into our method is that according to the equation (3) sum method sum up the probabilities but we are interested in *p(A and B),* when we apply logarithm to the probability rules we will have *log p(A)+ log p(B)) = log (p(A)p(B)),* so by this way we will have the equation (5).

$$\begin{cases} \log\big(p(c1|T1) + p(c2|T1) + \cdots p(cm|T1)\big) \leq logb \\ \log\big(p(c1|T2) + p(c2|T2) + \cdots p(cm|T2)\big) \leq logb \\ \vdots \\ \log\big(p(c1|Tn) + p(c2|T2) + \cdots p(cm|Tn)\big) \leq logb \end{cases}$$

Equation (5)

In fact applying logarithm to the solver will leads us to a more naïve results, and the behavior of these results are very good bench mark for analyzing the results.

3.5.4 Example for Sum method

Here we will present an example of sum method, in this example we use matrix (2) with dimension of 3x4 as a component test matrix. The columns are tests and the rows are components, We solve this example manually in order to give the clear idea about the method.

$$A = \begin{bmatrix} H & 0 & L & 0 \\ 0 & M & L & 0 \\ L & H & H & M \end{bmatrix}$$

Matrix(2)

First off all we will replace the H with 0.9, M with 0.5 and L with 0.1 so we will obtain the following matrix:

$$A = \begin{bmatrix} 0.9 & 0 & 0.1 & 0 \\ 0 & 0.5 & 0.1 & 0 \\ 0.1 & 0.9 & 0.9 & 0.5 \end{bmatrix}$$

The vector b in our problem we used 0.5 as starting threshold so for solving the problem we use a vector of 0.5, and the vector x is the result which are selected tests. So by replacing in the equation (2) we will have

$$\begin{bmatrix} 0.9 & 0 & 0.1 & 0 \\ 0 & 0.5 & 0.1 & 0 \\ 0.1 & 0.9 & 0.9 & 0.5 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} \geq \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}$$

by solving the equation we will obtain:

$$\begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

So we can see that by selecting test one and test two we will cover all the components with respect to the threshold.

3.5.5 Example for Logarithm method

The example solved her by logarithm method use the exactly same component test matrix (2), but as it is described in the method we will replace zeros with ones, H with 0.1, M with 0.5, and L with 0.9, so we will obtain the following matrix:

$$A = \begin{bmatrix} 0.1 & 1 & 0.9 & 1 \\ 1 & 0.5 & 0.9 & 1 \\ 0.9 & 0.1 & 0.1 & 0.5 \end{bmatrix}$$

Further we have to apply the equation (4), we use same threshold 0.5 for this example as well so we will have the following equation:

$$\log\left(\begin{bmatrix} 0.1 & 1 & 0.9 & 1 \\ 1 & 0.5 & 0.9 & 1 \\ 0.9 & 0.1 & 0.1 & 0.5 \end{bmatrix}\right)\begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} \geq \log\left(\begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \end{bmatrix}\right)$$

by solving the equation we will obtain:

$$\begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

As we can see the results obtain form both methods are same, the test one and test two are the best initial test sets for our example.

3.6. Further considerations

3.6.1 Robustness of Methodologies

To have more efficient results in this phase of the work we apply the robust on both methods, the aim is to robust the minimized initial test sets. Robustness of the results

gives us a new set of results which can be used for the analyzing the results, and assure that the obtained results from both methods are well defined or not.

For computing this part of the work we begin from the results obtained from each method, for Sum method we continue the computations with modification on the component test matrixes, by removing from the CTM the first solution given by first time applying the bintprog(f,-A,-b). further we run the test with the new CTM obtained and the solver bintprog(f, -A, -b) will solve the equation (2) with the new generated CTM. Further we use the results obtained by this robustness for analyzing the results of the Sum method.

We apply the robustness of logarithm method as well, in this case the results obtained from the first time applying bintprog(f,A,b) will be used. The modification will be applied on the component test matrixes (CTM), so same as previous, we will remove from the CTM the first solution obtained. Furthermore by applying again the solver bintprog(f,A,b) with respect to the equation (4) we will obtain a set of naïve results for the logarithm method, which is presented in the next chapter. We will use these results for analyzes of the results.

3.6.2 Customization of Methodologies

The customization of the methods is one of the important parts of this work, as we mentioned before that the project is proposed by the CISCO Company, and the company would like to apply the results in real production cycle.

For this aim we need to configure the methods to be useful for industrial base work. As we explained before in the method of obtaining the component test matrixes, we assumed that the test matrixes are generated by using real electronic circuits, so the component test matrixes have direct relation to the physical productions.

We can assume from the parameters which we used in this computations for both method the CTM which in equation (2) and (4) shown with latter A, is already defined form real production.

The only parameter used which can affect the usage of the methods we implement in this work is the cost function of f, for the first part of computations and as well for robustness we used vector ones, but for customization part of the method we apply the cost of the tests, and instead of the vector of ones we use the cost.

By this way the results obtained will be compared and analyzed with the theoretical results so we will have a clear knowledge that the methods are suitable for the industrial base tests.

3.6.3 Timing

As we mentioned before one of the parameters in our work is the computation time so for collecting the results of the time, we measure the time of computations by running twenty times the computations for Sum method and logarithm method, and as well for the robustness of both methods, and for the customized computations. then by gathering the data we have average time of computing of each method and also the robust parts. For measuring the time we use the cpu clock, by applying the tic toc, in Matlab code.

Chapter 4: Results

All the results shown in this chapter are performed by the methods explained in the previous chapter. The tests are run by using Matlab version 2008b, installed on a personal computer (PC), with the following performance Intel® Core™2 Duo Processor P8600 (2.4 GHz, 1066 MHz FSB, 3 MB L2 cache).

After the computations have been executed and terminated we copy the result from Matlab to Microsoft spreadsheet (Excel). In this step we can now analyze the results, by performing table and drawing proper charts for each part of the results. By using the charts which are shown in this chapter we can further drive a strong analysis on them, and also use them to have better view of the results.

The result which are shown here are computed and generated with Matlab by using initial test set obtained from component test matrix generator and the specific methodology explained in previous chapter. We used two set of matrixes for our computations with different dimensions. The first 10 set of matrixes have the dimension of 20x10 and the second set of matrixes with the dimension of 50x20.

The initial test sets which have been used for the computations will not be showed in this paper because it is out of focus of this paper, and we are only interested in the results obtained from those initial test sets or in other words component test matrixes.

4.1 Sum Methodology Results

By applying the sum methodology mentioned before, after computing initial test sets on Matlab and obtaining the results as it was expected the results are in format of binary, sets of zero and ones. The one index means that the component corresponding to the specific test has been covered by that test according to the coefficient assign to that test.

The test with more ones cover more components, in fact if we choose to start the tests, which in the specific coefficient are covering a lot of components, means beginning to find the faulty component among the larger group of components so it

can increase the time of testing in the next phase of the work, identifying the exact faulty components in the circuit.

But we have certain cost function, and we compute component test matrixes for the method one, by the coefficients starting from 0.5 up to 1.5 by adding the value 0.1 to the coefficient after each iteration, in the algorithm used for computing.

The results shown in the chart (1), shows us for each of assigned coefficients to the component test matrixes the values obtained, the algebraic sum of the ones. EX_1, EX_2 and so on are the initial test sets or matrixes which we used as the input to the system.

As we can see in the chart (1), the results are shown for ten first test matrixes that we use as initial test sets. We have ten components under test and twenty tests, so far at the lowest coefficient at least two components are covered, and in the highest coefficient maximum of nine components are covered by corresponding test. As we can see in the chart (1) the number of covered components increase by the increments in the value of the coefficient.

In chart (2) the same methodology as chart(1) has been applied, the only difference in the chart (2) is that the component text matrixes which we used as input for the computations in Matlab has the dimension of 50x20, so in this phase of computing we are dealing with larger matrixes.

It is shown in chart (2), that the minimum numbers of components covered are three and the maximum numbers of covered components are eight. Same as the chart (1), the number of covered components are increasing by increasing the value of the coefficients.

In general the results were expected according to the methodology used for computations and the obtained results are satisfying the goals we were looking for in this phase of computations.

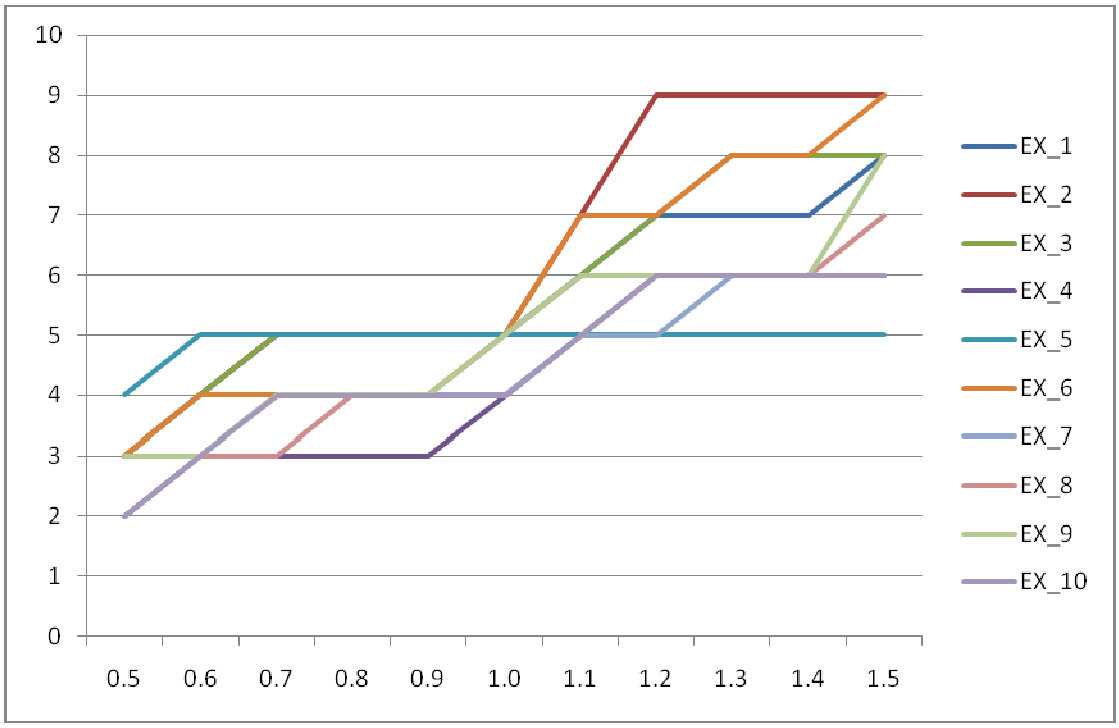


Chart (1),

Results for first ten initial test sets computed by applying the sum methodology from EX_1 to EX_10

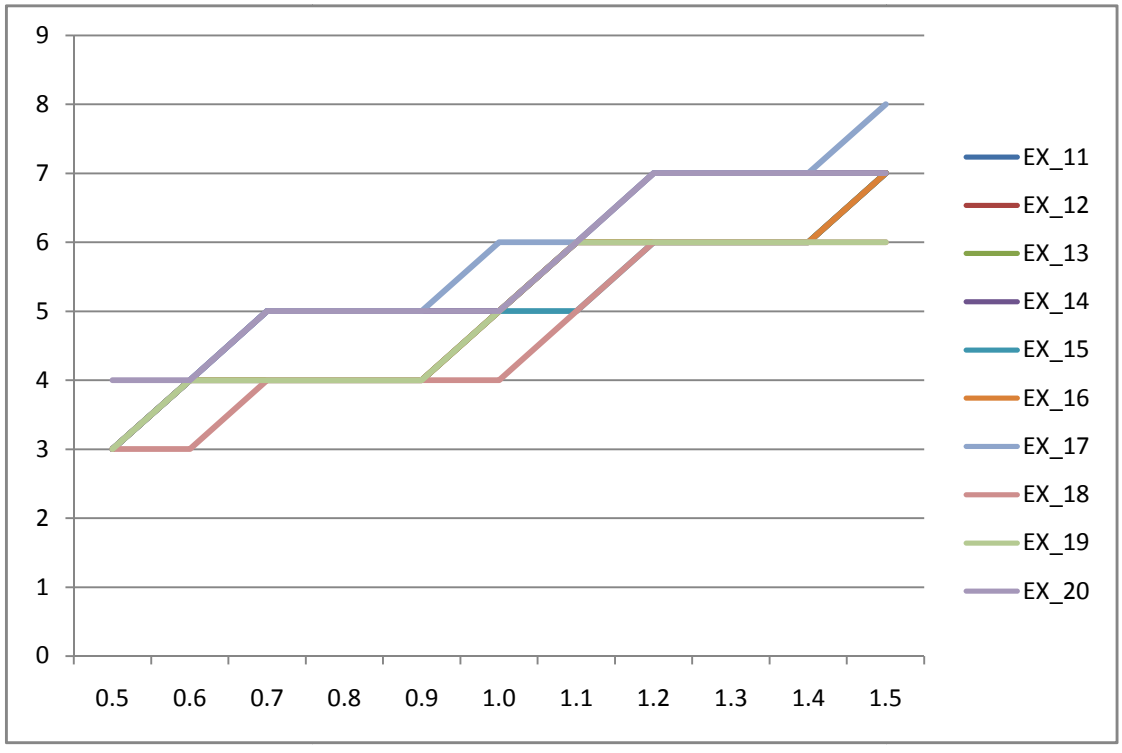


Chart (2),

Results for second part, ten initial test sets computed by applying the sum methodology from EX_11 to EX_20

4.1.1 Sum Methodology Robust Results

In this part the results obtained from applying the robust on the sum methodology will be presented in chart (3) and chart (4). It has been mentioned before that we remove the cover of the components in first row of the component test matrixes and run the test sets with the sum method again. Chart (3) is the result for the first ten test matrixes; we can see the results according to the coefficient, the minimum numbers of covered components are two and the maximum in this case is ten.

