FOURIER ANALYSIS BASED TESTING OF FINITE STATE MACHINES

A Dissertation Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

by
Savaş TAKAN

March 2019
İZMİR

We approve the thesis of **Savaş TAKAN**

Examining Committee Members:

Assoc. Prof. Tolga AYAV
Department of Computer Engineering
İzmir Institute of Technology

Assoc. Prof. Tuğkan TUĞLULAR
Department of Computer Engineering
İzmir Institute of Technology

Asst. Prof. Dr. Şebnem BORA
Department of Computer Engineering
Ege University

Asst. Prof. Dr. Selma TEKİR
Department of Computer Engineering
İzmir Institute of Technology

Asst. Prof. Dr. Mutlu BEYAZIT
Department of Computer Engineering
Yaşar University

25 March 2019

Assoc. Prof. Dr. Tolga AYAV
Supervisor, Department of Computer Engineering
İzmir Institute of Technology

Assoc. Prof. Dr. Tolga AYAV
Head of the Department of
Computer Engineering

Prof. Dr. Aysun SOFUOĞLU
Dean of the Graduate School of
Engineering and Sciences

To Duygu

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Tolga AYAV, who helped me during my doctoral dissertation study, for his interest and knowledge.

Also, I would like to thank Dr. Tuğkan TUĞLULAR, Dr. Şebnem BORA, Dr. Selma TEKİR and Dr. Mutlu BEYAZIT for their valuable contributions to this work.

Finally, I would like to express my gratitude to my dear mother, father, sister and my wife, Duygu ERGÜN TAKAN, for their support, patience and dedication until the completion of this work.

ABSTRACT

FOURIER ANALYSIS BASED TESTING OF FINITE STATE MACHINES

Finite state machine (FSM) is a widely used modeling technique for circuit and software testing. FSM testing is a well-studied topic in the literature and there are several test case generation methods such as W, Wp, UIO, UIOv, DS, HSI and H. Despite the existing methods, there is still a need for alternative techniques with better performance in terms of test suite size, fault detection ratio and test generation time. In this thesis, two new test case generation methods, F and F_w have been proposed. The proposed test generation methods are based on Fourier analysis of Boolean functions. Fourier transformations have been studied extensively in mathematics, computer science and engineering. The proposed F method only tests outputs whereas F_w method also tests the next state with the outputs. In this context, the proposed methods are compared with UIO and W methods in terms of *characteristic*, *cost*, *fault detection ratio* and *effectiveness*. The evaluation data are analyzed using T-Test and Hedges' g. Results show that F and F_w methods outperform the existing methods in terms of the fault detection ratio per test.

ÖZET

SONLU DURUM MAKİNELERİNİN FOURIER ANALİZİ TABANLI TESTİ

Sonlu durum makinesi (FSM), devre ve yazılım testlerinde yaygın kullanıma sahip bir modelleme tekniğidir. FSM testi iyi çalışılmış bir konudur ve literatürde W, Wp, UIO, UIOv, DS, HSI ve H gibi test üretim yöntemleri vardır. FSM'lerin testi için literatürde çeşitli yöntemler bulunmakla birlikte, modellerin büyümesi sonucu test kümesinin büyüklüğü, hata yakalama oranı ve test üretim süresi gibi konularda yüksek başarıma sahip alternatif test yöntemlerine ihtiyaç duyulmaktadır. Bu çalışmada, F ve F_w isimli iki yeni test üretme yöntemi önerilmiştir. Önerilen yöntemler diğer test üretme yöntemlerinden farklı olarak, Boolean fonksiyonlarının Fourier analizine dayanmaktadır. Boolean fonksiyonlarının Fourier dönüşümü matematik, bilgisayar bilimi ve mühendislik alanlarında yoğun olarak incelenmiştir. F yöntemi yalnızca çıktıları test ederken; F_w yöntemi, çıktılar ile birlikte sonraki durumu da test etmektedir. Bu bağlamda, önerilen yöntemler, *karakteristik*, *maliyet*, *hata yakalama oranı* ve *başarım* bakımından UIO ve W metotları ile karşılaştırılmıştır. Elde edilen sonuçlar, T-Test ve Hedges' g ile analiz edilmiştir. Sonuçlar önerilen F ve F_w yöntemlerinin mevcut yöntemlerden daha başarılı olduğunu göstermiştir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

The development of tests using formal methods is one of the important issues for software testing [40] [18]. These kinds of tests are used in many areas, from user interface design to aircraft software and real-time systems [10]. Tests using formal methods show whether the model meets the requirements. In the field of software, models can be expressed in many ways (Finite state machines (FSM), Petri nets, UML, etc.). FSM was chosen as the technique in this project. There are various ways of creating test suites using FSMs. The most common of these are Transition Tour [44], W [11], Wp [22], UIO [57], UIOv[68], DS [26], HSI[52] [51] [70] and H [16] [15]. In this project, two new methods, F and F_w, have been developed using Fourier analysis of Boolean functions.

The Transition Tour performs an output test. With the exception of the Transition Tour, most of the above methodologies perform (next) state verification and the output test. The most common methods used in (next) state verification are input and output sequence, distinguishing sequence and characterization sequence. However, these methods have some deficiencies. For example, the problem is that some methods such as Unique input-output sequence and Distinguishing sequence may not be found in the FSM. Also, most of them produce too many test cases which is the main problem of the methods.

Figure 1.1 shows the creation of tests in methods using (next) state verification. As shown in this graph, the transfer sequence, reset sequence, and the state transition under the test are the same for all methods using state verification. The basic difference takes place in the state verification sequence. Therefore, the proposed methods and other methods such as W and UIO was compared in the state verification sequence.

When creating a test, there are two important situations. The first is the control of the outputs. The second is the control of the next state. The reason that two different methods are proposed in this study is to develop methods for both cases. From this point on, two types of methods have been identified to create the test. The first method (F method) performs only the output test, such as the transition tour. In the other recommended method (F_w method), the output test and the state verification test are used together. In this study, the main focus of both methods is output verification. For this

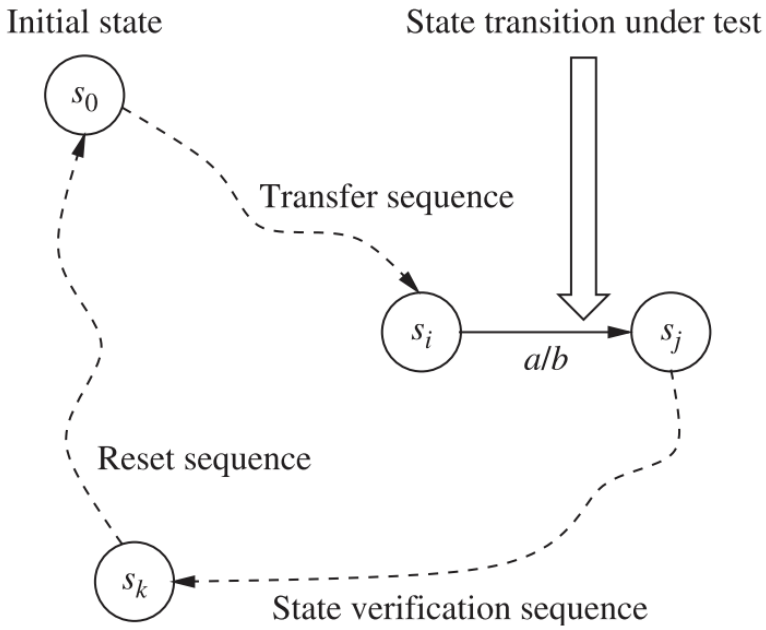

Figure 1.1. Test Case With State Verification

purpose, the transitions that affect the output most were selected. This allowed shorter test cases to be created. Thus less time and place were used.

The contribution of this study to the literature is to prioritize the output tests in FSM according to the most influential transitions and to identify potentially problematic pathways. This can be similar to the fact that the braking system and the door handle of a vehicle are not of the same importance for passenger safety. Similarly, the purpose of this proposed method is not to capture all errors, but to detect and control the points that are very likely to cause errors. Thus, without a serious drop in the rate of error detection, the test is aimed at reducing the amount of the suite. This will save time and space.

1.1. Goal and Objectives

W and UIO are among the most commonly used methods [42]. In this project, W and UIO methods are implemented. The general feature of these methods is that they scan all transitions and states (output and state verification). These methods do not use any priority calculation when scanning them. However, while some of the paths are very important, some of the paths are not important in terms of fault detection capability.

Based on this situation, it was considered appropriate to prioritize the state and transition. Thus, in this study, a priority calculation was used in terms of states and transitions. When selecting roads, instead of focusing on all states and transitions, the most used ones are detected. By determining the most commonly used routes, it is aimed to obtain more high performance tests by using less power.

The technique uses the Fourier transform on binary functions. Fourier transformation is studied in a wide range of fields in mathematics, computer science and engineering [48]. Each coefficient in the Fourier transform shows the effect on the function of the variations of the variables [64]. In the project, the variables with high effect in the function are chosen. The transitions were selected by looking at the changes in the variables. Then, the test suite was obtained with the greedy algorithm which combines transitions. The new method was compared with earlier methods by mutation analysis.

1.2. Organization of Thesis

In the chapter 2, the studies on the subject from the past to the present are mentioned. In Chapter 3, the subject is deepened with the definitions and research about Testing Finite State Machines. In this context, first of all necessary information about Finite States Machines is given. Subsequently, Testing Methods are described. In Chapter 4, Fourier Expansion and F-method are explained. In Chapter 5, there are graphical and statistical evaluations. In Chapter 7, the final results are discussed.

CHAPTER 2

RELATED WORK

A lot of research has been done in recent years about FSM testing [5] [18] [19] [2] [21] [12]. There are many surveys among the mentioned research [40] [37] [30] [14] [18]. The most common research topic of FSM testing is choosing test suites. The most important point of choosing test suites is the conformance between specification and application. The formal definition of this problem, called the conformance test, is found in many articles [58] [9].

Test generation methods are the main topic of the conformance test. Test generation methods aim to find all the errors in the given domain. Some methods require a reduced specification; others are valid only for deterministic models. However, the and/or model properties are typically used to effectively implement methods to find errors in a particular domain. In this way, it is possible to make assumptions about errors. Thus, it can be proven that all the faults in a domain are found with the test sets.

The proof in the FSM is dependant on the connection between application and specification. This means that all assumptions need to be met so that the test set can prove equivalence between specification and application. Test creation techniques are derived from a number of assumptions. The first one is that there exists a dependable procedure that gets the machine returning to its initial state. A second assumption is that the application is abstract and represented with a model. Another is that the tester knows the upper limit of the states in the application. The last assumption is reduced specifications. This situation is expected to be met by most of the test generation methods. These assumptions are needed by many techniques when you look at the literature [66] [11] [22] [41] [17] [63]. The W-set is the source of these methods.

The W-set uses the transition cover set to reach the states, while the characterization set is used to identify the states. In other methods, different algorithms have been developed to identify the state. Examples of these algorithms include identification sets [22] and separation families [41]. The length of the test suite is the main criterion in the performance of test methods. For this reason, research focuses on shortening test suites without reducing performance. The experimental results on this subject are in many stud-

ies [18][14]. These studies show that traditional methods can produce many short test cases; current methods produce less and longer test cases to reduce the test set. Current methods related to the subject are based on defining differential test cases with less effect on the test dimension. This strategy is called on-the-fly strategy [17] [31] [62]. In addition, current methods attempted to remove some of the assumptions [28] [26] [61] [55]. However, many test set creation methods have different limitations. Examples include strongly connected and reduced FSMs.