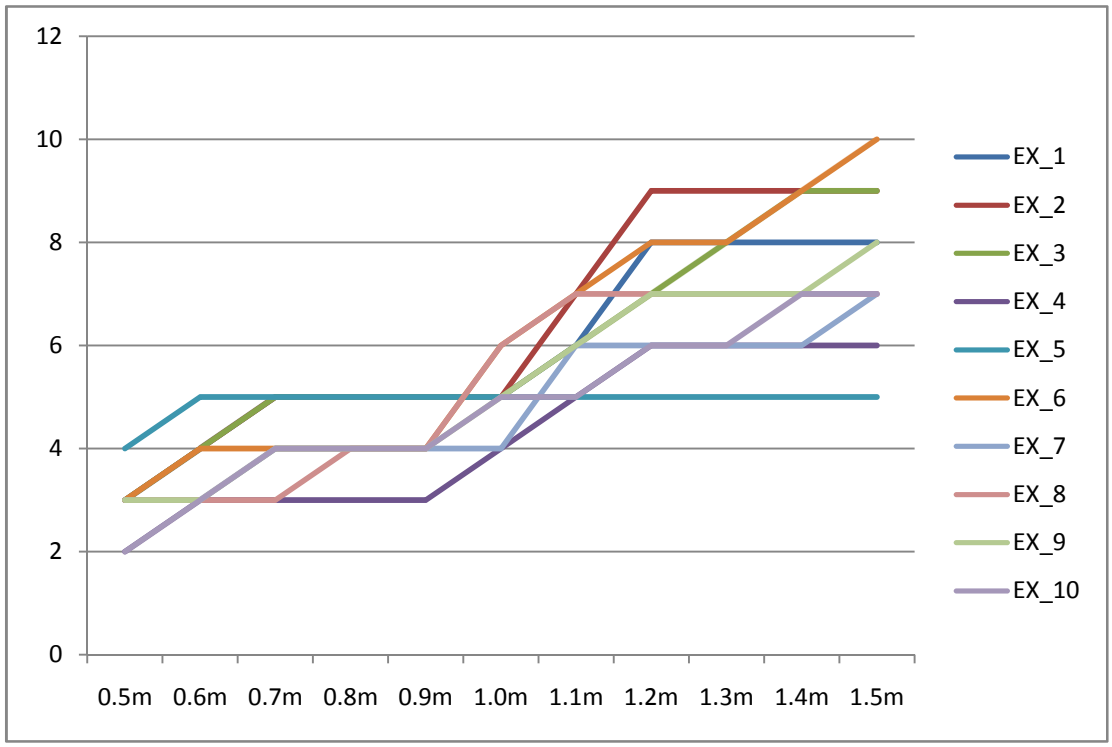


Chart (3),

Results for first ten initial test sets computed by applying the robust on sum methodology from EX_1 to EX_10

Chart (4) displays the results that obtained by applying the robust on sum methodology, but on the EX_11 to Ex_20, the minimum numbers of covered components are three and the maximum is eleven.

It is possible to see that using larger component test matrixes necessarily do not increase the number of covered components; this means that if we have well defined component test matrixes (CTM), with testing the proper number of the components

we can find the guilty components and we do not need to test very larger number of components if the circuit become more complex.

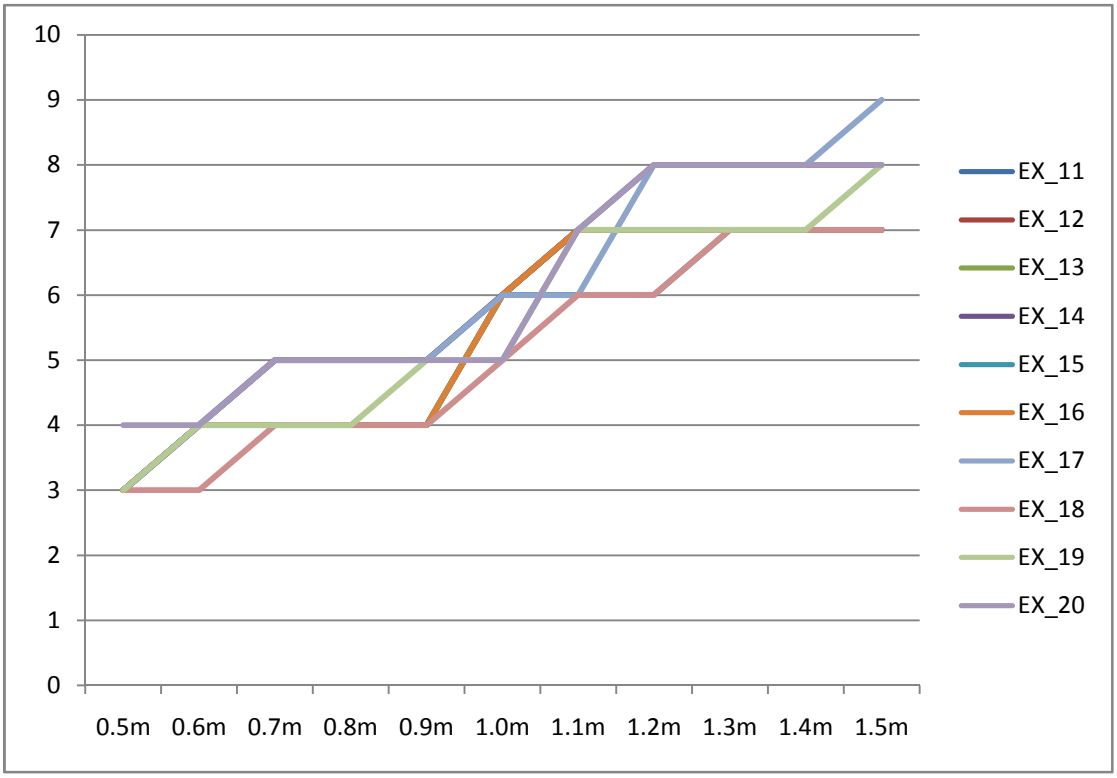


Chart (4),

Results for second part, ten initial test sets computed by applying the robust on sum methodology from EX_11 to EX_20

After obtaining results for the sum methodology and robust it in the next step we will compare the results and define the differences.

4.1.2 Comparing Sum and Robust Results

In order to have some clear idea about the differences of the sum method and the robust one we decide to integrate the data obtained from both methods together with respect to the dimension of the component test matrixes.

In chart (5) we integrate the results obtained from sum method and robust of it for EX_1 to EX_10, and in chart (6) integrated results for EX_11 to EX_20. The results

have been linearly integrated in order that result from same coefficients for sum method and robust one are just after each other and then for the next coefficient up to the number 1.5.

In general speaking and according to the results obtained if the component test matrix be generated in the proper way with respect to the physical layer of the circuit and be well defined, we will see that several tests has same degree of covering on components and they cover the reasonable number of components of the circuit.

If we apply the robust method the algebraic sum of the covered components will not change with very highly difference or in the best case may not change at all, in the coefficient with low value.

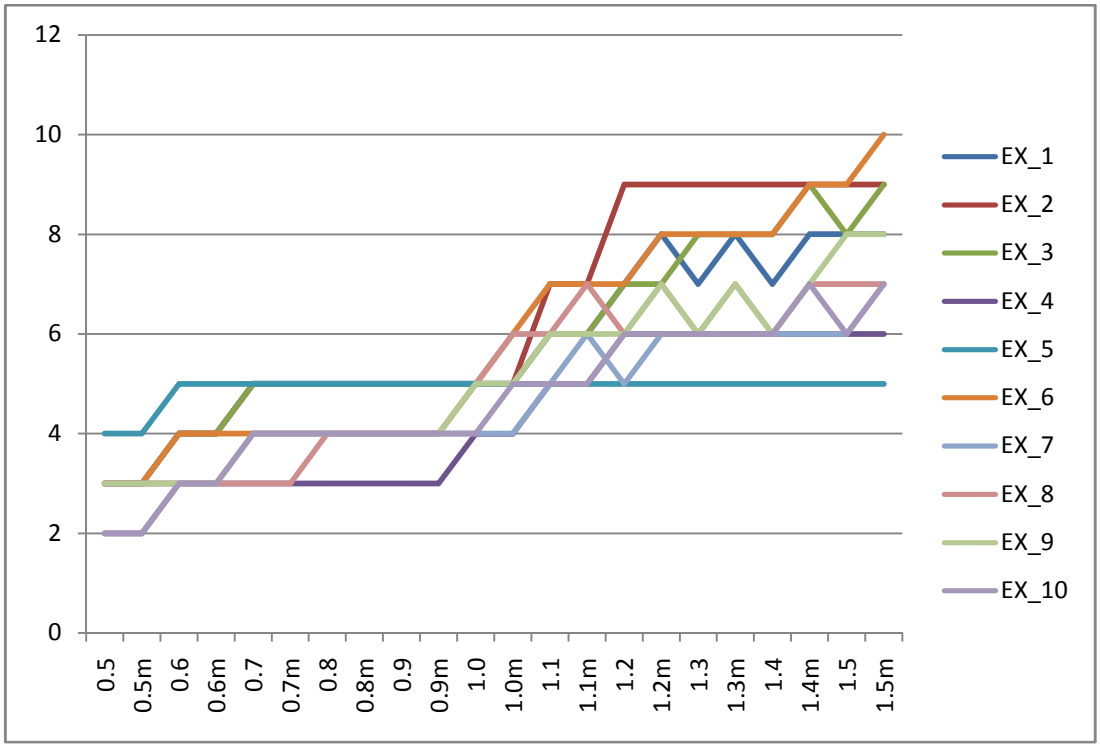


Chart (5),

Results for first ten initial test sets integrating robust and sum methodology from EX_1 to EX_10

In chart (5), following the chart lines we can see the zigzag in the lines, means some times the line is losing value, we will realize that the drop points are sum method and

the pick points are the robust ones, as the value of the coefficient get larger and larger the zigzag part appears in more examples.

In chart (6), we can see as well the same affect which we explained for chart (5). It is possible to see that both set of components test matrixes with different dimensions are displaying same kind of results, so we assume that the tests are performed correctly and the algorithms designed are working properly.

The reason that the algebraic value of the robust model sometimes increase is that when we robust the model in fact we removed one of the potential solutions. In some examples that one can be the critical solution and most efficient one and removing that set affect the whole covering on the system, but if the component test matrixes are well defined removing one of the potential solutions will not fail the test results, just will have affect on the initial test set. In our case the EX_5 is one of the component test matrixes which is well defined and the sum method and robust method are giving the same result.

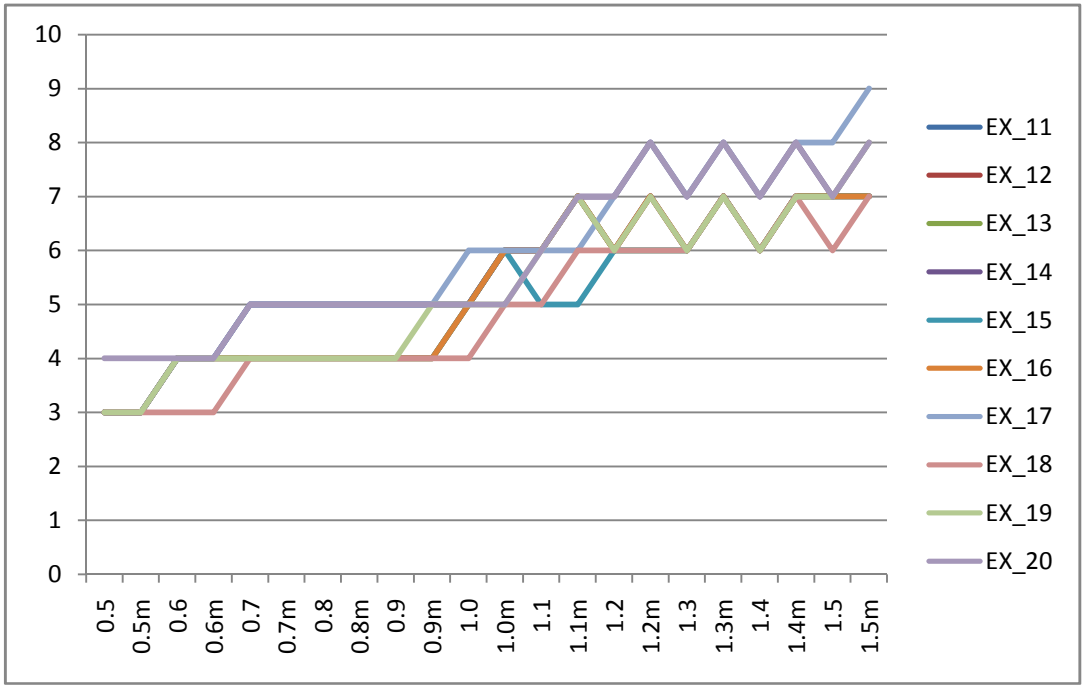


Chart (6),

Results for second part, ten initial test sets integrating robust and sum methodology from EX_11 to EX_20

Further we will have some discussions about time for the sum method and in addition the robust part.

4.1.3 Sum and Robust Timing Results

The time of the execution of the methods in our work is one of the indicators that we are interested to measure it. In this section the results of measuring the cpu clock of the system while computing the sum method and the robust part for the sum method has been measured.

By one time computing and collecting the time, we cannot by sure complain that the time is proper. For avoiding any technical problem of the system for example running some automatic updates and unexpected application we measure time by average of twenty times of running the same algorithm for sum method and as well when we robust it.

In chart (7), the results of the measuring time for the first set of matrixes are shown as we can see as the numbers of the coefficients are increasing the time average increases as well for that point but not necessarily the computation time for all examples always increase by increasing the value of the coefficient assume that we expect that, because of the way that the bintprog function work.

By sure having the low time of the computation in any type of computational works is an advantage as well as in our work it is one of the advantages to find the best initial test set in the less time as possible.

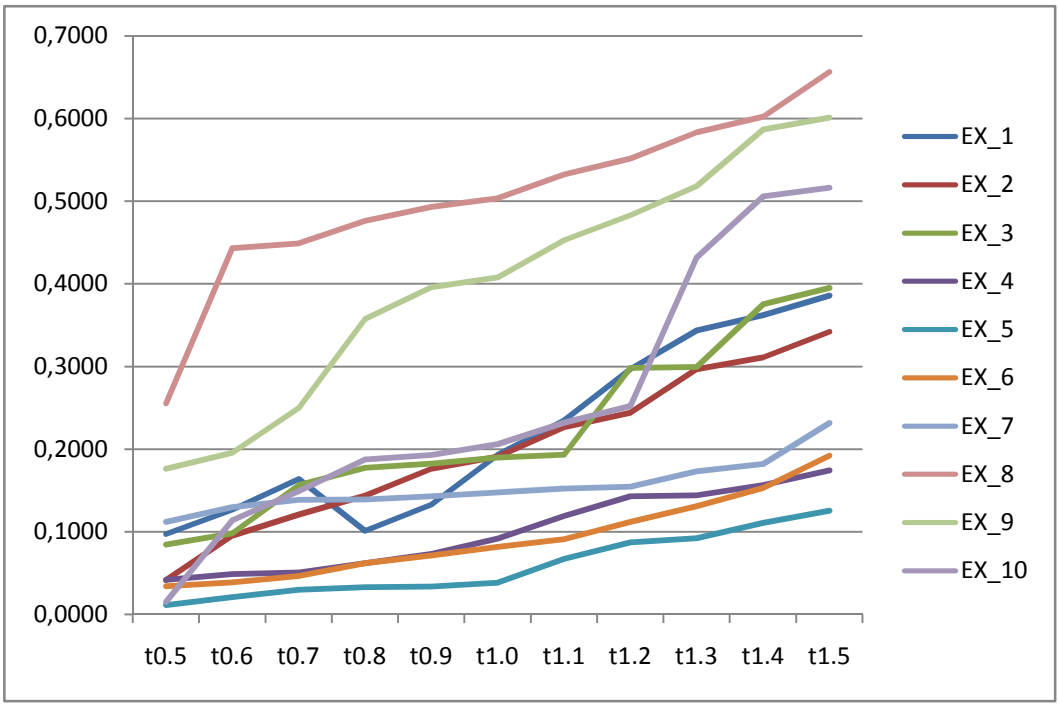


Chart (7),

Time measured for the sum methodology from EX_1 to EX_10

When we applying the robust on method we get higher timing for the computations on that specific coefficient, in generality the increasing of the time are reasonable because of removing one potential solution and make the result more naïve, time of computations raise in average, the result shown in the chart (8).

We can see that when we apply the robust method the average times are always increasing, this time increasing can show us that removing the first test coverage and make it naïve affects the time of computing and for larger coefficient the time increases or same as previous phase, so we can assume that the component test matrixes all well defined that removing one solutions leads to have longer time to find the best initial test set.

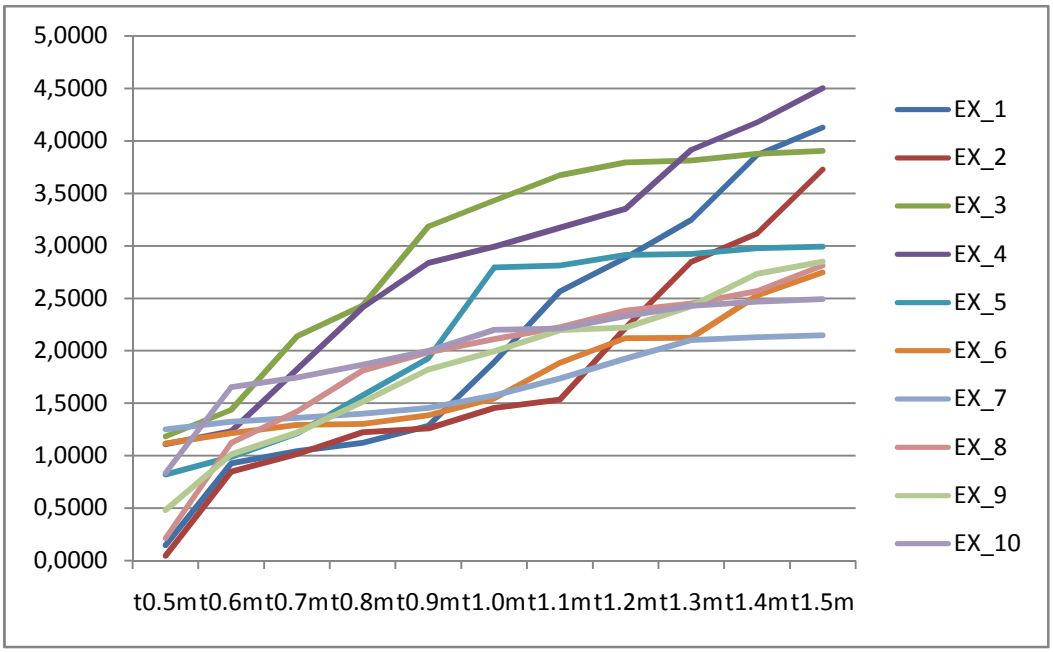


Chart (8),

Time measured for the robust sum methodology from EX_1 to EX_10

In chart (9) the measured time for the second set of matrixes are shown, because the component test matrixes dimensions is greater than the dimension of the first set of matrixes the computing time is larger.

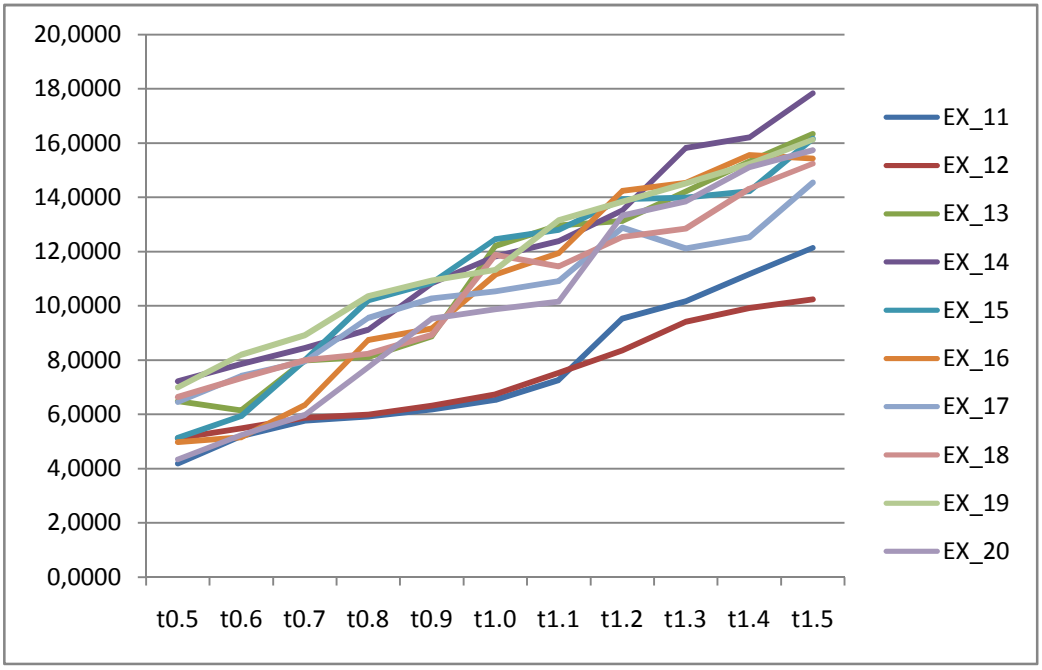


Chart (9),

Time measured for the sum methodology from EX_11 to EX_20

The time for the robust on method one is measured too, for the second set of matrixes as same as for the first set of matrixes, the computing time is greater than the computing time for first set, because the dimension of this set of matrixes are larger the results display in chart (10).

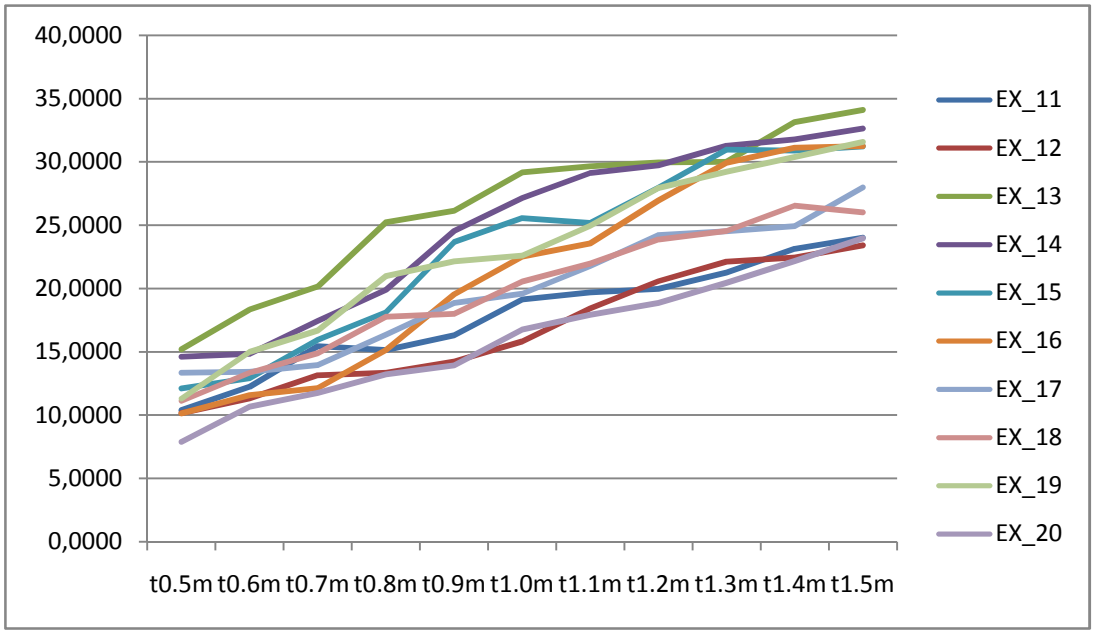


Chart (10),

Time measured for the robust sum methodology from EX_11 to EX_20

By having more precise look on the time measured charts for the both sets of matrixes we can see the same behaviors on the charts obtained from different dimension, this can assure that the results obtained are proper.

4.2 Logarithm Methodology Results

In this section the results of the Logarithm Methodology are presented and analyzed, as it was mentioned before, for computing the results by this method we apply the logarithm to the input data for the computing and then collect the results, in this method we use coefficients from 0.5 but we take away by 0.1 up to the 0.1 instead of increasing the coefficients.

We have two sets of component test matrices, first set ten matrixes with dimension of 20x10 and second set 10 matrixes with dimension of 50x20, in fact the component test matrixes are same as the matrixes we used for the sum methodology but because we assign values to the high, medium, low, and not covered wise verse than previous method the component test matrixes my look different.

The chart (11) shows us the results obtained from the computing of the first set of component test matrixes, as we can see in the chart(11) the algebraic sum of the ones, in the obtained results which indicates the number of components covered with respect to the coefficient is not always increasing and in some cases after specific number of coefficient is constant, for example the EX_10 in the chart (11) after the coefficient 0.4 is always showing same value as well as EX_5.

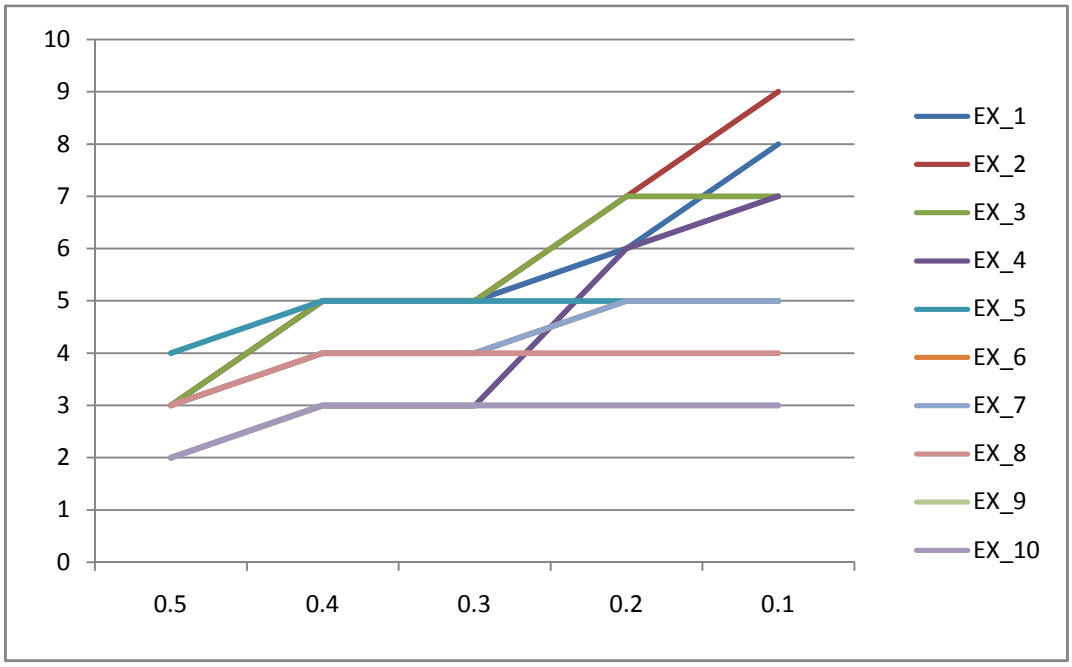


Chart (11),

Results for first ten initial test sets computed by applying the Logarithm methodology from EX_1 to EX_10

Same as previous method in this method we also have two sets of component test matrixes to assure the results by parallel computations, the chart (12) show the result obtained from the second set of component test matrixes which have the larger dimension.