It is possible to say that Gonenc [26] inspired other methods using DS to construct test cases, with his work in 1970. Gonenc creates test cases by manipulating two types of sequences in his method. The first sequence aims to identify all the states. The second sequence is created to test transitions. However, the prevalence of the W set has eliminated the use of DS [55]. The W set can be applied to all reduced FSMs [24] [55]. Nevertheless, the amount of sequences when you look at the W set causes the control sequence to increase exponentially.

Most discussions are about deterministic FSMs. In addition, non-deterministic FSMs were discussed in a small number of articles. There are two situations that bring about non-determinism [71] [29] [52]. The first is that a state reacts differently to the same input. The second case is that the internal transitions are found and the machine switches to a different state without any output [71]. Non-determinism often precludes deciding which one is next. For this reason, the input sequences used for deterministic FMS are replaced by test strategies. These test techniques in many cases are described as trees. Inputs that may or may not be used are indicated by tree transitions. After a certain output, which input will be applied, is determined by the transition of trees. Cheung and Zhang [71] investigated some optimization problems about identifying and reaching the states. Hierons [29] provides an algorithm that generates a test case under the input / output sequences already observed in each case. Hierons' adaptive algorithm aims to reduce the size of the test package in non-deterministic FSMs. The test case is additionally accepted as a tree. Yevtushenko and Petrenko [53] suggest an algorithm for generating test cases in non-deterministic FSM. Researchers define the assumptions that need to be taken into consideration. In doing so, they guarantee that all the faults in the given domain can be found, as in the methods of deterministic FSMs.

Almost all test case production methods are based on a reduced specification. A variety of algorithms have been proposed to increase the flexibility of test generation methods and to manipulate simpler models in a redundant state in order to eliminate re-

dundancy in the FSMs [27] [50]. Most minimization algorithms identify and remove unnecessary states and transitions [27] [50]. What's more, doing so preserves the equivalence between the original and the modified FSM. As a result, a reduced FSM is created. Getting a simplified model is one of the unwanted and difficult to solve situations. Unspecified entries in a state are assumed to be of the "do not care" type by the mentioned minimization algorithms [41]. This hypothesis does not exist in the majority of real life systems. Unspecified inputs often mean that an input is not active. As a result, the implementation of traditional FSM minimization causes some test cases to fail. In such cases, additional operations can be performed on the model. Thus, the necessary conditions are met. The realization of these operations requires the use of heuristic methods. As a result, the error detection power of test sets may be weak.

It is important that the maximum amount of states in the application is defined because this information affects the error detection power. Some techniques utilize this to catch more errors when creating tests. The excess of the error results from the contradictions of the amount of states in the application, in the test model.

State-Counting [52] can be applied to non-reduced and reduced models. State-Counting also gives similar results with traditional methods in terms of test capture. As far as we know, State-Counting is the only method in the literature that provides these two properties. State-Counting also works the same way as conventional methods when applied to reduced specifications. The difference of SC from other methods is that it can be applied directly to non-reduced FSMs. On the other hand, SC has some scalability issues because the strategy used to separate states produces rapidly growing test sets according to the number of test cases. As a result, large, non-practical test suits can be obtained from models.

In this work, Fourier expansions of boolean functions are used in the selection phase of test cases. Moving from this, when looking at the Fourier expansions of boolean functions, they have been used for many years [69]. One of the most important situations in the Fourier expansion is that Boleen functions take real values. Walsh, one of the earliest researchers on Fourier expansion, has established a complete orthonormal basis for $L^2([0,1])$, which consists of fixed, +1,-1 valued functions at dyadic intervals. The first major work on Walsh functions was performed by Paley [49]. In his work, Paley achieved strong results in shortening L^p norms of the Walsh series. The next major development of the Walsh series is the work that Vilenkin [67] and Fine [20] did independently of each other. In these studies, a more natural view of Walsh functions was introduced as a

discrete group (Z_2^n) characterization. In the 1950s and 60s important studies were carried out. These studies are based on the arrangement of Walsh and Rademacher functions using binary expansions. On the other hand, Bonami [8] and Kiener [36], in their work on the subject, interpret the bits symmetrically and arrange the fourier characters using — s — instead of max (s). In addition, Bonomi got the earliest hypercontractivity results for Boolean cubes. Bonomi's research is considered an important tool for the analysis of Boolean functions.

In Boolean functions and Computer Science, in research on "switching functions," the use of Boolean logic dates back to the late 1930s. It is also seen that the idea was mostly contributed to by Nakashima [45], Shannon [59] and Shestakov [60]. Muller [43] is the first person to use Fourier coefficients in the research of Boolean functions. Muller suggests that when classifying all functions according to certain equations, they are calculated. Ninomiya [46] is one of the most important people publishing the Fourier coefficients of binary functions. In his studies, Ninomiya expanded the use of Muller's fourier coefficients in the categorization of Boolean functions according to different isomorphisms. On the other hand, Golomb [25] worked independently on the same project. What is more, he is the earliest person to discover the relation to the Wash series.

From the beginning of the 1960s, "Fourier-Walsh analysis" has begun to be used in Boolean functions. With reference to the works of Lechner [38] and Karpovsky [34], many symposiums were held in the 1970s in relation to the applications of Walsh functions. However, until the work of Kahn, Kalai and Linial [33] in 1988, the use of Boolean analysis in theoretical computer science had decreased. The original analysis for the linearity test of Blum, Luby and Rubinfeld [7] is a combinatorial. Essentially, the essence of the analysis emerged in the work of Roth [56] in 1953. Besides Bellare and his friends, Kaufman, Litsyn, and Xie [35] are among those who have improved the results of Roth's analysis. The sortedness function was developed by Ambainis, [3] and the hemi-icosahedron function was developed by Kushilevitz [47]. Furthermore, the fast algorithm for calculating Fourier transform was developed by Lechner [39].

In this article, the F and F_w method was used to create test generation. An important point of the F-method is that the assumptions used are more flexible than the other methods. In addition, the length of test suites can be changed with parameters used in the F-method. What's more, transitions cover can also be added to the F-method as W uses.

The F-method uses Fourier expansions of binary functions during the test selection phase. The transitions that affect the function most are determined by the Fourier

expansions of the binary functions. This ensures that the most effective transition to the function is achieved. That is, critical transitions are achieved. So, the most critical transitions are tested.

The main purpose of this work is to find the points that have the greatest effect on the function and to create tests for the errors that may occur at those points, because the slightest change in the points that have the most effect on the function causes the fault. While creating the test, it is expected that prioritizing these points will increase the critical performance.

CHAPTER 3

TESTING FINITE STATE MACHINES

Finite State Machine (FSM), as a formal modelling technique to represent both circuits and software, has been widely used in testing. FSM testing is a well-studied subject and there are several test generation methods like W, Wp, UIO, UIOv, DS, HSI and H in the literature. However, the current increase in the demand for pervasive and safety critical systems as well as the increase in software size calls for more rigorous methods that can produce better test suites, particularly in terms of test suite size, test case lengths, time spent for test generation, fault detection ratio and fault exposing potential.

3.1. Finite State Machines

The finite state machine (FSM) model is a mathematical model with discrete inputs and outputs. This model is used for modeling hardware and software systems. Examples of software and hardware systems are the text editor, compilers and synchronous sequential circuits. Even digital computers can be seen as suitable systems for this model. Therefore, the finite state machine model for computer science and engineering is an important model.

There are many types of finite state machines and it is possible to classify as finite state recognizer and output generating state machine models. Each of these classes has their own subclass. However, when we say finite state machine in short, the basic model, the deterministic finite state recognizer model, is understood.

3.1.1. Finite State Recognizer

3.1.1.1. Deterministic Finite State Machine

In the basic model, deterministic finite state machine is defined as follows:

$$DFA =< Q, \Sigma, \delta, q_0, F > \tag{3.1}$$

- Q is a set of finite number states,
- Σ is input alphabet with finite number of input symbols,
- δ is state transition function,
- q_0 is the initial state,
- F is the set of final states.

With the state transition function, each current state and the input symbols can be said to match the next state in the deterministic finite state machine.

Transition Diagram For the definition of the transition function, a so-called transition diagram is often used. In the example (Figure 3.1), the arrows refer to the transitions, and the circles refer to states. The state referred to by start is the start state. Double circle is the output state. The numbers on the transitions refer to the inputs.

Set of Strings Recognized by DFA Depending on the model described, an input symbol is applied to the input of the DFA each time. Each input symbol causes the DFA to move the next state. The DFA maintains its state until a new input symbol is applied.

Regardless of the number of input symbols applied, the model works because it is deterministic. Initially in the case of q_0, the DFA reaches a certain state after a string of a certain number of input symbols is applied. The relationship between q_0, w and q_i (reached state) can be used by using the state transition function as follows:

$$\delta(q_0, w) = q_i \tag{3.2}$$

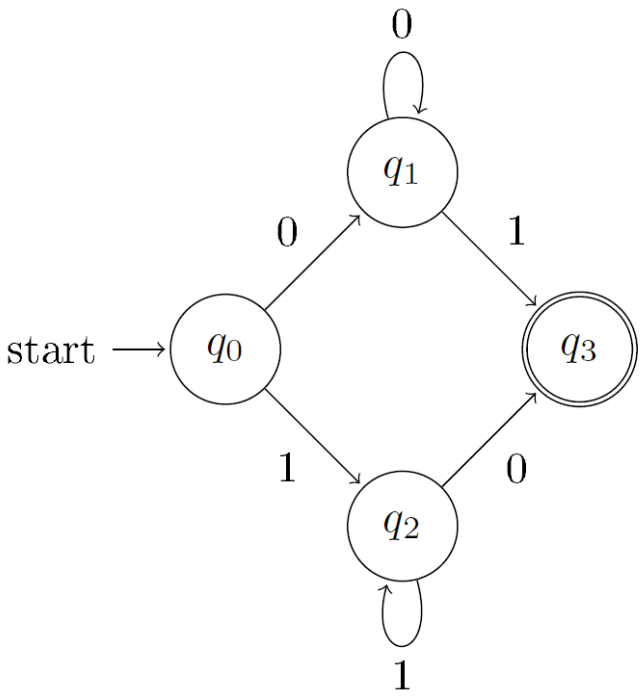

Figure 3.1. Transition Diagram

Starting from the q_0 state and reaching the q_i state after the introduction of the w input string, the DFA recognizes the w input string if the q_i is an exit state. The input alphabet S = $I_1, I_2, I_3.....I_k$ is an infinite set of input strings that can be applied to a DFA. Each string of finite or infinite lengths of symbols in the input alphabet is contained in this set. Some of the strings of symbols in the input alphabet are recognized by DFA, while others are unrecognized strings. At this stage, it is possible to define a set of strings recognized by a finite state machine (M) as follows:

$$T(M) = w|\delta(q_0, w) = q_i \epsilon F \tag{3.3}$$

3.1.1.2. Nondeterministic Finite State Machine

Due to the difficulty of use of the deterministic model, a non-deterministic model is developed which is easier and more flexible to use. Non-deterministic finite state machine (NFA) is defined as follows:

$$NFA =< Q, \Sigma, \delta, q_0, F > \tag{3.4}$$

In this definition, the meanings of Q, Σ, q_0 and F are the same as those of the deterministic model. The only difference between the two models is in the definition of the transition function.

In the deterministic model, the transition function is defined as a mapping from (Q x Σ) to Q, whereas in the non-deterministic model the transition relation is defined as a mapping to subsets of Q from (Q x Σ).

3.1.2. Output Generating State Machine Models

The output generating model is a model that generates an output string in response to an input string. In this respect, it is also possible to see the output generating output model as a model that converts input strings into output strings. Thus, two main types of finite state machine can be characterized as recognizers and transducers. In fact, it can be thought that the recognizers produced output. From this point of view, the output generating finite state machine model can be seen as a wider model, including the recognizer model.

There are two types of output generating state machine called Moore and Mealy Machine. While the Moore machine is a model that produces state-level output, the Mealy machine is a model that produces output at the state transition level.

3.1.2.1. Moore Machine

Moore Machine is defined as

$$M =< Q, \Sigma, \Delta, \delta, \lambda, q_0 > \tag{3.5}$$

- Q is set of finite states,
- Σ is set of input symbols,
- Δ is set of output symbols,

- δ is state transition function (A mapping from (Q x Σ) to Q),
- λ is output function (A mapping from Q to Δ),
- q_0 is the initial state.

The Moore machine can be seen as the generalization of the DFA model. In other words, the DFA model can be considered as a special Moore machine.

3.1.2.2. Mealy Machine

Like the Moore machine, the Mealy machine is also defined as M = $< Q, \Sigma, \Delta, \delta, \lambda, q_0 >$.

The definition of the 5 elements other than the output function is the same as in the Moore machine. In other words, the only difference between Moore and Mealy machines is the output function. The output function, defined as a mapping from Q to Δ for Moore machines, is defined as a mapping from (Q x Σ) to Δ for Mealy machines.

3.1.3. Features of The Finite State Machine

The basic features of the FSM are as follows:

- **Completely Specified:** An FSM M is said to be completely specified if the behavior of the FSM is specified for every state and every valid input.
- **Deterministic:** An FSM M is said to be deterministic if
  1. for each input a ϵ I there is at most one state transition defined at each state of M
  2. there is no internal event causing state transitions in the FSM. A timeout is an example of internal events that can cause a state transition.
- **Strongly Connected:** An FSM M is said to be strongly connected if any state is reachable from any other state.

- **Distinguishable States:** To distinguish between the S_1 and S_2 states of the M machine, if an input sequence of at least n length is required, these are called n-distinguishable states. If the S_1 and S_2 states can be n-distinguishable, then these two states are (n-1) -equivalence.

- **Machine Equivalent:** If there is an equivalent state in the M_2 machine for every state of the M_1 machine, and vice versa, the M_1 and M_2 machines are equivalent.

- **Machine Minimization:** Reduction or simplification of a machine M means finding a machine M-reduced, equivalent to M, with the smallest number of states.

An example FSM is shown in Figure 3.2. There are 4 states, 2 inputs and 2 outputs. It is completely specified, deterministic, reduced, and strongly connected. This FSM will be used in the examples.

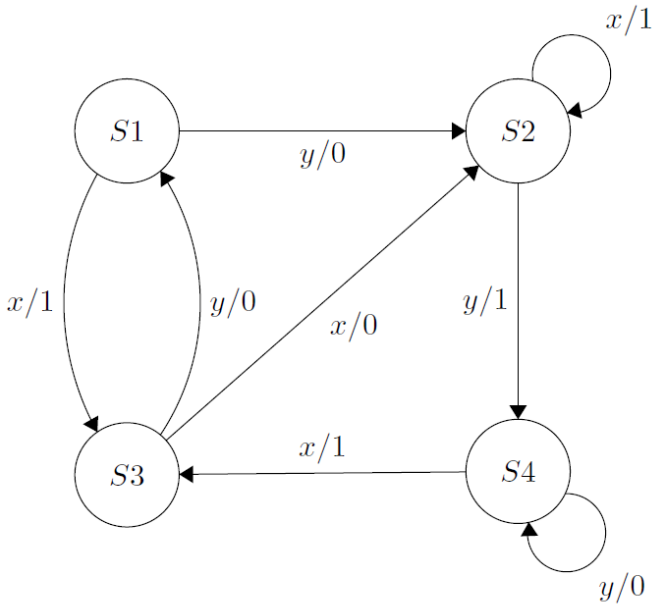

Figure 3.2. The Example FSM

3.1.4. Eliminate Incompletely Specified Problem

Some parts in FSMs are not filled or forgotten because these parts in FSMs are considered unnecessary. Such machines are called Incompletely Specified. FSMs with

such features are a problem for FSM testing tools, because FSM tests use each input variation of states. In this case, the lack of some variations creates problems. That is, knowing which state to pass from a state is of great importance in FSM tests.

Some FSM tests have solved the Incompletely Specified problem. HSI is an example. But the important thing for us in HSI is that there was no change in the characterization set. The main change in HSI is the presence of spanning tree and topological sort operations on it before the characterization set is applied in FSM. HSI then combines the result of the characterization set and topological sort.

For FSMs with an incompletely specified problem, a new state is added to the FSM to solve this problem. The state added to the FSM returns to its next states without output. All missing values in the FSM are filled with the new state without output. Once these conditions are created, only the outputs are missing. This problem is solved by accepting the outputs as a new variable. An example solution is given in the Table 3.1.4 and Table 3.1.4.

Current State	Next State, Output	
	x	y
S_1	S_3,0	-
S_2	S_2,1	S_4,0
S_3	S_2,1	S_3,1
S_4	-	S_4,0

Current State	Next State, Output	
	x	y
S_1	S_3,0	S_5,-
S_2	S_2,1	S_4,0
S_3	S_2,1	S_3,1
S_4	S_5,-	S_4,0
S_5	S_5,-	S_5,-

3.1.5. Reduction of Finite State Machine

Multiple models have been defined so far in terms of finite state machine such as the recognizer model and output generating model. In both the recognizer model and output generating model, the complexity of the defined finite state machine is directly proportional to the number of states. Therefore, when a finite state machine is given, the reduction of this finite state machine can be important. The reduction or simplification of a finite state machine is the equivalent of this finite state machine, with the least number of finite states.

Let's assume that M Machine has $\{S_1, S_2, ..., S_i\}$ states and 0,1 inputs expressed by x. S_2 is the **x-successor** of S_1 if switching from S_1 to S_2 with x input symbol in M. If the machine switches from the S_1, S_2, ..., and S_i states to the S_j state with the x input symbol, the **x-precedence** of the S_j state is $\{S_1, S_2, ..., S_i\}$ in M.

When the machine M is in either states S_1 and S_2, whichever input symbol is applied, if the machine always produces the same output symbol, these are called **1-equivalent states**. When the M machine is in any of the S_1 and S_2 states, regardless of which input string is used, the length of which is n or less, if the machine always produces the same output string, these conditions are called **n-equivalent states**. When the machine M is in either states S_1 and S_2, regardless of its length, no matter which input string is applied, if the machine always produces the same output string, it is called **equivalent states**.

In order to differentiate the S_1 and S_2 states of the machine, if an input string of at least n length is required, these are called n-distinguishable states. If the S_1 and S_2 states are **n-distinguishable**, these two conditions are equal to n-1 equivalent.

When an M machine is given, the reduction or simplification of this machine means that the smallest number of states is found from machines equivalent to this machine.

Equivalence partitions are used for the reduction of finite state machine. For an M machine, the k-equivalence partition indicated by P_k is a division in which the k-equivalent states are located in the same section.

P_0, P_1, P_2,... is found in order to find the equivalence partitioning of the machine until P_{k+1} = P_k is obtained. When P_{k+1} = P_k is reached, it is understood that the equivalence partition is P = P_k and the derivation is finished. The states in the same section are merged as single state. After this process, reduced FSM is obtained. The main formula

for equivalence partitioning is below.

- S_1 and S_2 must be equivalent (P_m must be in the same section in the equivalence partition).
- For all x input symbols, the x-successors of the S_1 and S_2 states must also be equivalent ((P_m must be in the same section in the equivalence partition).

3.2. Testing Methods

Assume the design of an FSM model called M, which consists of the requirements of the system. Conformance testing is used to check whether the implementation of M meets all the requirements. The general features of conformance testing are as follows:

- To obtain a sequences of state transitions and output functions from M,
- Transformation of each sequence of state transitions and output functions into a test sequence,
- Testing of M's implementation by the obtained test sequences,
- Finally, test the result using enough test sequences.

FSM design has state transition and output function. There are methods that only test the output function (transition tour method). Methods that only test the output function are easy to test because the FSM designs produce output. But in FSM, when it goes from one state to another, it is difficult to understand the next state. Hence, it doesn't produce any output. Many methodologies have been developed to do this such as unique input output sequence and characterizing sequence.

These methods are generally applied by the general procedure. This procedure first goes to the place to be tested. Then, test input is confirmed. Next, this test input confirms the next state. Finally, it returns to the beginning.

3.2.1. Some Definitions

The following are some definitions of test, test case, and test suite.

Test Case: A set of a string developed to verify a particular feature or functionality for FSM.

Test: A set of one or more test cases.

Test Suite: A set of several test cases for a system under test to verify that it has some specified set of behaviours.

FSM testing: A nonfunctional testing technique which validates whether the model meets standards or not. The basic processes of FSM testing are as follows.

Two basic tests will be explained in this section : **unique input output sequence** and **characterization Set**.

3.2.2. Unique Input Output Sequence

The method will be briefly explained here, and details can be consulted in the literature [23] [9]. The FSM must have certain properties to use the UIO sequence. These are completely specified, deterministic, reduced, and strongly connected.

It consists of two phases. The first stage is to create the UIO tree, and the second stage is to get the results from the tree according to the priority order. The algorithm for creating a UIO tree is given in Algorithm 1.

An input output sequence $y/\lambda(s_i, y)$ is said to be a UIO sequence for state s_i if and only if $y/\lambda(s_i, y) \neq y/\lambda(s_j, y), i \neq j, \forall s_j$in M.

Input: FSM
Output: UIO tree
1 Create an initial path matrix (PM);
2 **while** *Find a nonterminal PM ϵ UIO Tree* **do**
3 | New path vector(PM') is created with a given PM and an input - output (x / y) pair.;
4 | Add PM' to UIO Tree.;
5 | **if** *PV' satisfies termination condition C1 or C2* **then**
6 | | Mark PV' as a terminal node.;
7 | **end**
8 **end**

Algorithm 1: Generation of UIO Tree

- *C1:* If there is only one state within the path vector, it is called **singleton matrix**. If multiple states are connected to the same state in the tail, it is called **homogeneous matrix**.

- *C2:* On the path from the initial vector to PM, there exists PM" such that $PM' \subseteq PM''$.