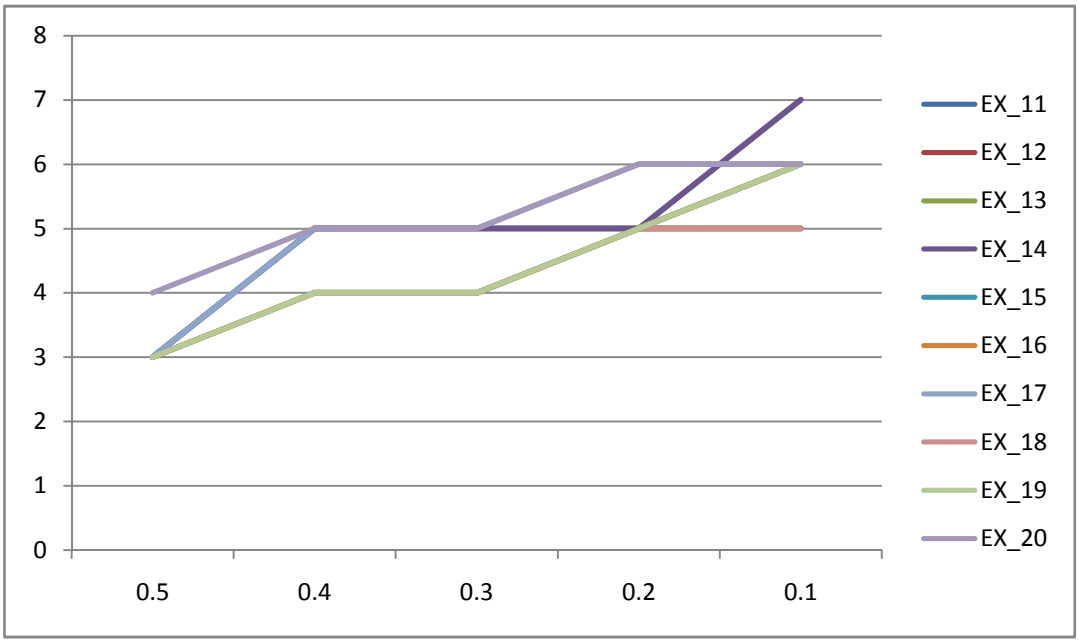


Chart (12),

Results for second ten initial test sets computed by applying the Logarithm methodology from EX_11 to EX_20

In the both charts we can see that the number of covered components are increasing or having same value, when the coefficient number is decreasing, so we have same model of behavior for two set of matrixes with different dimension.

4.2.1 Logarithm Methodology Robust Results

In this section we present the results collected from applying the robust method on the Logarithm Methodology and analyze it. Applying robust method on this phase is same as it was applied on the sum method.

The results displayed in chart (13), are obtained from applying the robust on computing the first set of ten matrixes, and the chart (14) the results of robust on the second set of ten matrixes. As we can see the general behavior of both charts all same as it is desired.

Further we will integrate the results obtained from the logarithm method and robust method applied on it, with each other, respecting the dimension of the matrixes and we write some analysis on it.

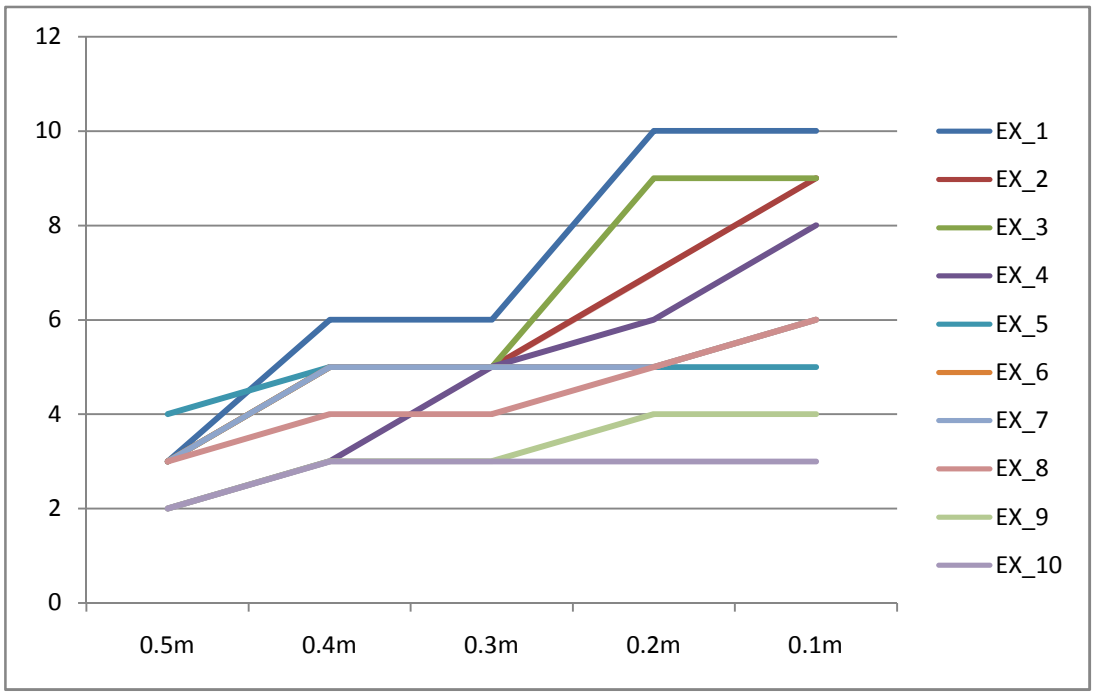


Chart (13),

Results for first ten initial test sets computed by applying the robust on Logarithm methodology from EX_1 to EX_10

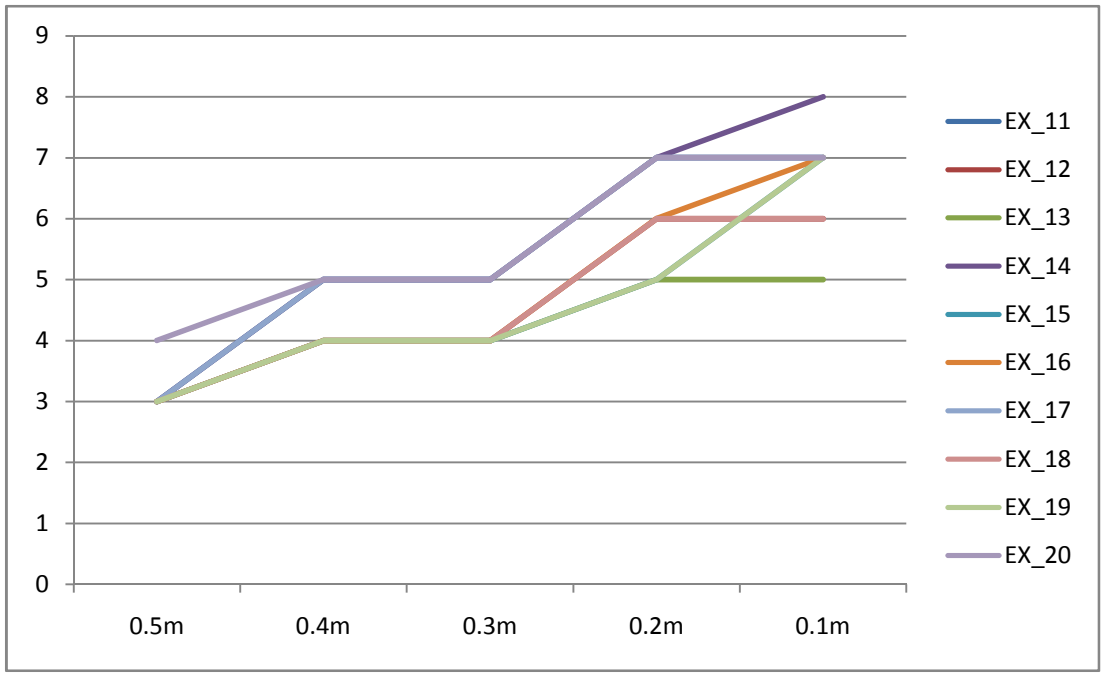


Chart (14),

Results for second ten initial test sets computed by applying the robust on Logarithm methodology from EX_11 to EX_20

4.2.2 Comparing Logarithm and Robust Results

Again to have clear idea about the differences of the sum method and the robust one we integrate the data obtained from both methods together with respect to the dimension of the component test matrixes.

Integration results obtained from logarithm method and robust of it for first set of component test matrices are shown in chart (15), integrated results for second set is shown in chart (16). The results have been integrated using the same principal for the previous method the only difference is that at this step we have coefficients from 0.5 down to 0.1.

As it was mentioned before the component test matrixes are same as the component test matrixes used in previous method so we expect the same behavior. The zigzag form of behavior appears again in this case we can see the common behavior in chart (15) and chart (16) as well.

We can see that by applying the robust method the algebraic sum of the covered components did not change with very highly difference or in the case of EX_10 did not change at all, also EX_5 did not change as well.

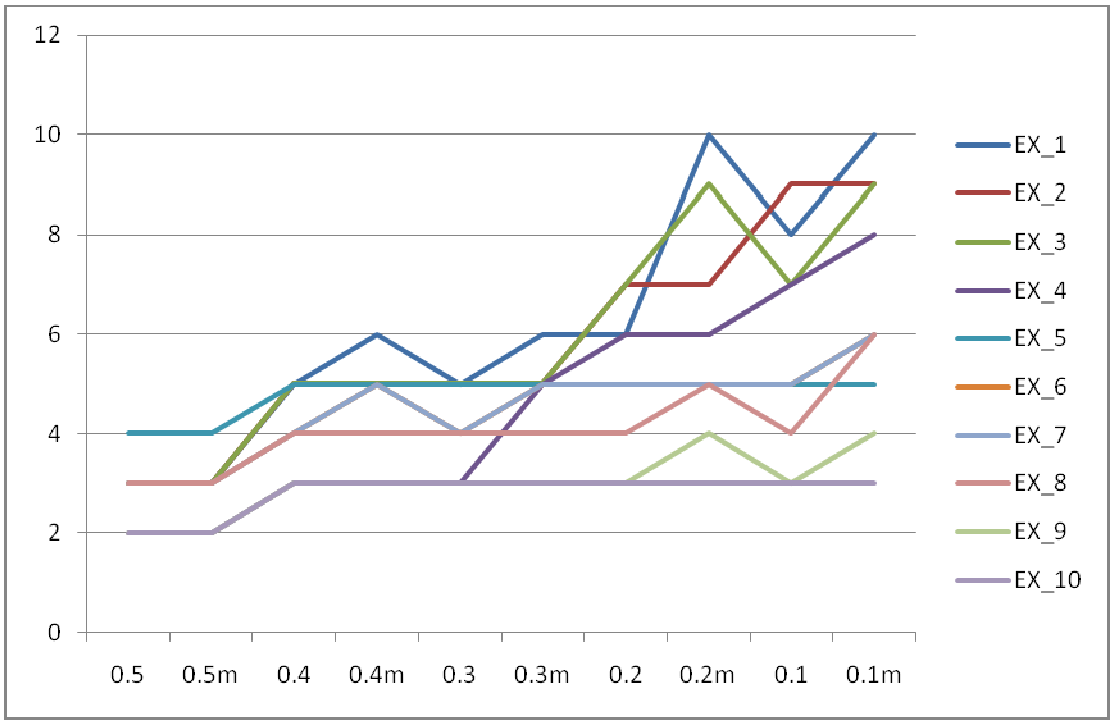


Chart (15),

Results for first ten initial test sets integrating robust and logarithm methodology from EX_1 to EX_10

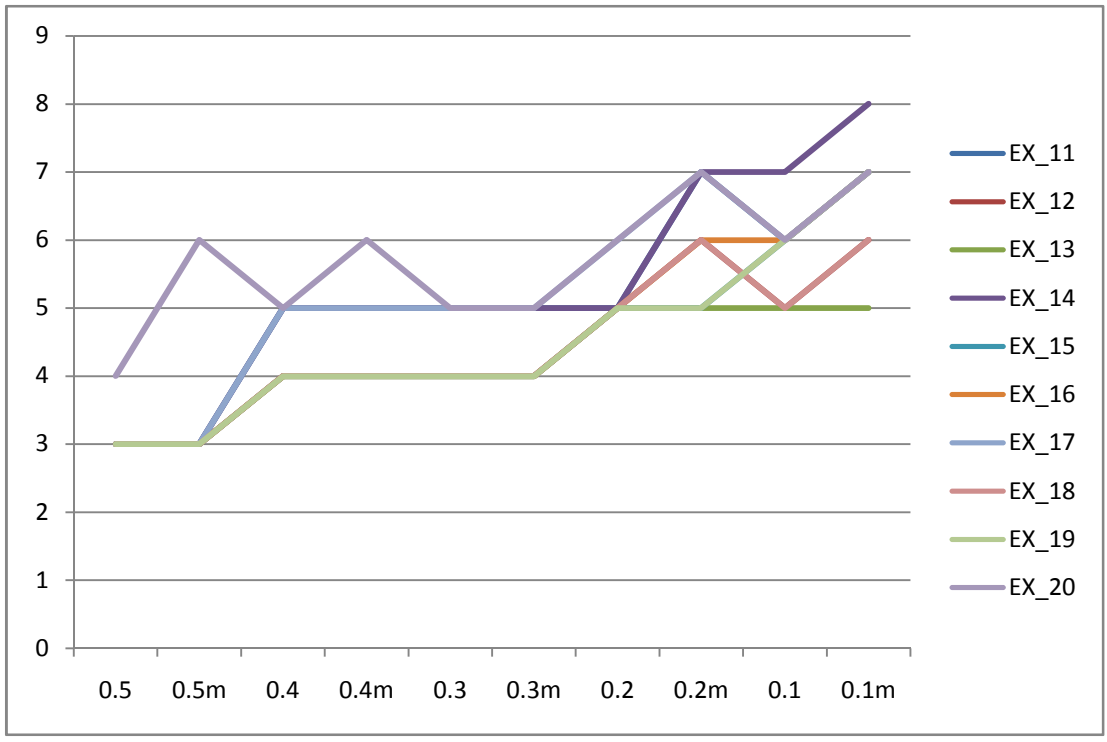


Chart (16),

Results for second part, ten initial test sets integrating robust and logarithm methodology from EX_11 to EX_20

In chart (16) we can see more drops and pick points in compare with the chart (15). There can be two potential reasons for this, first of all by having component test matrices with larger dimension because of the more number of components when we remove one potential covering set from the component test matrix it influence system more.

The second reason can be that when generating the component test matrixes, as the number of the components and tests increase the generated matrix is not as efficient as generated matrixes with lower number of components and tests.

In fact as much as the physical layer of the circuit be more complex, it reflects all phases of designing of the test strategies and test methods, the sum effect of it on our work is that we will have larger component test matrixes.

All this will lead us to a higher time of computing and as result longer time spending to find the proper result, further we are going to present the results of the logarithm method and the robust part of it in the manner of time.

4.2.3 Logarithm and Robust Timing Results

In this part the results of measuring the cpu clock of the system while computing the logarithm method and the robust part for the sum method has been measured and displayed.

Same as previous method measure time by average of twenty times of running the same algorithm for logarithm method and also when we robust it.

In chart (17), which displays the results of measuring time for the first ten set of matrixes, we can see that as it was expected from the computations the average time of computing increases according to the increase of coefficients. As we explained before this is the natural result of applying the bintprog.

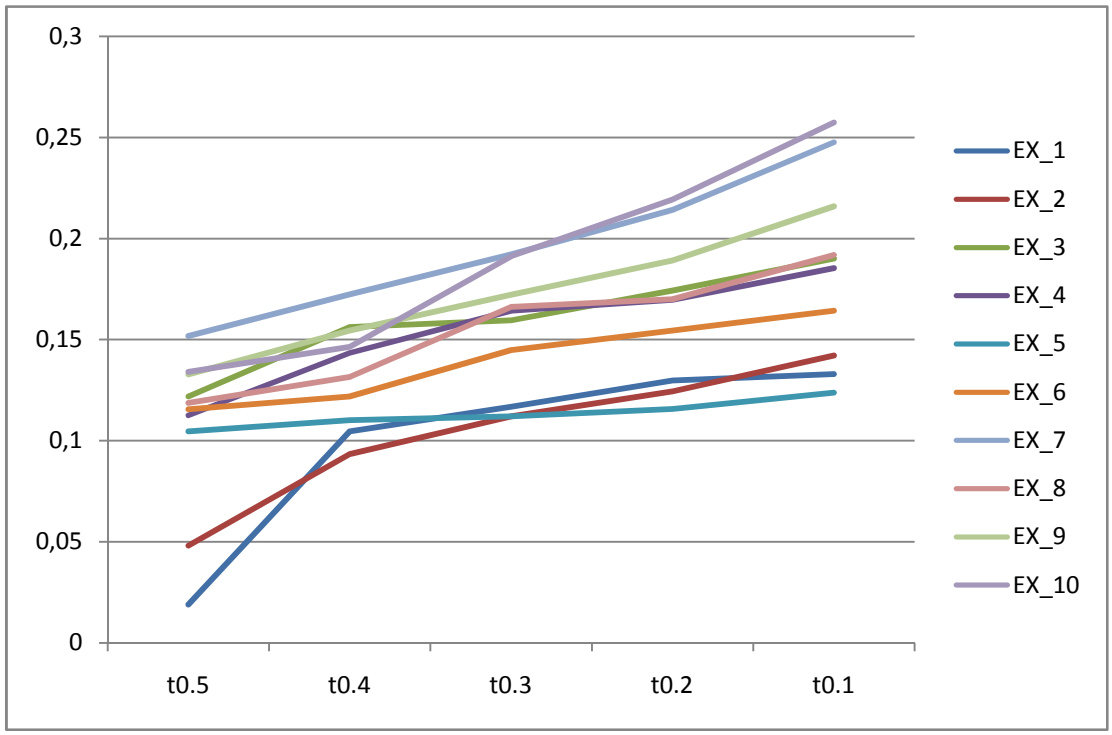


Chart (17),

Time measured for the logarithm methodology from EX_1 to EX_10

The chart(18), results of the robust on logarithm method has been showed, we get higher total timing for the computations comparing with the logarithm method.

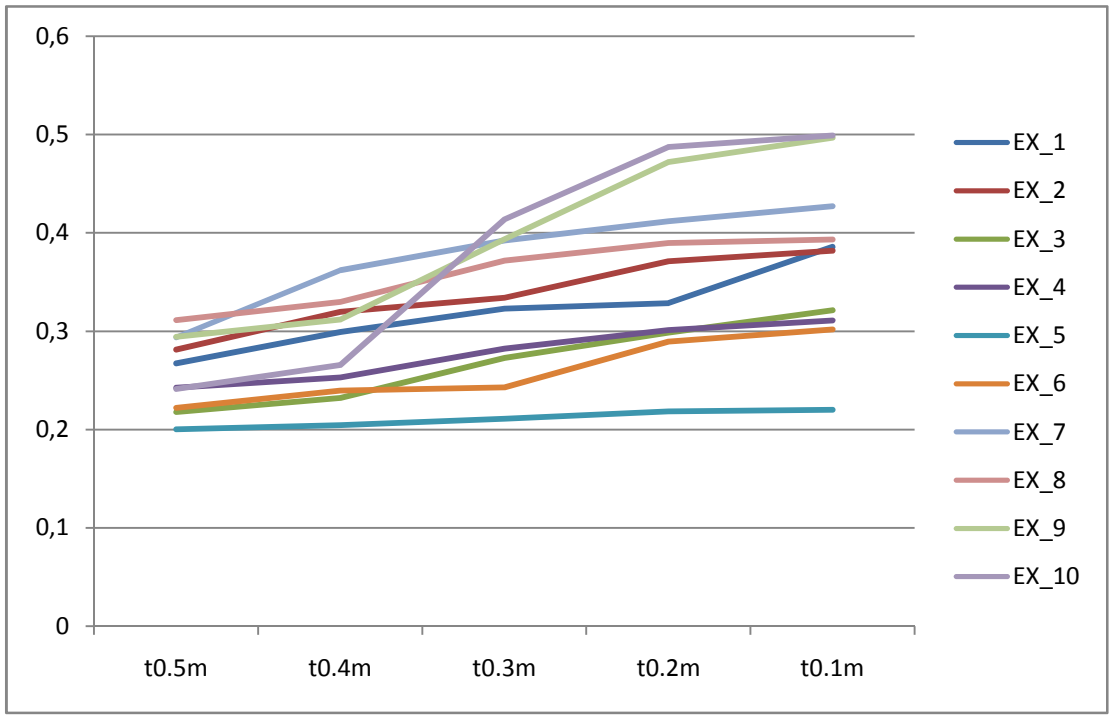


Chart (18),

Time measured for the robust logarithm, methodology from EX_1 to EX_10

The total time needed for obtaining the results shown in chart(17) and the total time for obtaining the results shown in chart (18) is less than applying the previous method on same components matrixes, but we have to consider that this is because the number of coefficients in this method is less than previous method, so if we assume that the lower time of computing is an advantage we can say that logarithm method saves some time comparing to the sum method.

In the chart (19) the results of the timing for the second ten sets are shown and the chart (20) the robust results of the second ten test sets are shown.

Same as the timing results for the first ten sets and the robust on them, the total time shown in chart(19) and the total time for obtaining the results shown in chart (20) is less than applying the previous method on same components matrixes, but as we explained before the number of coefficients in this method is less than previous method.

In the next chapter we will make the conclusion about all the works done in this paper and the future works.

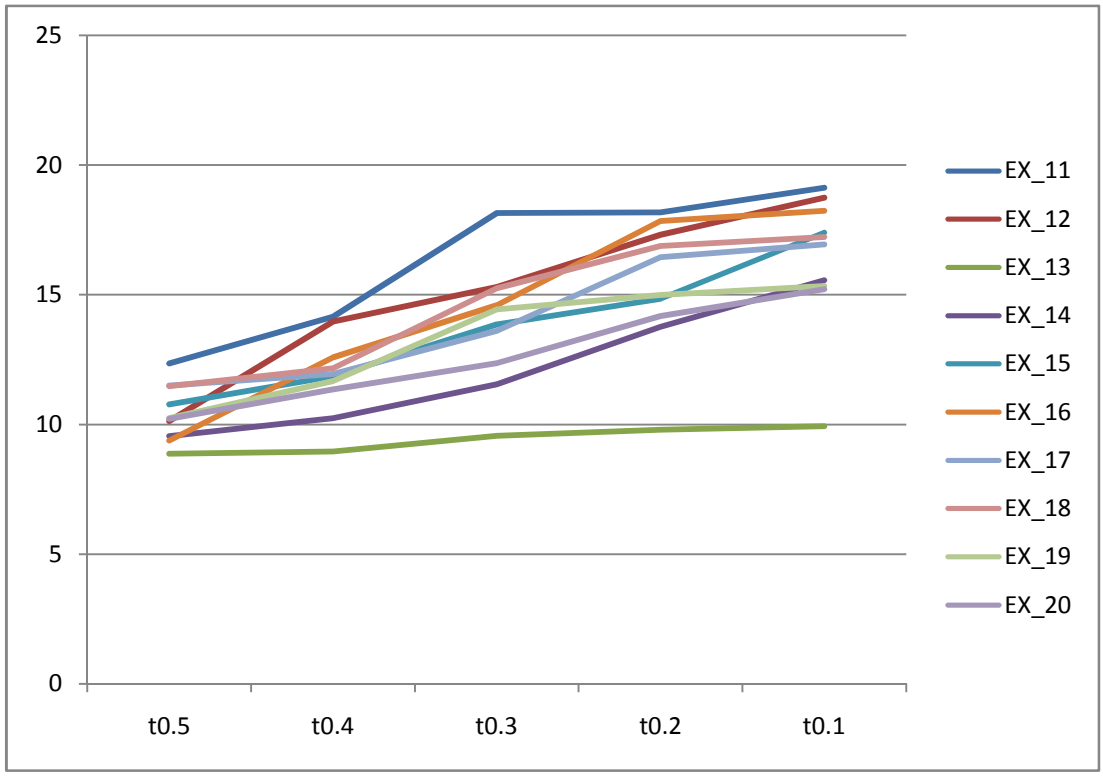


Chart (19),

Time measured for the logarithm methodology from EX_11 to EX_20

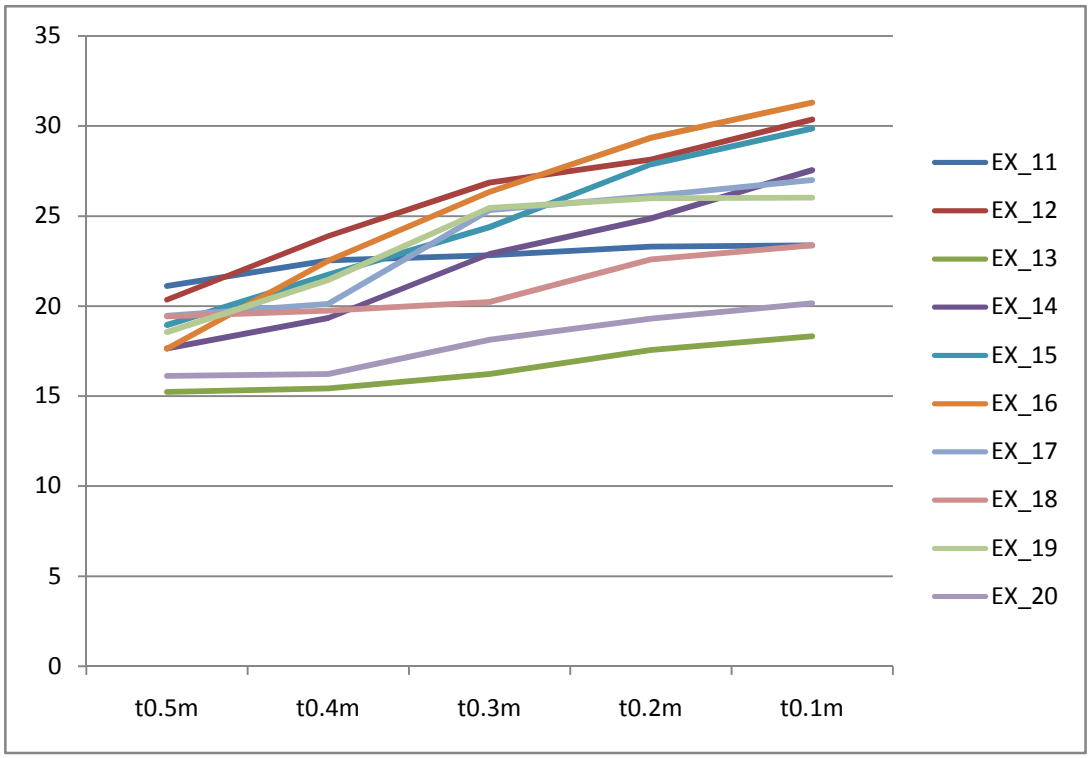


Chart (20),

Time measured for the logarithm methodology from EX_11 to EX_20

4.3 Customizing the methodologies

In this part we present the results of customizing both methods used in this research work. It is obvious that the results obtained according to the methods used above are theoretically developed. In this section by adding cost to the tests we run the tests with both of the methodologies for couple of the component test matrixes.