Firstly, the path matrix structure is created in an FSM. This structure consists of the head and tail parts. The head contains the initial states. The tail defines the currently available states $\begin{matrix} head-> \\ tail-> \end{matrix}\begin{bmatrix} x & y & z & t \\ x & y & z & t \end{bmatrix}$. Head cover all states in the FSM. In the beginning of the Algorithm 1, the head and tail states are equal to each other. The path Matrix $\begin{bmatrix} x & y & z & t \\ k & p & s & r \end{bmatrix}$ indicates that the FSM is going from x to k, from y to p, from z to s, and from t to r. Paths between path matrices are determined by inputs and outputs $\begin{bmatrix} x & y & z & t \\ f & g & h & k \end{bmatrix} \xrightarrow{input/output} \begin{bmatrix} x & y & z & t \\ y & s & m & r \end{bmatrix}$. A new path matrix is created with a given path matrix and an input - output (x / y) pair. When x input and y output is given, we can find the new states. In this way, using new states, the new path matrix is created in UIO Tree. $\begin{bmatrix} x & y & z & t \\ y & s & m & r \end{bmatrix}$ is node and $\xrightarrow{input/output}$ is arrow in UIO Tree. This continues until singleton or homogeneous state is found. If there is only one state within the path matrix, it is called singleton matrix $\begin{bmatrix} x \\ y \end{bmatrix}$. If multiple states are connected to the same state in the tail, it is called homogeneous matrix $\begin{bmatrix} x & y & z & t \\ a & a & a & a \end{bmatrix}$. If the path vector, which isn't the singleton or homogeneous vector, is located in the upper branches of the tree, it is terminated again.

For example, UIO sequence is applied to example FSM:

Firstly, all states are selected as root in UIO sequence. Then the states are determined according to which input value they will go to. For example, from x / 0, only 3 to 2 states are possible. Thus, when x / 1, y / 0, y / 1 is found, the first level is finished (Figure 3.3). The algorithm uses breadth first search. There are two situations that prevent branching. The first one is that there are one or more same states. The other is that the same states are found above in UIO tree. Branching is done using the above rules (Algorithm 1). At the end of the branching, the states from the top to the bottom are selected. As an example, 3 and 2 is selected in the first section, 4 in the second section and 1 in the third section. As a result, x, y, yy, yxx tests are created(Table 3.1).

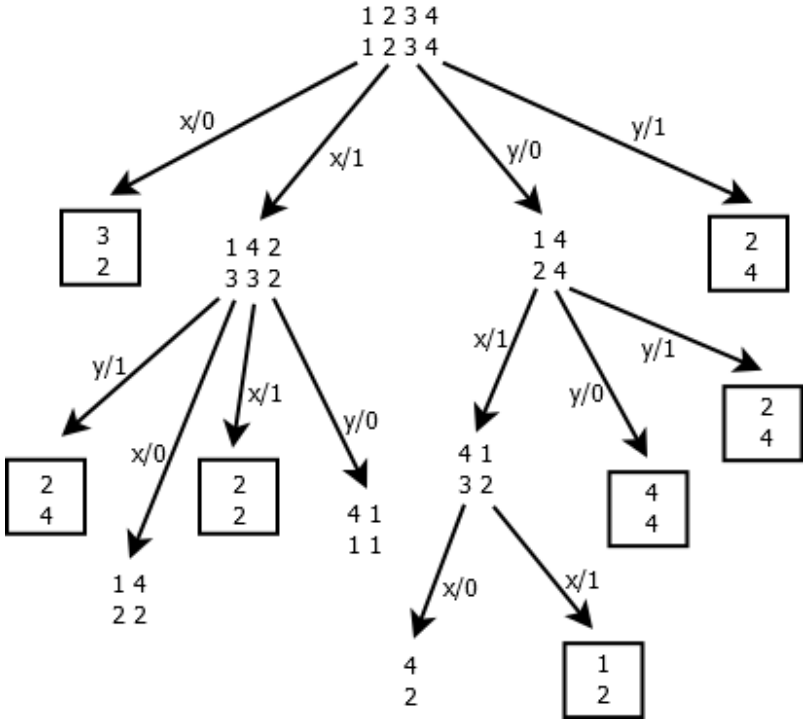

Figure 3.3. UIO Tree

Table 3.1. Created Test Scenarios

States	Input	Output
S_1	YXX	011
S_2	Y	1
S_3	X	0
S_4	YY	00

3.2.3. Characterization Set

Let machine M be a minimal and complete. Characterization set (W-set) is any states q_i and q_j, W-set contains a string s (input sequences that distinguishes every two different states) such that $O(q_i, s) \neq O(q_j, s)$. For each complete reduced FSM, there always exists a distinguishably(characterization) set W.

Input: FSM table
Output: W-set
1 Construct k-equivalence partitions: $P_1, P_2, ...P_m$ (Explained in Section 3.2.3.1);
2 Traverse the k-equivalence partitions in reverse order to obtain distinguishing sequence for each pair of states(Explained in Section 3.2.3.2);

Algorithm 2: The Construction of W-set

3.2.3.1. Construct K-equivalence Partitions

First, to find the W-set, equivalence partitions are found. In Mearly machines, for equivalence partitions, groups are created first by using outputs. The new groups are then derived from these groups until we don't have any new groups. When all states are separated into groups, the algorithm terminates. These processes always converge. If the FSM is a minimum, all states are separated.

When creating the first table below, the states in an FSM are written in the current states. Then, output and next state values are written according to x and y. This table specifies which output is to be taken when the input value is given. At the same time, according to the input value, we can see which one is next state.

To give an example, the first separation was made using the outputs. When performing this separation process, these values are also taken into account if the separation occurs with more than one input value. For example, in the following table, there is separation according to x and y. The first one is separated, and then the others. In the example, all values except S_1 and S_4 are separated from each other.

Table 3.2. FSM Table

Current State	Output		Next State	
	x	y	x	y
S_1	1	0	S_3	S_2
S_2	1	1	S_2	S_4
S_3	0	0	S_2	S_3
S_4	1	0	S_3	S_4

After separating the states using the outputs, P_1 is obtained. This is achieved by writing each separated state to different groups in the table. The states in the Next state table are represented by group indexes. For example, the state S_2 of the second group

goes to S_4 with the y input. Since S_4 input belongs to group 1, it is written as S_{41}. The last number indicates the which group belongs to S_4.

Table 3.3. P1 Table

Group	Current State	Next State	
		x	y
1	S_1	S33	S22
2	S_2	S22	S41
3	S_3	S22	S33
1	S_4	S33	S41

After P1 values are found, other P values are obtained by looking at group differences of states. This process ends when all states are separated. Another important point is that the total amount of P cannot be greater than the number of states.

The values of the P_2 table are given in Table 3.4. When these values in the table are considered, it is seen that all the states are separated. If the FSM was not a minimum, it would not have been separated. Therefore, this is the minimum requirement for this algorithm. On the other hand, the number of P's are equal to the maximum input length in the generated test set. So in this example, since the last P is P_2, the maximum input length is 2 in test suite.

Table 3.4. P2 Table

Group	Current State	Next State	
		x	y
1	S_1	S33	S22
2	S_2	S22	S44
3	S_3	S22	S33
4	S_4	S33	S44

3.2.3.2. Obtain Characterization Sequence For Each Pair of States

All states in the FSM are selected in pairs. We look whether the selected pair is in the same group in P_i and in different groups in P_{i+1}. When this is detected, it is

found that the P_i table has the separating input. The separating input is saved. The pair corresponding to this input is detected. With these new pairs found, the algorithm is run again. The algorithm terminates when it reaches the FSM table.

For example, looking at the pair S_4 and S_1, P_i is equal to P_1; P_{i+1} is equal to P_2. It is understood that the value separating them is y. When y is applied to the states, the pair S_2 and S_4 are obtained. For the pair S_2 and S_4, the Pi is equal to P1, so the FSM table is checked. As a result, it is seen that it is separated by the y value. This value is combined with the y value previously found. Then, it is rewritten backwards and the characterization sequence is found for these two values. Then, the algorithm is run again for other pairs. The algorithm terminates when all pairs are finished [42][14][65].

- S_1,S_2 = y
- S_2,S_3 = y
- S_3,S_4 = x
- S_4,S_1 = yy
- S_1,S_3 = x
- S_2,S_4 = y
- W-set = {yy,y,x}

3.2.4. Creating of Transition Cover Set

To visit all transitions, the transition cover set is created. The algorithm used to create the Transition cover set is given below.

- Let's assume that the test tree has been created up to the k level. k + 1 level is as follows.
- n node in k level is selected. If n node is in any level from 1 to k, then n is the node. In this case, n is not further expanded. If n is not a leaf node, ie not in any level from 1 to k, then new branches are added between nodes and n.
- This applies to all nodes in k level.

- When there is no more new branching, the transition cover set is created by finding all the paths starting from root and ending on any node.

The tree generated according to the above algorithm is as Figure 3.4.

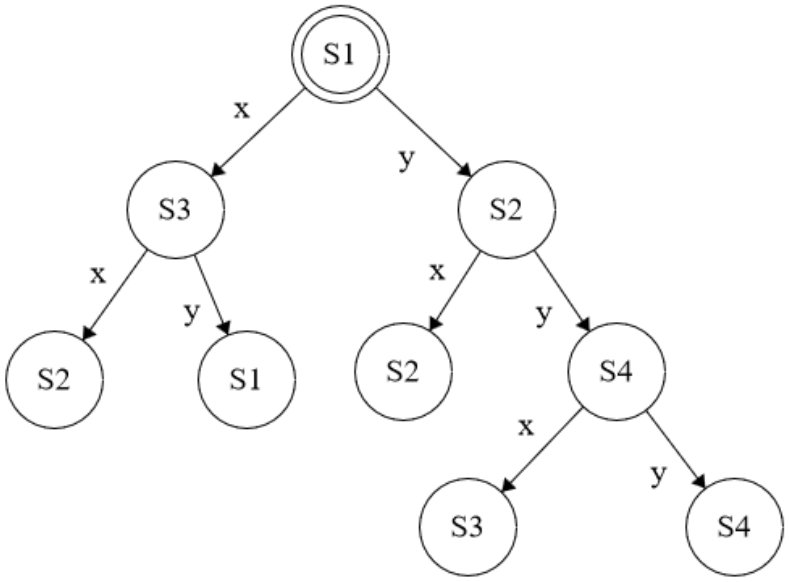

Figure 3.4. The Transition Cover Tree

The transition cover set created from the above tree:

- Transition cover set = $\{x, y, xx, xy, yx, yy, yyx, yyy\}$

CHAPTER 4

FOURIER ANALYSIS-BASED TESTING OF FINITE STATE MACHINE

Finite state machine (FSM) is a widely used modeling technique for circuit and software testing. The FSM test is a well-studied topic, and in the literature there are several test production methods, such as W [11], Wp [22], UIO [57], UIOv[68], DS [26], HSI[52] [51] [70] and H [16] [15]. Although there are various methods in the literature for the testing of FSMs, there is a need for alternative test methods with high performance in subjects, such as the size of the test suite and fault detection ratio.

In this study, two new test production methods, F and F_w, have been proposed. The proposed test creation methods are based on the Fourier analysis of Boolean functions, unlike other test generation methods. With the Fourier analysis of Boolean functions, discrete structures can be converted into polynomials. Thus, how much input combinations affect output can be found. Fourier transformations have been studied extensively in mathematics, computer science and engineering.

The proposed methods differ in terms of the points tested. The F method only tests outputs; the F_w method also tests the next state with the outputs. To create an F method, first, FSM is converted to a binary form. This is because the Fourier transform function accepts input in binary form. For example, State A is 00, state B is 01, X is 0, and Y is 1. Using Fourier transform function, coefficients are calculated. The coefficients show us the powers of input and state combinations to influence the output. The obtained coefficients are listed. The input combination of the highest coefficient is then taken according to the given parameter. After this operation, appropriate transitions are taken to the selected input combination. Selected transitions are combined with the greedy algorithm to create tests.