By comparing the results obtained from the computing methodologies and the customized version of them we can understand that if we apply the methods in the real industry, for our case CISCO company apply them as a part of diagnosing methods to find fault components it have compatibility with real electronically circuits.

4.3.1 Customizing the Sum methodology

The results shown in chart (21) obtained from applying the sum methodology on the two components test matrixes chose from the first set of the initial test sets. Chart (22) uses two components matrixes from the second set of initial test sets.

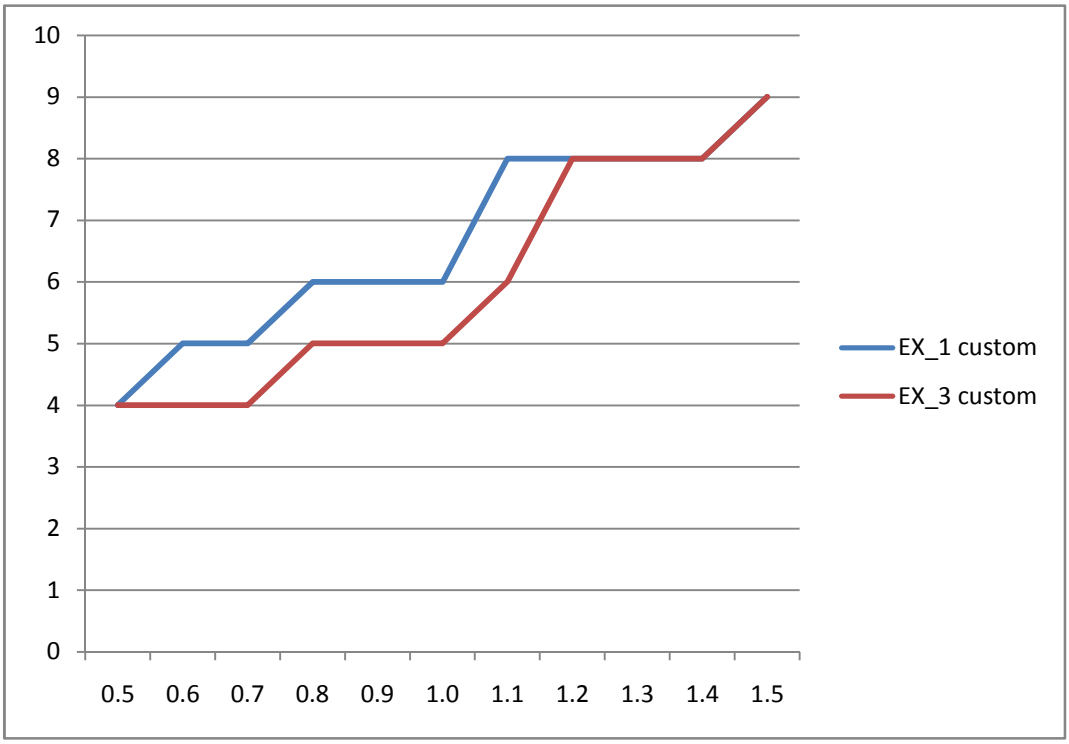


Chart (21),

Results computed by applying customized sum methodology for EX_1 and EX_3

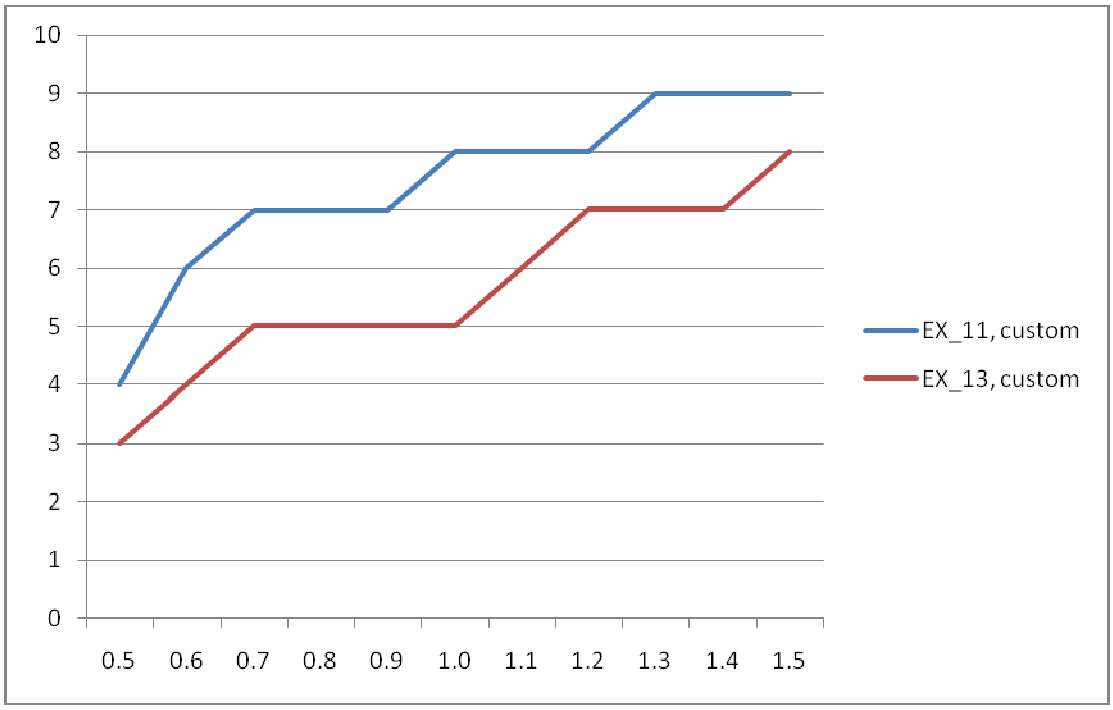


Chart (22),

Results computed by applying customized sum methodology for EX_11 and EX_13

If we compare results obtained from chart(21) with the chart (1) and chart(22) with (2) we can see the same behavior from all of them, so we can assume that by applying the custom cost to the tests we still get the same behavior.

To have more clear results we compute the value fval, which in our case is f*c. the results shown in chart (23) are computed by using the EX_1 and EX_3, and chart (24) are computed by using EX_11 and EX_13.

In general the behavior of this charts are also incremental, means that as the value of the coefficient increase we can see that the values of fval increase as well. The customized sum method is returning the expected behavior, so fval can be also one indicator that shows us results obtained are perfectly correct, and the industrial usage of the method also return the expected results.

Chose of two examples from first ten sets and two examples from second test sets are randomly but as we get same behavior from all components test matrixes in all computations done before in this research work, we can be sure that computing the

rest of the component test matrixes which we used before do not influence the results.

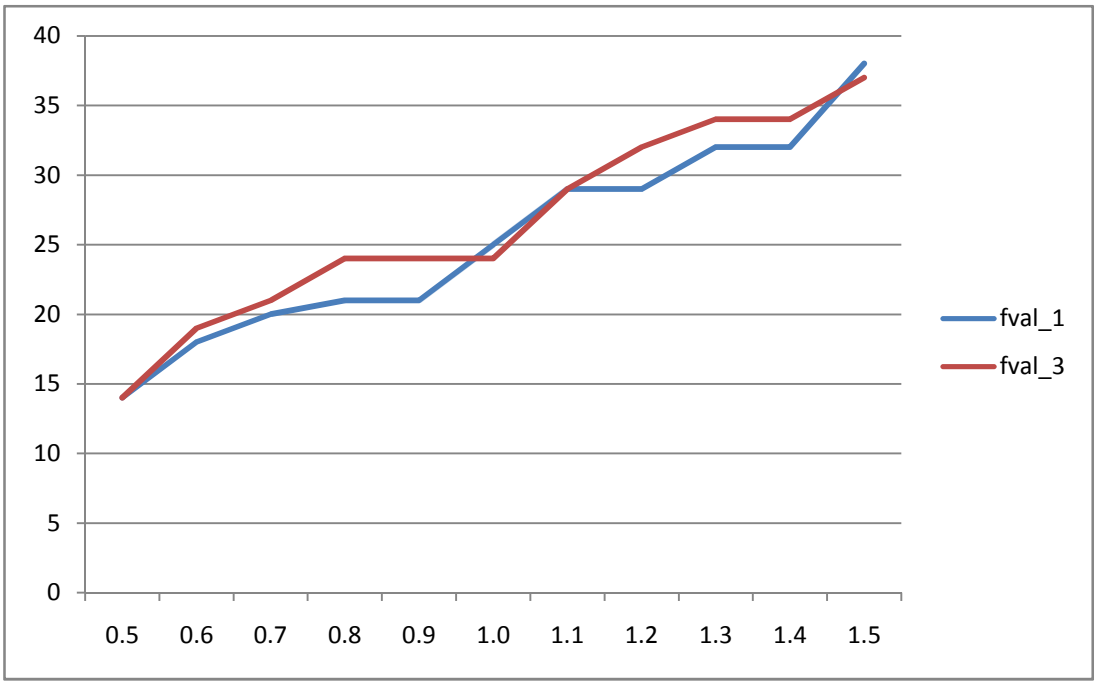


Chart (23),

Results fval by applying customized sum methodology for EX_1 and EX_3

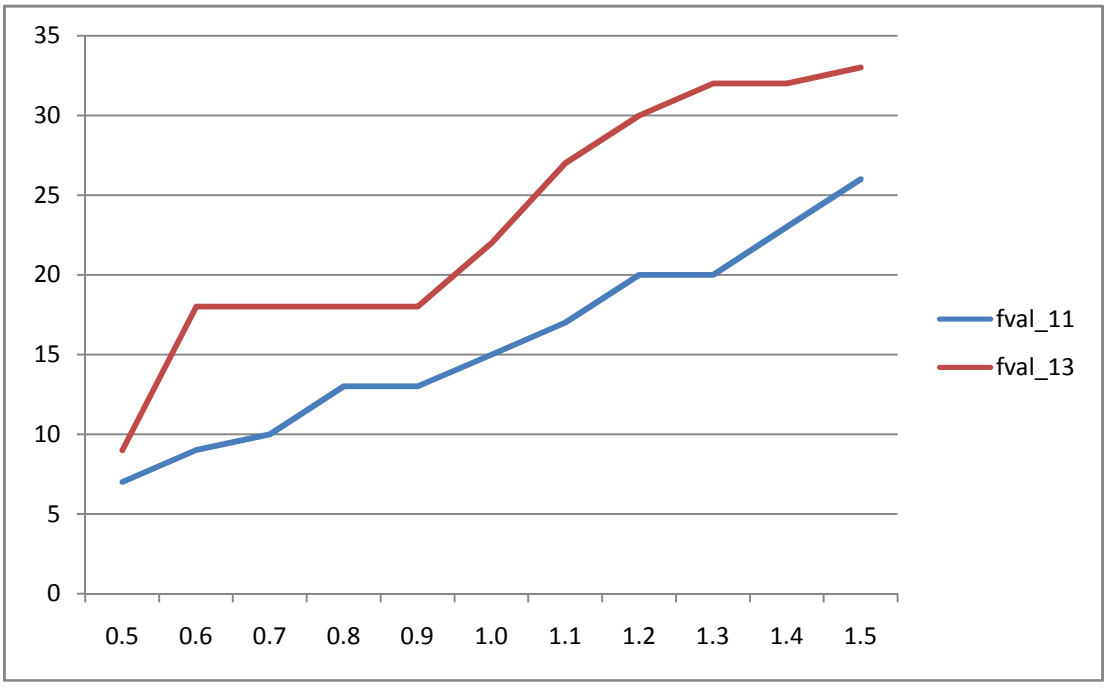


Chart (24),

Results fval by applying customized sum methodology for EX_11 and EX_13

4.3.2 Time measuring for customize Sum methodology

As the time is one of the critical indicators in this work so we compute the time also when we apply the customized methods. We use same two components test sets chose from first ten initial test sets and same two components test set chose from second ten initial test sets, which we used in previous computations of the customized method.

Chart (25) and chart (26), shows the time of the computing obtained in this part of the computation work. The time measured by average of twenty times running the computation.