To create the F_w method, the state verification part of the W method is taken. The output of the F method with the state verification part of the W method is combined with the Cartesian product. Thus, with the outputs, the next state is tested.

4.1. Fourier Expansion

In this section, the basic concepts of Fourier analysis of binary functions are introduced. Although Fourier analysis of boolean functions has been a subject of great interesting in the field of mathematics and engineering over the past decade, it has of yet very few practical applications. Boolean functions are usually expressed as $\mathbb{B}^n \rightarrow \mathbb{B}$ where $\mathbb{B} = \{True, False\}$. As a requirement of Fourier analysis of boolean functions, instead of *True* and *False*, 1 and -1 will be used as false and true values ($f : \{-1, 1\}^n \rightarrow \{-1, 1\}$). Definitions and theorems are given below without proof. Further explanations, examples and theorems can be found in the work of O'Donnell [48] and Wolf [13].

Theorem 4.1.1 (Fourier expansion) *Every function $f : \{-1, 1\}^n \rightarrow \mathbb{R}$ can be uniquely expressed with a Fourier expansion,*

$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$

where c_S is the Fourier coefficients and $\prod_{i \in S} x_i$ is parity function.

Inner product inner product can be found by

$$< f, g >= \frac{\sum_{x \in \{-1,1\}^n} f(x)g(x)}{2^n}$$

The Fourier coefficients can be calculated by

$$\hat{f}(S) =< f, x_s >$$

In the Fourier analysis of Boolean functions, the Boolean function is composed of spectral components. Each spectral component is assigned a coefficient. Note that the use of {-1, + 1} in real valued functions instead of {true, false} makes the exclusive-or operation a simple multiplication. Figure 4.1 shows the eight components of a 3-input Boolean function. Hereby, the spectral components with more than one variable depict X-or operation. For example, the component ac actually means $a \oplus c$, and the component abc means $a \oplus b \oplus c$.

Here is how the Fourier expansion is calculated using a simple binary function in the Table 4.1. Our function is:

$$f = a + bc$$

It can be easily seen that the accuracy of f is as follows:

[F F F T T T T T]

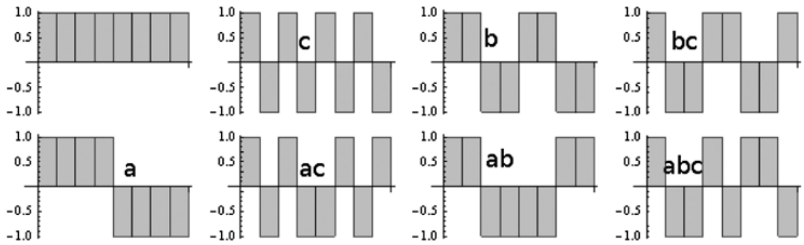

Figure 4.1. The eight spectral components of a 3-input Boolean function

Table 4.1. Truth table for function $f = a + bc$

a	b	c	$f : \mathbb{B}^n \mapsto \mathbb{B}$	$f : \mathbb{B}^n \mapsto \mathbb{R}$
F	F	F	F	1
F	F	T	F	1
F	T	F	F	1
F	T	T	T	-1
T	F	F	T	-1
T	F	T	T	-1
T	T	F	T	-1
T	T	T	T	-1

The general structure of the Fourier expansion is:

$$f = \hat{f}(\emptyset) + \hat{f}(1)a + \hat{f}(2)b + \hat{f}(3)ab + \hat{f}(4)c + \hat{f}(5)ac + \hat{f}(6)bc + \hat{f}(7)abc$$

according to Definition 4.1, the first Fourier coefficient $\hat{f}(\emptyset)$ is calculated as follows:

$$\hat{f}(\emptyset) = \frac{1}{2^3}(1 + 1 + 1 - 1 - 1 - 1 - 1 - 1) = 0.25$$

Similarly, the second coefficient is found as follows:

$$\hat{f}(1) = \frac{1}{2^3}(1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 - 1 \cdot 1 - 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1) = 0.75$$

Similarly, the third coefficient is found as follows:

$$\hat{f}(2) = \frac{1}{2^3}(1 \cdot 1 + 1 \cdot 1 - 1 \cdot 1 - 1 \cdot -1 + 1 \cdot -1 + 1 \cdot -1 - 1 \cdot -1 - 1 \cdot -1) = 0.25$$

Similarly, the fourth coefficient is found as follows:

$$\hat{f}(3) = \frac{1}{2^3}(1{\cdot}1{\cdot}1{+}1{\cdot}1{\cdot}1{+}1{\cdot}{-}1{\cdot}1{+}1{\cdot}{-}1{\cdot}{-}1{-}1{\cdot}1{\cdot}{-}1{-}1{\cdot}1{\cdot}{-}1{-}1{\cdot}{-}1{\cdot}{-}1{-}1{\cdot}{-}1{\cdot}{-}1) = 0.25$$

Similarly, the fifth coefficient is found as follows:

$$\hat{f}(4) = \frac{1}{2^3}(1 \cdot 1 - 1 \cdot 1 + 1 \cdot 1 - 1 \cdot -11 \cdot -1 - 1 \cdot -11 \cdot -1 - 1 \cdot -1) = 0.25$$

Similarly, the sixth coefficient is found as follows:

$$\hat{f}(5) = \frac{1}{2^3}(1{\cdot}1{\cdot}1{+}1{\cdot}{-}1{\cdot}1{+}1{\cdot}1{\cdot}1{+}1{\cdot}{-}1{\cdot}{-}1{-}1{\cdot}1{\cdot}{-}1{-}1{\cdot}{-}1{\cdot}{-}1{-}1{\cdot}1{\cdot}{-}1{-}1{\cdot}{-}1{\cdot}{-}1) = 0.25$$

Similarly, the seventh coefficient is found as follows:

$$\hat{f}(6) = \frac{1}{2^3}(1{\cdot}1{\cdot}1{+}1{\cdot}{-}1{\cdot}1{-}1{\cdot}1{\cdot}1{-}1{\cdot}{-}1{\cdot}{-}1{+}1{\cdot}1{\cdot}{-}1{+}1{\cdot}{-}1{\cdot}{-}1{-}1{\cdot}1{\cdot}{-}1{-}1{\cdot}{-}1{\cdot}{-}1) = 0.25$$

Similarly, the eighth coefficient is found as follows:

$$\begin{aligned}\hat{f}(7) = \frac{1}{2^3}(&1 \cdot 1 \cdot 1 \cdot 1 + 1 \cdot 1 \cdot -1 \cdot 1 + 1 \cdot -1 \cdot 1 \cdot 1 + 1 \cdot -1 \cdot -1 \cdot -1 \\ &- 1 \cdot 1 \cdot 1 \cdot -1 - 1 \cdot 1 \cdot -1 \cdot -1 - 1 \cdot -1 \cdot 1 \cdot -1 - 1 \cdot -1 \cdot -1 \cdot -1) = 0.25\end{aligned}$$

When all coefficients are calculated, the Fourier expansion is obtained as follows:

$$f = 0.25 + 0.75a + 0.25b + 0.25ab + 0.25c + 0.25ac - 0.25bc - 0.25abc$$

4.2. Construction of F-set

The normal form is FSM tables that can be encoded with any character. The binary form is FSM tables which can only be encoded in binary. For example, a case can be written as S_1, S_2... in normal form, while the same is expressed in Binary form as 01, 00, 10... Because the Fourier Transform requires a binary form, FSM are transformed into a binary form. This conversion is shown in Table 4.2 and Table 4.3:

Table 4.2. FSM Normal Form

S	X	S	O
S_1	x	S_3	1
S_1	y	S_2	0
S_2	x	S_2	1
S_2	y	S_4	1
S_3	x	S_2	0
S_3	y	S_3	0
S_4	x	S_3	1
S_4	y	S_4	0

The states in the above table are converted to two digits, as can be seen in the following table. For example, "S_1 is converted to 00; S_2 is converted to 01". The input values x and y are converted to 1 and 0 respectively.

Table 4.3. Converted To Binary Form

S_1	S_2	Input	S_1	S_2	Output
0	0	1	1	0	1
0	0	0	0	1	0
0	1	1	0	1	1
0	1	0	1	1	1
1	0	1	0	1	0
1	0	0	0	0	0
1	1	1	1	0	1
1	1	0	1	1	0

The output function is shown in Equation 4.1. It shows that the output function is dependent upon the inputs and the states.

$$O = \bar{S_1}\bar{S_2}X + \bar{S_1}S_2X + \bar{S_1}S_2\bar{X} + S_1S_2X \tag{4.1}$$

The truth table is obtained using the output function. Truth table, inputs and states are used together to get Fourier coefficients. The example is given in Equation 4.2:

$$O = 0 - 0.5S_1 + 0.5S_2 + 0 + 0.5X + 0 + 0 + 0.5S_1S_2X \tag{4.2}$$

As you can see in the example, some values are low. Therefore, the effect of these values on function is weak. Removing these values with low effect reduces the test set size. This reduces the cost. For this reason, low values have been removed by using a threshold. Two types of parameters are used to determine threshold in the project.

In the first parameter, the coefficients are sorted according to their importance. The user enters the parameter of how much of the coefficients to take. For example, 1/2, 1/4, 1/6. Starting with the highest value, the user selects the scale coefficients determined by the parameter.

In the second parameter, after the coefficients are determined according to the first parameter, the maximum number of transitions for each coefficient is regulated by the parameter.

By applying a certain threshold to these coefficients, the values with little or no effect are subtracted from the Fourier coefficients. The example is given in Equation 4.3.

$$O = -05S_1 + 0.5S_2 + 0.5X + 0.5S_1S_2X \tag{4.3}$$

As shown in Equation 4.3, after selecting the coefficients, appropriate transitions

are selected. As is known, Fourier transform measures change. In order to measure the change, it must contain the new x values, as can be seen in Equation 4.3. However, as can be seen in the table 4.3, the input value does not exist after the change. For this reason, a table containing new input values is arranged. The edited version of the table is given in Table 4.4.

Table 4.4. After all coefficients are applied

S_1	S_2	X	S_1	S_2	Output
0	0	1	1	0	1
0	1	0	1	1	1
1	0	0	0	0	0
0	0	0	0	1	0
1	1	1	1	0	1
1	0	1	0	1	0

After obtaining the Table 4.4, an algorithm was developed to select appropriate transitions with Fourier coefficients shown in Algorithm 3. There are three entries according to this algorithm such as coefficients, transitions and the two parameter values. The output of this algorithm is transitions selected by the coefficients. The maximum number of transitions that are suitable for the coefficients is determined by the transitionsSize variable. The algorithm looks at the suitability of each transition for each coefficient. This is done with the check algorithm.

As for the algorithm of check, the algorithm takes the Fourier variables as a pattern (For example, S_1S_2X is 111 as pattern and If combination is S_2X, pattern is 011). It is checked through a pattern to see if transition is suitable. If the pattern is one and there is no change, the program will return false; otherwise, it will return true. On the other hand, if the pattern is zero and there is a change, the program again returns false; otherwise, it returns true in Algorithm 4 .