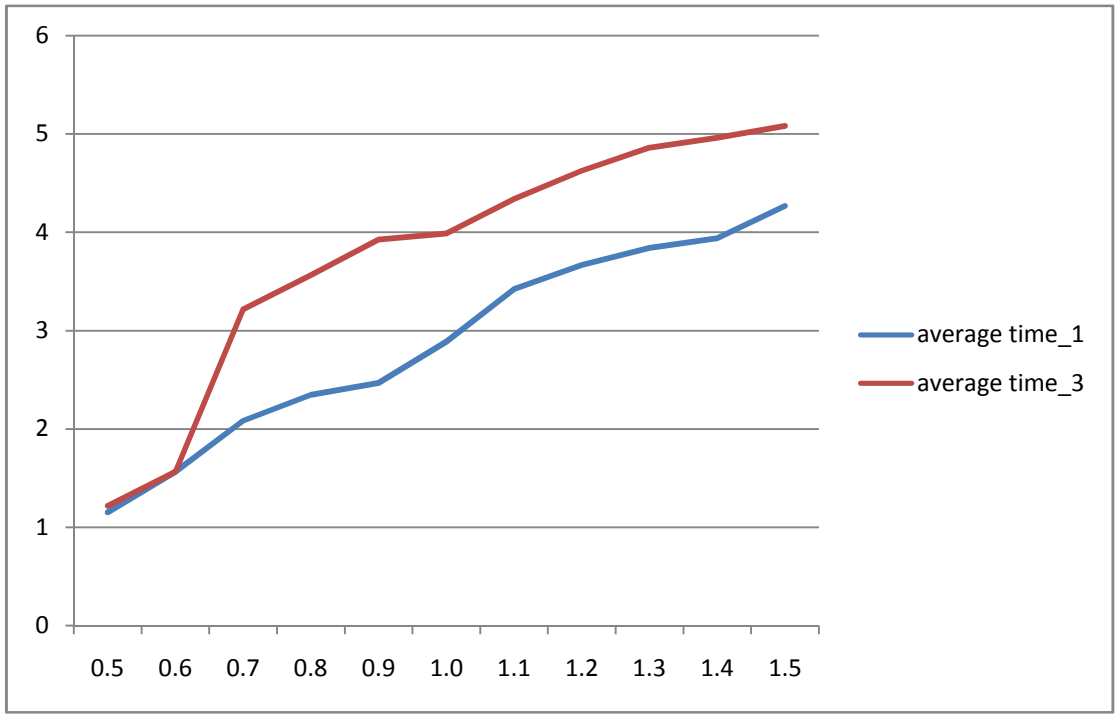


Chart (25),

Timing results by applying customized sum methodology for EX_1 and EX_3

As it is possible to see the behaviors of the time charts obtained by the customized sum method and sum method are same, the time increase by the number of coefficients increase.

We can see that also by applying the cost the time do not change significantly so the cost do not affect the timing that much that it influence the results.

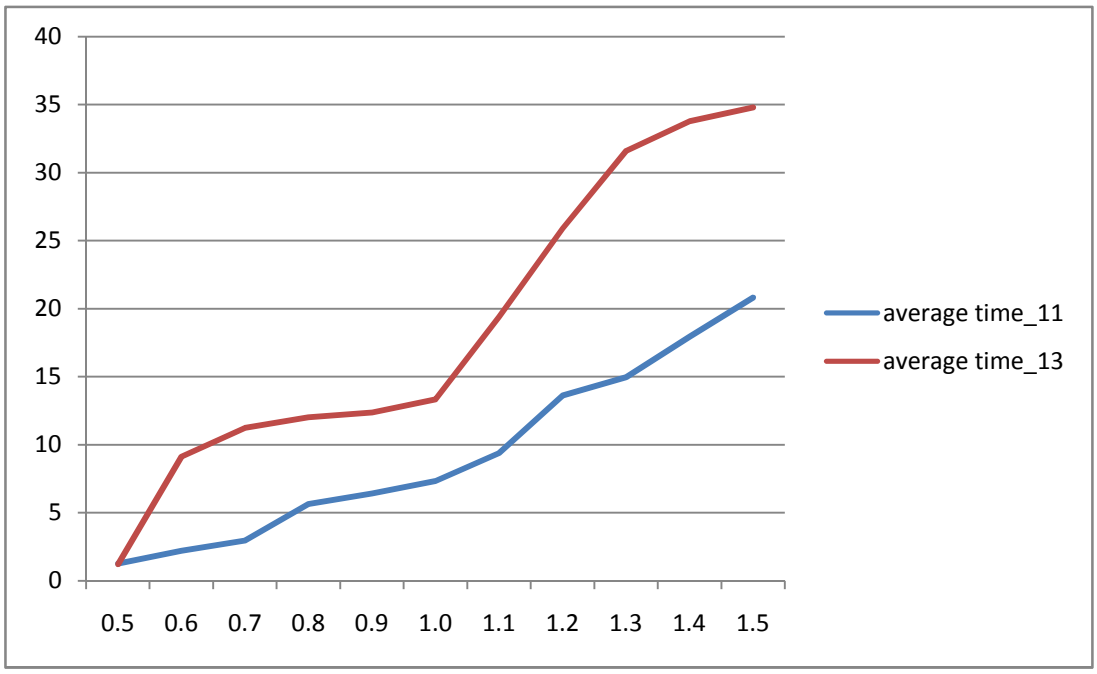


Chart (26),

Timing results by applying customized sum methodology for EX_11 and EX_13

4.3.3 Customizing the Logarithm methodology

Same as the customization of the sum methodology we apply the customization on logarithm method by assigning the cost to the computations, we use the two component test matrixes form first ten initial test sets and two component test matrixes from the second ten initial test sets, same which where use above for the computations in this section.

Charts (27) and chart (28), shows the sum of binary results obtained from applying the customization on the logarithm method, as we can see the behavior of the charts are same as the chart (11) and chart (12).

As we mentioned before same behavior was expected and we got the results which we were expecting, the number of components are increasing by decreasing the number of the coefficient in this case.

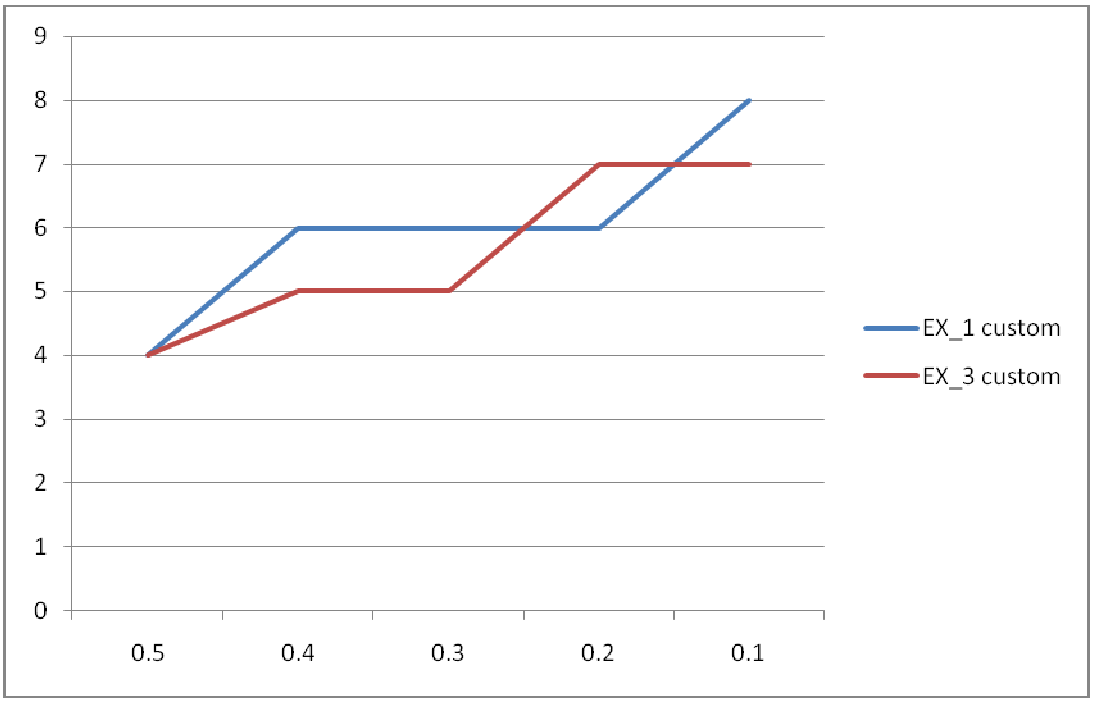


Chart (27),

Results computed by applying customized logarithm methodology for EX_1 and EX_3

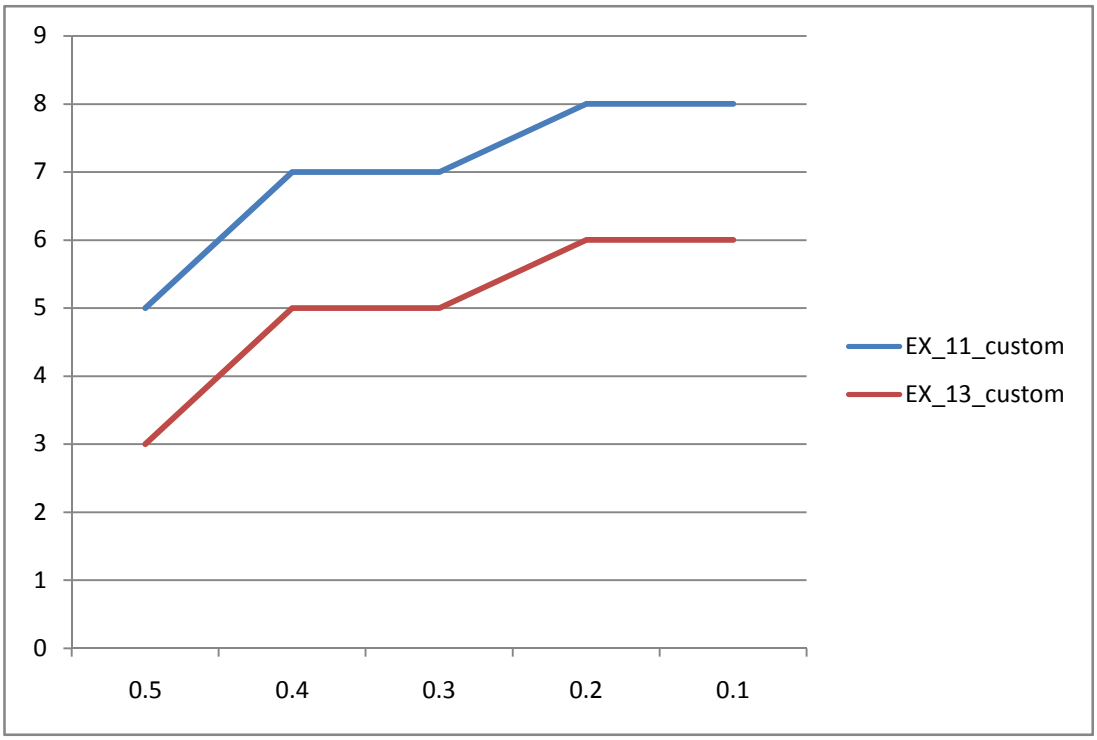


Chart (28),

Results computed by applying customized logarithm methodology for EX_11 and EX_13

Same as above we compute the value fval, which in our case is f*c. the results shown in chart (29) are computed by using the EX_1 and EX_3, and chart (30) are computed by using EX_11 and EX_13.

As it should be the behavior of this charts are incremental, means that as the value of the coefficient in this case decrease the values of fval increase. The customized logarithm method is returning the expected behavior as well as sum method, in fact as we mentioned before fval can be one indicator which can be used in real industrial cases.

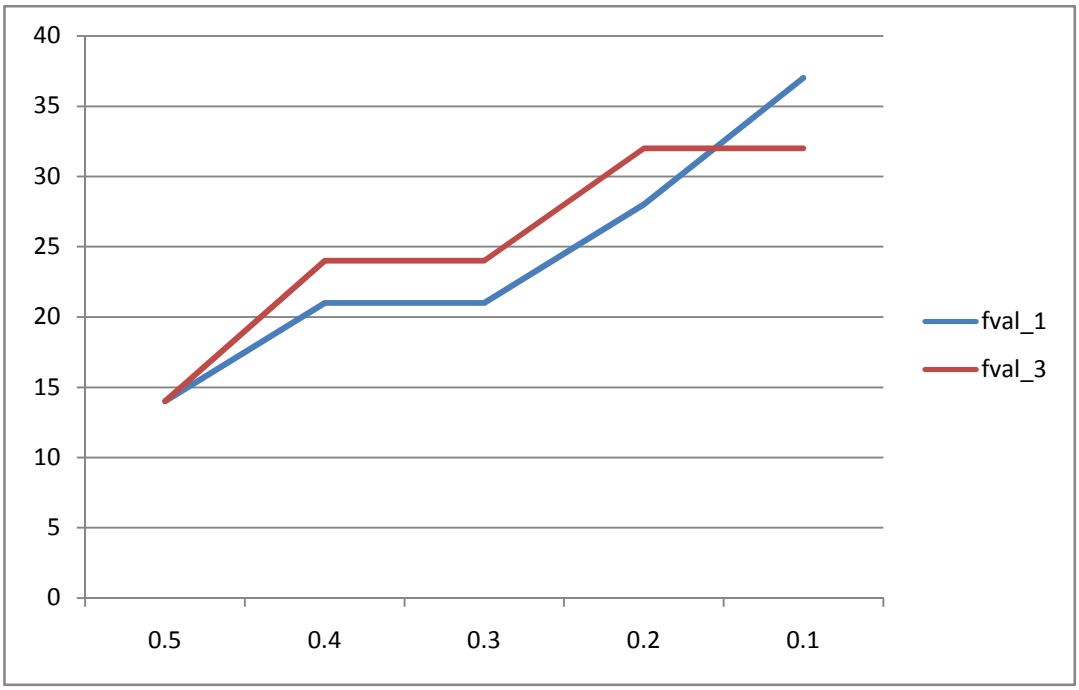


Chart (29),

Results fval by applying customized logarithm methodology for EX_1 and EX_3

The more precise look on the chart (29) and chart (30), the differences are that the number of fval when we have the larger dimension matrixes, are larger.

But the importance for us is the behavior of the charts, the incremental behavior are the expected, and both set of component test matrixes satisfy the expected result.

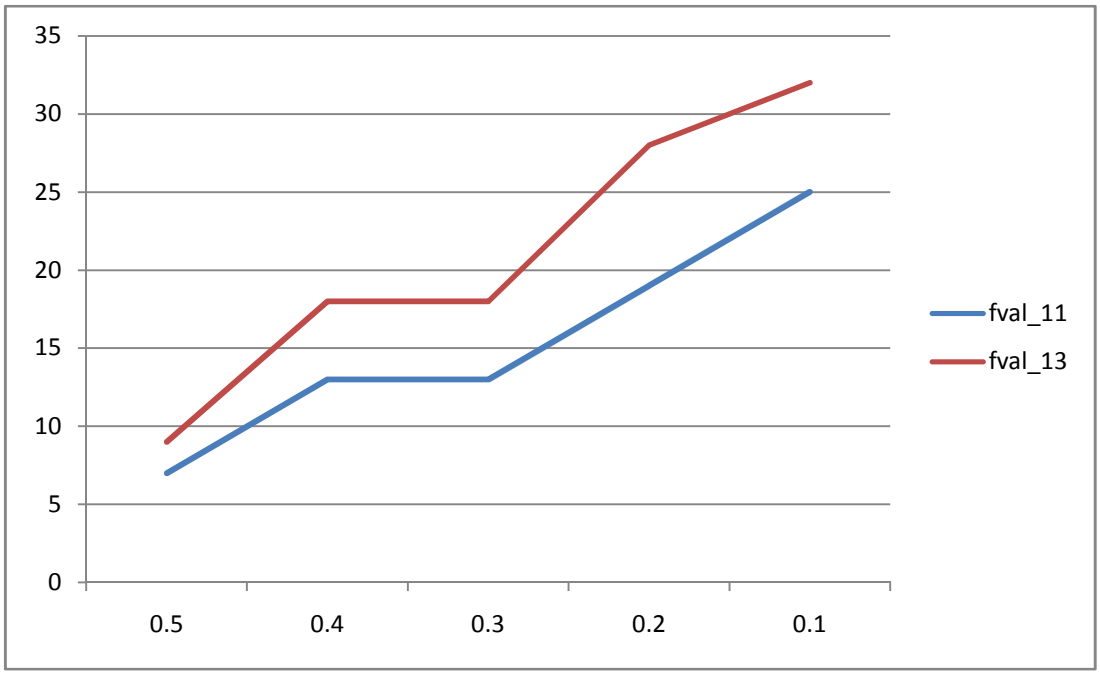


Chart (30),

Results fval by applying customized logarithm methodology for EX_11 and EX_13

4.3.4 Time measuring for customize logarithm methodology

The time indicators have been computed when we apply the customized logarithm method. We use same two components test sets, which we used in previous computations of the customized method.

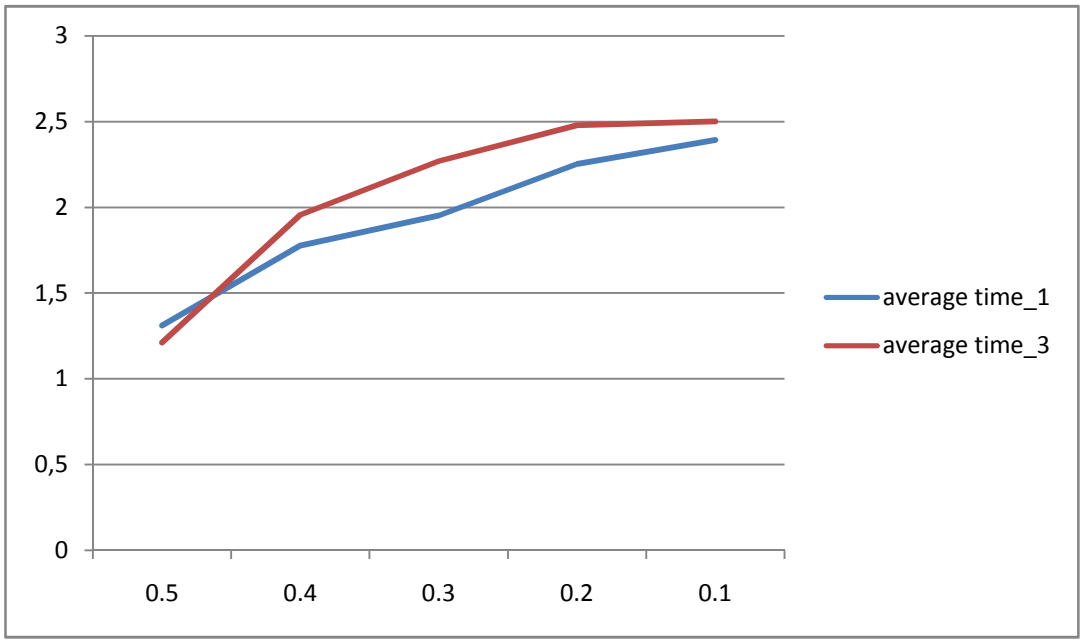


Chart (31),

Timing results by applying customized logarithm methodology for EX_1 and EX_3

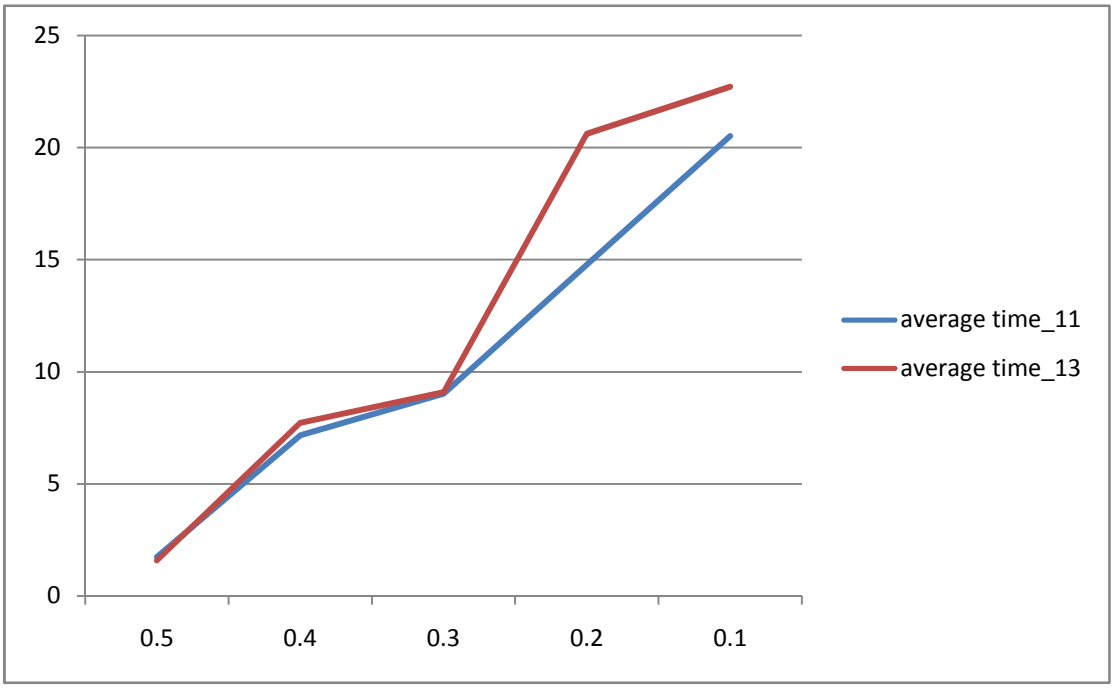


Chart (32),

Timing results by applying customized logarithm methodology for EX_11 and EX_13

Chart (31) and chart(32) displays the measured time for the customized logarithm method.

Chapter 5: Conclusion

The Sum method that we introduced and run the computations based on it return us the set of selected test sets, the results obtained from this method are as we were expecting.

As the value of threshold get larger the number of selected tests are more, so we can see an incremental behavior on the selected tests, also the time increase as well. The robustness on this method shows us that removing one solution from the initial test set do not influence the behavior of the results so we can assume that the test sets are well defined and our method is working properly.

Logarithm methodology as it was explained work with vice versa input comparing with the Sum method and then we apply the logarithm as it was explained before, so the expected behavior of the results should be same as Sum method, by having a survey on results we can see that in fact we obtain such behavior, the results have incremental behavior in this case too.

Applying robustness on the Logarithm method also returns us sustainable results with the same behavior as Sum method. The time in this method also has incremental behavior in cases, directly applying logarithm method and robustness on it. So this method is working well, and the results are acceptable and adequate.

The customization which we applied on both methods as it was explained above, we assign cost to the tests, the customization on both methods, Sum and Logarithm returns same behavior as the results obtained from applying directly both methods, this leads us to the conclusion that the methods are working properly.

By considering all the results obtained from the methods introduced, we can say that the goal of this project which was finding the best initial test set, for beginning of the tests on electronic boards to find the guilty component has been satisfied. Hopefully the outcome of this work can improve the quality of testing and saving time and cost to find the guilty components on complex systems.

Future works

The future works conducting our thesis, finding more efficient methods to define the initial test sets or solving the set covering problem with new methods which can work faster.

Further steps on this project can be working on finding an automatic strategy to identify guilty components on complex systems, after defining the test set for beginning the tests the test engineers need and strong strategy to find the exact guilty component or components.

After all same as all automatic processes we need to give an stop point to the system, means that when it is necessary to stop testing and by sure all guilty components are identified, so defining an stop point on running the tests can be one of the future works to our work.

References

1- www.**ieee**.org

2- Fault Diagnosis of Electronic Systems Using Intelligent Techniques: A Review. William G. Fenton, Member, IEEE, T. M. McGinnity, Member, IEEE, and Liam P. Maguire

3- Soft Computing Approaches to Fault Diagnosis for Dynamic Systems: A Survey. R J Patton, F J Uppal & C J Lopez-Toribio.Control and Intelligent Systems Engineering, Faculty of Engineering and Mathematics, The University of Hull, Cottingham Road, Hull Hu6 7RX, United Kingdom.

4- http://www.gnu.org/software/glpk/

5- http://www-01.ibm.com/software/integration/optimization/cplex/

6- http://www.mathworks.com/

7- http://www.microsoft.com

8- A better-than-greedy approximation algorithm for the minimum set cover problem Refael Hassin1 and Asaf Levin2 February 3, 2005

9- http://en.wikipedia.org/wiki/Scalability

10- http://en.wikipedia.org/wiki/Computational_complexity_theory

11- Rampone S. (1998), "Recognition of Splice-Junctions on DNA Sequences by BRAIN learning algorithm", Bioinformatics Journal, 14(8), 676-684.

12- Probability-driven Greedy Algorithms for Set Cover, Salvatore Rampone

13- Greedy Approximations: Set Cover and Min Makespan, Date: 1/30/06.

14- A Generalization of the Weighted Set Covering Problem Jian Yang,1 Joseph Y-T. Leung2, Department of Industrial and Manufacturing Engineering, New Jersey Institute of Technology, Newark, New Jersey 07102.

15- A Greedy Heuristic for the Set-Covering Problem. V. Chvatal Mathematics of Operations Research, Vol. 4, No. 3 (Aug., 1979), pp. 233-235 Published by: INFORMS

16- R. Bar-Yehuda and S. Even, "A linear time approximation algorithm for the weighted vertex cover problem," Journal of Algorithms, 2, 1981.

17- A Tight Analysis of the Greedy Algorithm for Set Cover, Journal of Algorithms, Slavìk P. (1997), 25(9), 237-254.

18- A Threshold of ln n for Approximating Set Cover. In J. ACM 45(4), Uriel Feige. pp 634-652. (1998).

19- A sub-constant error-probability low-degree test, and sub-constant error probability PCP characterization of NP", R. Raz and S. Safra, Proc. STOC 1997, 475- 484, 1997.

20- Algorithms for set covering problem, Alberto Caprara, Matteo Fischetti, Paolo Toth. University of Bologna, University of Padova, Italy.

21- A Lagrangian relaxation based heuristic for the consolidation problem of airfreight forwarders. Kuancheng Huang a, Wenhou Chi b,Department of

Transportation Technology and Management, National Chiao Tung University, accepted 28 August 2006
22- Algorithms for Weighted Boolean Optimization. Vasco Manquinho1, Joao Marques-Silva2, Jordi Planes, University College Dublin, Universitat de Lleida, March 6, 2009
23- Holland J. Adaptation in natural and artificial systems. Cambridge: University of Michigan, MIT Press, 1992.
24- Goldberg DE. Genetic algorithms in search, optimization & machine learning. Publishing Company, USA: Addison-Wesley, 1989.
25- A parallel genetic algorithm to solve the set-covering problem, Mauricio Solar, VmHctor Parada, Rodrigo Urrutia Departamento de Ingenierna& Informatica, Universidad de Santiago de Chile, Av. Ecuador 3659, Santiago, Chile
26- An Indirect Genetic Algorithm for Set Covering Problems. *Journal of the Operational Research Society*, **53** (10): 1118-1126, Dr Uwe Aickelin, School of Computer Science, University of Nottingham, NG8 1BB UK,
27- Genetic Algorithms, Numerical Optimization, and Constraints. Zbigniew Michalewicz, Department of Computer Science, University of North Carolina, Charlotte, NC 28223
28- Genetic Algorithm and Direct Search ,Toolbox, MATLAB user guide.
29- Multiproject Scheduling with Limited Resources: A Zero–One Programming Approach,'' *Management Science* 1969, by A. A. B. Pritsken, L. J. Watters, and P. M. Wolfe.
30- An Incremental approach to functional diagnosis. F.Salice, L.Amati, Departmet of electrical and informatics politecnico di Milano.