After the transitions are selected, they are combined using the Greedy algorithm. To do this, the first transition is considered a solution. The following transitions are always attempted to be added to the previous solutions. Transition that is not included in existing solutions is considered a new solution shown in Algorithm 5.

After the greedy algorithm was applied, sequences were formed.

- $\{S_2S_4S_3, S_1S_3S_2, S_3S_1S_2\}$

Input: Transitions, Terms, TransitionsSize, TermsSize
Output: the most important transitions are selected with using TransitionsSize and TermsSize parameters
Transitions : is list of Transition which consists of edges from one state to another (the next state) with input and output
Terms : is list of Term which consists of the fourier coefficient and parity function)
TransitionsSize: is maximum transitions size given from user
TermsSize : is maximum terms size given from user

```
1  Sort Terms;
2  n = 0;
3  for i ← termSize to 0 do
4      Get last Term in Terms and remove it from Terms;
5      if Fourier coefficient of Term isn't equal to 0 then
6          foreach Transition in Transitions do
7              if check(term, Transition) then
8                  ▷ Check if Transition is suitable for
                     selected transitions or not
9                  Add transition to selected transitions;
10                 if selected transitions size is equal to TransitionsSize then
11                     return selected transitions;
12                 end
13                 Add coefficient to selected coefficients;
14             end
15         end
16     end
17     if selected coefficients size is equal to TermsSize then
18         return selected transitions;
19     end
20 end
21 return selected transitions;
```

Algorithm 3: Pseudo-code that give the selected transitions

In order to link with the root of the FSM, a transfer sequence is added to each sequence obtained after applying the Greedy algorithm. For example, the transfer sequence of the sequence $S_2S_4S_3$ is S_1S_2. The transfer sequence of each sequence varies according to the FSM root distance of that array. Once the transfer sequence has been detected, this transfer sequence is added to the previously obtained sequences. The resulting array is defined as the F-set sequence. In the above example, the F-set sequence is $S_1S_2S_4S_3$. The F-set sequences of the arrays obtained after the Greedy algorithm are combined with the transfer sequences are as follows.

- $\{S_1S_2S_4S_3, S_1S_3S_2, S_1S_3S_1S_2\}$
- $F - set(Output) = yyx, xx, xyy$

Input: Transition, Term
Output: if Transition is suitable for selected transitions or not
Transition: Consists of edges from one bitwise encoded state to bitwise encoded another (the next state) with input and output
Term : Consists of the fourier coefficient and parity function (bitwise variables)

```
1  for i ← 0 to bit length of the bitwise encoded state of Transition do
2      if i'th variable of term is equal to '1' then
3          if i'th bit of the bitwise encoded state of transition is equal to i'th bit
             of the bitwise encoded next state of transition then
4              return false;
5          end
6      else
7          if i'th bit of the bitwise encoded state of transition isn't equal to i'th
             bit of the encoded next state of transition then
8              return false;
9          end
10     end
11 end
12 return true;
```

Algorithm 4: Check(...) Pseudo-Code

4.3. Test Suite Generation Using F-set

In the F and F_w methods, if you want to use the output and state verification test at the same time, any state verification sequence can be added at the end of the output test. In this case, the F-set is replaced by the state transition under the test section. If the state verification sequence is not added to the end of the output test, only the output test is performed, such as the Transition Tour. Consequently, to create a test, transitions that highly affect the output are selected. In both cases, the F and F_w methods allows the creation of shorter test suites without a significant reduction in the rate of error detection. Thus, less time and space are used.

For example, the F-set sequences, which are combined with the transfer sequences after the Greedy algorithm is applied, also form the F-method.

- F-set = F = $\{yyx, xx, xyy\}$

To create the FW method, the state verification part of the W method is taken. The state-verification part of the W method and the F-set are combined with the Cartesian product. Thus, with the outputs, the next situation is tested. The F method with the W

```
Input: Transitions
Output: Solutions
Transitions: is list of Transition which consists of edges from one state to
             another (the next state) with input and output
Solutions  : is hash table where value is solution which consists of list of
             Transition, key is id of solution
1  foreach transition in transitions do
2      if size of solutions is equal to zero then
3          Add transition as first solution in solutions;
4      else
5          assign check to true ▷ check says whether there is a
             new solution or not
6          ;
7          foreach solution in solutions do
8              if the bitwise encoded next state of last transition of solution is
                 equal to the bitwise encoded state of transition then
9                  add transition to end of solution;
10                 check is equal to false;
11             else if the bitwise encoded state of first transition of solution is
                 equal to the bitwise encoded next state of transition then
12                 add transition to begin of solution;
13                 check is equal to false;
14         end
15         if check is equal to true then
16             add transition to new solution;
17         end
18     end
19 end
20 return temp;
```

Algorithm 5: FindSolutions(...) Pseudo-Code

method was run on the same sample FSM. As a result, W set was found as yy, y, x. The combination of the W set and the F set with the Cartesian product is as follows:

$$F_w = Fset \times Wset = \{yyx, xx, xyy\} \times \{yy, y, x\}$$
$$= \{yyxyy, yyxy, yyxx, xxyy, xxy, xxx, xyyyy, xyyy, xyyx\}$$

Transition cover set = $\{x, y, xx, xy, yx, yy, yyx, yyy\}$

Above, the W-set with the F-set is combined with the Cartesian product. If, as above, the F method was not used, the F method would be combined with the transition cover set and the W set Cartesian product instead. This result is shown below for an understanding of the difference.

$$W = TransitionCoverSet \times Wset = \{x, y, xx, xy, yx, yy, yyx, yyy\} \times \{yy, y, x\} =$$
$$\{xyy, xy, xx, yyy, yy, yx, xxyy, xxy, xxx, xyyy, xyy, xyx, yxyy, yxy, yxx, yyyy, yyy, yyx,$$
$$yyxyy, yyxy, yyxx, yyyyy, yyyy, yyyx\}$$

4.4. Complexity Analysis

Binary encoded S states results in m bits such that $S = 2^m$. The Fourier expansion consists of 2^m terms. The maximum number of transitions in a FSM is $S(S-1)$. The complexity of Fourier transformation is given as $O(2^m)$. The complexity of the whole algorithm can be expressed as $O(2^m + 2^m \cdot S(S-1) \cdot m + S(S-1))$=$O(S + S \cdot S(S-1) \cdot \log_2 S + S(S-1))$=$O(S^3)$.

Complexity is polynomial in terms of the number of states. However, when the values are calculated on the fly, there is a significant increase in performance. In the case studies, the method was up to about 32350 states with an Intel i7 processor and a 16gb ram computer. There was no optimization in the algorithm. Therefore, with more optimization, better performance is possible.

Test execution generally takes more time than test generation. Hence, once the test suite has been created, it is applied to a large number of systems. In the proposed methods, the test execution time is significantly reduced depending on the test suite size even though the test generation time is expected to be longer.

CHAPTER 5

EVALUATIONS

In Chapter 5, there are graphical and statistical evaluations. In making these evaluations, four methods were used. The methods used during the evaluation are UIO Sequence, characterization set, F and F_w methods. These methods were examined in terms of characteristic, cost and effectiveness. Input, output and state parameters are used during the examination. Then, statistical differences were examined. For this, basic statistical information and T-Test were applied.

5.1. Performance Metrics

In this section, methods are studied in terms of characteristic, cost, effectiveness. The number of resets and the length of the test determines the characteristic of the test suite. To calculate the cost, the test suite length has been looked at. To see which method is effective, the error capture rate is checked. These values have examined the interaction with state, input, and output. There are a total of 3 types of configurations used for each one in each chart:

1. input is variable, output 4-valued, state 4-valued
2. output is variable, input 4-valued, state 4-valued
3. state is variable, input 4-valued, output 4-valued

5.2. Generating FSMs

A built-in tool called Genstate was used to create FSM [1] [54]. Genstate has the parameters such as the number of inputs, the number of outputs, and the number of transitions. Genstate also has many different parameters that are not used in this study.

This tool makes deterministic and accessibility controls while creating FSM. This ensures that FSM is deterministic and all states are accessible.

Software usage:

Genstate -i n -o n -t n $[filename]$

If you want to print the output of the program to file, file name must be added to $[filename]$. Genstate produces output in KISS file format.

When it comes to system features, the processor in the system is Intel Core i7 7th generation. Nvidia Geforce 940mx is the graphics card of the system. The memory of the system is 16gb. The programs used in the system are Microsoft Visual Studio 2017 and Windows operating system. Also the program was tested with Ubuntu 18.10 and gcc.

5.3. Analysis Results

5.3.1. Reset Numbers

The target in Figure 5.1 is that the reset remains low while the state increases. When the graph is examined for this purpose, it is seen that the highest reset values belong to UIO method. This is not intended. Furthermore, the reset value of the UIO method continued to increase throughout the graph. The UIO method has reset values in the last state range, well above the other methods, as in the beginning. In this respect, the UIO method has the lowest success rate in all of the reset-state graph compared to other methods.

When we look at the W method, it is seen that the UIO method started to graph at almost the same reset values. In addition, the reset values of the W method continued to increase in all of the graphs similar to the UIO method. However, the W method rise is lower than the UIO method. In the last state range, the W method's reset values are lower than the UIO method. On the other hand, W method's reset values are much higher than the F and F_w methods. In this case, the W method's performance seems to be close to the UIO method. However, the W method has a higher performance compared to the UIO method.

When the F_w method is examined, it is seen that the graph has much lower reset

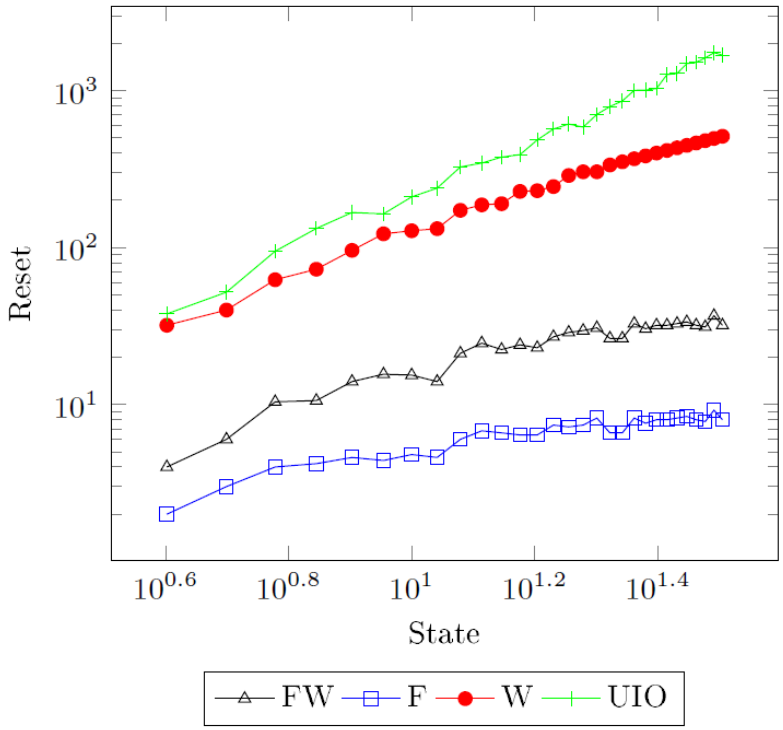

Figure 5.1. State and Reset Relation Representation

values than W and UIO methods. In general, the F_w method 's reset values have increased as other methods have. However, the increase value of the F_w method is lower than the UIO and W methods. On the other hand, the increase value of the F_w method is slightly faster than the F method. At the end of the graph, the reset value of the F_w method is much lower than the UIO and W methods. This trends does'nt change from the beginning to the end of the graph. This is closer to the intended situation. Accordingly, it is possible to say that F_w method gives better results in terms of reset-state than W and UIO method.

The F method has much lower reset values than the other methods at the beginning, and end of the graph. The reset values of the F_w method are close to the F method's reset values. In the graph, there is a significant difference between the F and F_w methods and the W and UIO methods in the maintenance of the reset values. This difference increased further at the end of the graph. The reset values of the F method slightly increased throughout the graph. The highest performance according to the reset-state graph belongs to the F method. After F method, the best performance belongs to F_w method.

In Table 5.1, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F method follows the F_w method in terms of performance

Table 5.1. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

rate. The performance of F and F_w methods yielded better results than W and UIO methods. See Section A.1.1 in Appendix for more information.

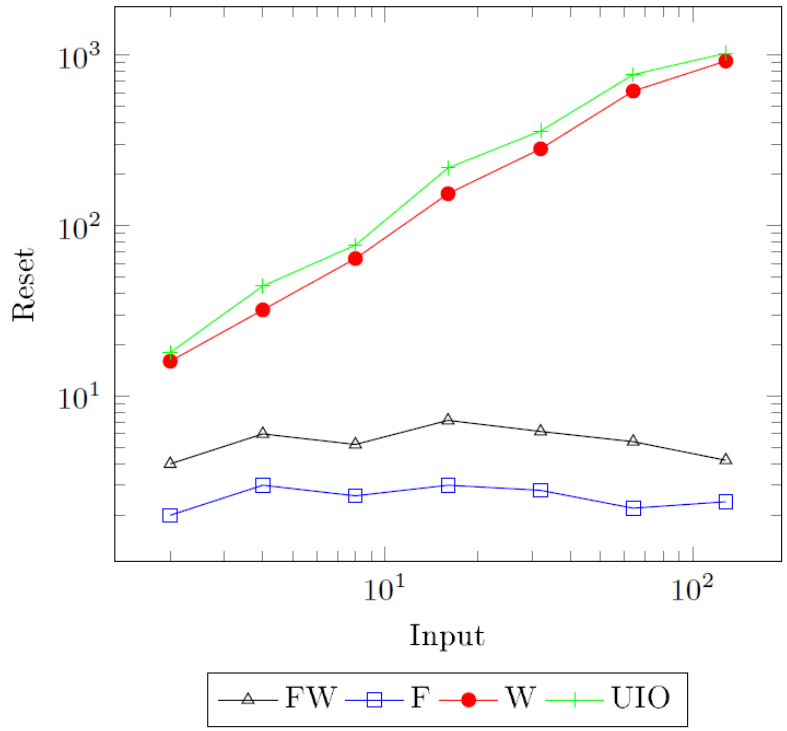

Figure 5.2. Input and Reset Relation Representation

The target in the Figure 5.2 is that the reset remains low while the input increases. When the graph is examined for this purpose, UIO and W methods are close to each other at the beginning of the graph. Then, the reset values of the UIO and W methods continued to rise rapidly in a similar manner. At the end of the graph, the UIO method whose Reset values increased rapidly in all of the graph reached high Reset values. This is far above the intended value. In general, the UIO method is close to the W method, but it has reached higher reset values. In this respect, the lowest performance compared to all other methods belongs to the UIO method.

The W method has similar values to the UIO method from the beginning to the

end of the graph. The values of the W method are similar to those of the UIO method across the graph. However, these values are slightly lower than the UIO method. At the end of the graph, the W method was at the same reset values as the UIO method. In this respect, the W method's performance is very close to the UIO method but slightly higher than the UIO method.

At the beginning of the graph, the F_w method has a much lower reset value than the UIO and W methods. In contrast to the UIO and W methods, the reset values of the F_w method did not increase. It is even possible to say that the values of the F_w method have decreased. In this respect, F_w method performance is higher than UIO and W methods, but it is lower than F method.

The F method has started at the lowest reset values. The graph continued with slight fluctuations. However, it retained the initial reset values throughout the graph. So the F method's reset values did not increase. This is the desired result. The F values of the method are similar to those of the F_w method. However, the F method gave better results than the F_w method. On the other hand, UIO method and W method have much higher values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

Table 5.2. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

In Table 5.2, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. The performance of F and F_w methods in Reset-Input yielded better results than W and UIO methods. See Section A.1.2 in Appendix for more information.

The target in the Figure 5.3 is that the reset remains low while the output increases. When the graph is examined for this purpose, it is seen that the UIO method starts to graph at the highest reset values. In the continuation of the graph, the reset values of the UIO method decreased slowly, then rapidly. In the average output time, the values are balanced. From the middle of the graph, the values of the UIO method have become stable. Until the end of the graph, the UIO method retained the reset values reached in the middle of the graph.

The W method also proceeded in a similar manner to the UIO method. However,

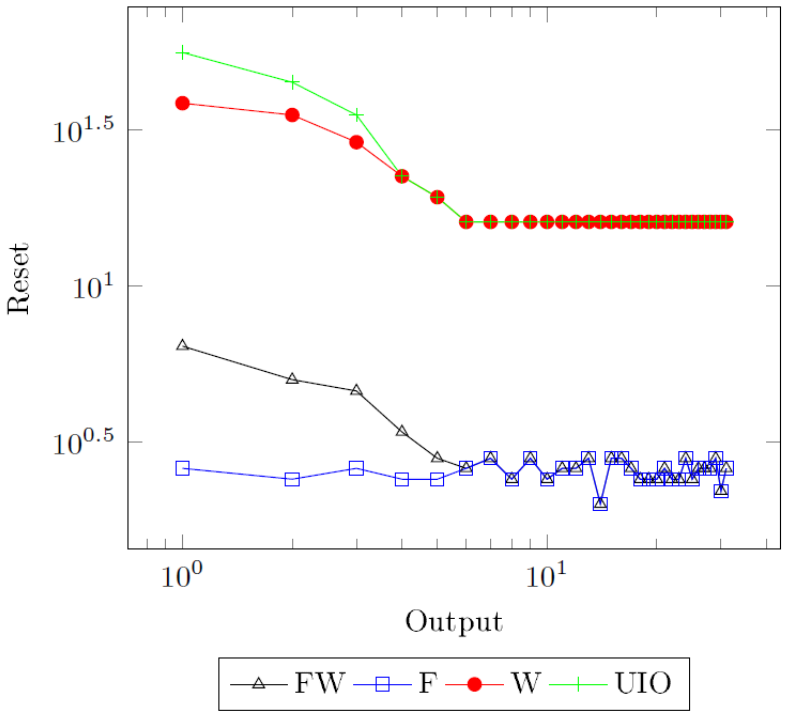

Figure 5.3. Output and Reset Relation Representation

the W method started with a slightly lower reset value than the UIO method. The characteristic of the W method curve is similar to that of the UIO method. Starting from the initial values, the slow and then rapidly falling W method values progressed at the same reset values from the middle of the graph. At the end of the graph, these reset values were maintained. The W method was combined with the same reset values as the UIO method after the first quarter of the graph. Until the end of the graph, the UIO method and W method proceeded at the same reset values.

At the beginning of the graph, the F_w method has a much lower reset value than the UIO and W methods. Similar to the UIO and W methods, the Reset values of the F_w method decreased in the graph. Similar to the UIO and W methods, the F_w method also met the same reset values as the F method in half of the graph. In this respect, F_w method performance is higher than UIO and W methods, but it is lower than F method.

The F method has started at the lowest reset values. In the continuation of the graph, it proceeded with very slight fluctuations. On the whole, however, it retained the initial reset values. That is, the values of the F method have not increased. This is the desired result. The F values of the method are similar to those of the F_w method. However,

the F method gave better results than the F_w method. On the other hand, UIO method and W method have much higher values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

Table 5.3. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

In Table 5.3, the highest performance with respect to T-Test and Hedge's g values belongs to the F_w method. In other words, the performance of F and F_w methods in Reset-Output yielded better results than W and UIO methods. See Section A.1.3 in Appendix for more information.

5.3.2. Average Test Case Length

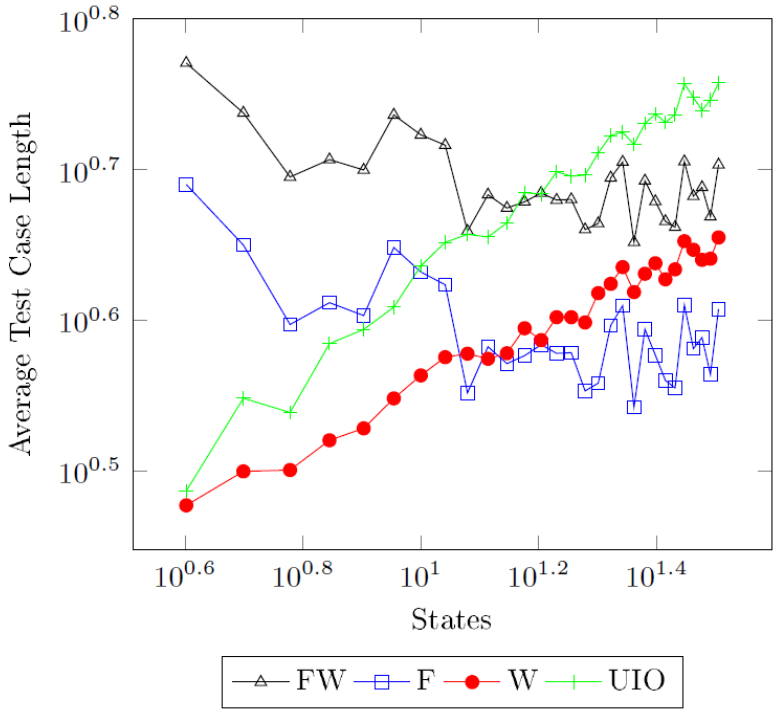

Figure 5.4. States and Average Test Case Length Relation Representation

The goal in the Figure 5.4 is that when the states increase, the Average Test Case Length remains high. Looking at the graph above, it is noteworthy that the UIO and W method started at similar values. On the other hand, the Average Test Case Length values of F and F_w methods are higher than the other methods. The UIO method started at a very low value in contrast to the target in the Average Test Case Length - States graph. However, the UIO method showed a significant increase in the graph and completed the graph at the highest values.

When looking at the W method, it is seen that at the beginning, the UIO is located very close to the method. The W method also started with a very low Average Test Case Length value as UIO method. The W method was increased in the same way as the UIO method. However, the acceleration of the W method is slightly lower than the UIO method. As a result, the W method completed the graph with a lower Average Test Case Length value than the UIO method.

The F_w method at the beginning of the graph is at the highest Average Test Case Length values. In the continuation of the graph, Average Test Case Length values of the F_w method progressed with zigzags. In the average States period, the values started to be more balanced. From the middle of the graph, the values of the F_w method have become more stable. At the end of the graph, the F_w method maintained approximately the reset values reached in the middle of the graph.

The F method was lower than the F_w method, but UIO and W methods were higher than the Average Test Case Length values. The progress of the F method is characteristic of the F_w method. F method has also progressed with zigzags such as F_w method. From half of the graph, the F method values are less fluctuated. At the end of the graph, the F method preserved approximately the reset values reached in the middle of the graph.

Table 5.4. T-Test and Hedge's g results

	W	UIO
F is	same with	lower than
F_w is	higher than	same with

As a result, in Table 5.4, the highest performance according to the T-Test and Hedge's g values belongs to the F_w method. See Section A.2.1 in Appendix for more information.

The target in the Figure 5.5 is that the Average Test Case Length remains high while the input increases. When the graph is examined for this purpose, it is seen that

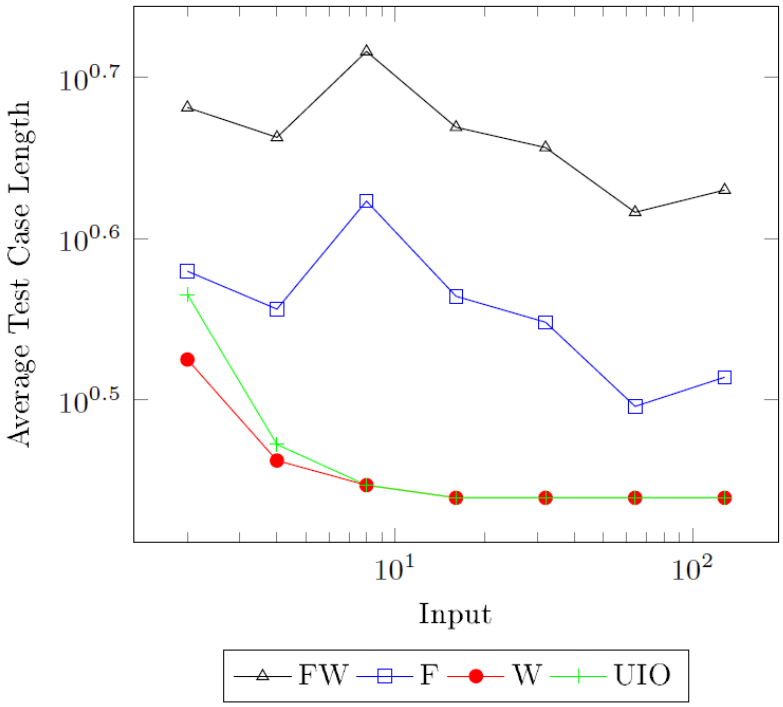

Figure 5.5. Input and Average Test Case Length Relation Representation

UIO and W methods proceed in a similar way. On the other hand, the UIO method, in contrast to its intended use, has remained at low average values. In detail, it is seen that UIO method starts to graph on average values. However, the values of the UIO method have dropped rapidly in a very short Input range. From the first quarter of the graph, the average value of the UIO method is fixed. Until the end of the graph, the UIO method was at low average values.

The W method has started at a lower average than the UIO method. Initially, the W method, which had the lowest average values, dropped rapidly until the first quarter of the graph. The W method has the same values as the UIO method from the first quarter of the graph. Like the UIO method, the W method also retained its value until the end of the graph. Thus, the W method has the lowest performance compared to the graph.

It appears that the F_w method starts graphing at the highest Average Test Case Length. In the continuation of the graph, Average Test Case Length values of the F_w method were progressed by zigzags. The values of the F_w method decreased even slightly in the continuation of the graph. However, the F_w method has achieved the highest values in all of the graphs compared to other methods.

The F method is lower than the F_w method at the beginning of the graph; however,

it has higher Average Test Case Length values than UIO and W methods. The progress of the F method is characteristic of the F_w method. F method has also progressed with zigzags such as F_w method. The values of the F method have fallen, albeit slightly, in the continuation of the graph. At the end of the graph, and in general, the F method has a lower average value than the F_w method but higher than the UIO and W methods. In this regard, the F method has the best performance after the F_w method.

Table 5.5. T-Test and Hedge's g results

	W	UIO
F is	higher than	higher than
F_w is	higher than	higher than

As a result, in Table 5.5, the highest performance F method according to T-Test and Hedge's g values belongs to F_w method. In other words, the performance of F and F_w methods in Average Test Case Length -Input yielded better results than W and UIO methods. See Section A.2.2 in Appendix for more information.

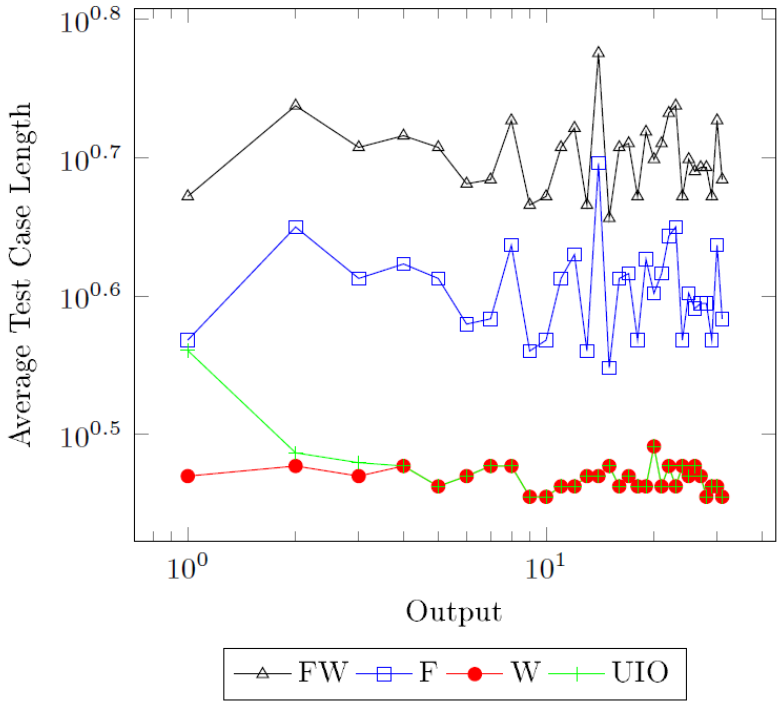

Figure 5.6. Output and Average Test Case Length Relation Representation

In the Figure 5.6, it is aimed to keep the Average Test Case Length high while

the output increases. When the graph is examined for this purpose, the starting point of the UIO method is the same as the F method in average. The average of the UIO method, however, declined rapidly in the first quarter of the graph. At the end of the graph, the UIO method has an average stability. From the first quarter of the graph, the UIO method has progressed to stable values. The UIO method maintained approximately stable progressive average values at the end of the graph.

The W method has the lowest average value at the starting point. The average values of the W method are approximately the same across the entire graph. The W method has stable values except for very small ups and downs. Approximately half of the graph approached the values of the UIO method to the same values as the W method. As a result, from almost half of the graph, the values of the UO method and the W method proceeded in line with each other. At the same time, these values are the lowest average values according to other methods. Therefore, the performance of W and UIO methods is low.

On the other hand, F_w method has the highest values in average and output relation compared to other methods. The F_w method progressed along the graph with zigzags. However, the F_w method also completed the graph at the highest average point. The F_w method showed the average curve over all other methods in the graph. Thus, in terms of the relationship between average and output, the F_w method performed better than other methods.

The F method has lower Average Test Case Length values than the F_w method but higher than the UIO and W methods. The progress of the F method is characteristic of the F_w method. F method has also progressed with zigzags such as F_w method. At the end of the graph, the F method retained the reset values that the graph had overall. Thus, in terms of the average and output relationship, the F method has the best performance after the F_w method.

Table 5.6. T-Test and Hedge's g results

	W	UIO
F is	higher than	higher than
F_w is	higher than	higher than

As a result, in Table 5.6, the highest performance according to T - Test and Hedge's g values belongs to F_w method. The F method follows the F_w method in terms of performance. Therefore, the performance of F_w and F methods in Average Test Case

Length - Output yielded better results than W and UIO methods. See Section A.2.3 in Appendix for more information.

5.3.3. Test Suite Length

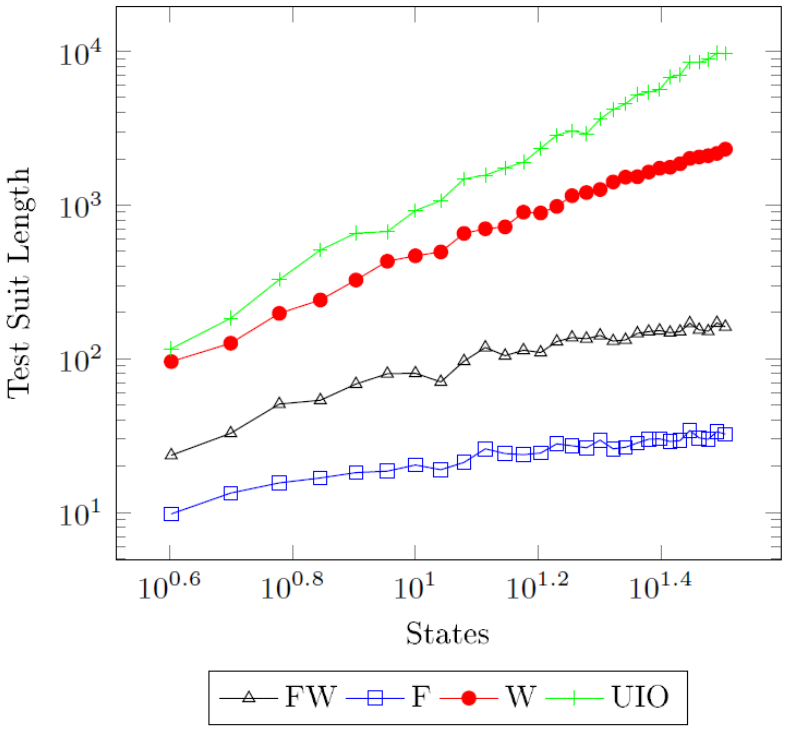

Figure 5.7. State and Test Suite Length Relation Representation

The target in the Figure 5.7 is that the Test suite Length remains low while the state increases. When the graph is examined for this purpose, it is seen that the highest Test Suite Length value belongs to UIO method. This is not intended. Furthermore, the reset value of the UIO method continued to increase in the entire graph. The UIO method has higher Test Suite Length values at the end of the graph than the other methods. In this respect, the UIO method has the lowest performance in the entire Test Suite Length -state graph compared to other methods.

When we look at the W method, it is seen that the UIO method starts to graph almost the same Test Suite Length value. In addition, the x values of the W method continued to rise in the graph as similar to the UIO method. However, the W method rise

is lower than the UIO method. In the last state range, the W method is lower than the UIO method but with the F_w method much higher than the F methods. In this case, the W method's performance seems to be close to the UIO method. However, the W method has a higher performance than the UIO method.

When the F_w method is examined, it is seen that the graph has a much lower Test Suite Length value than the W and UIO methods. In the overall graph, the Test Suite Length value of the F_w method increased as well as other methods. However, the increase value of the F_w method is lower than the UIO and W methods. On the other hand, the increase value of the F_w method is slightly faster than the F method. At the end of the graph, the Test Suite Length values of the F_w method are much lower than the UIO and W methods. According to the F method is high. This order has never changed from the beginning to the end of the graph. This is closer to the intended situation. Accordingly, it is possible to say that the F_w method gives better results in terms of Test Suite Length -state than W and UIO method.

The F method has much lower Test Suite Length values than the other methods at the beginning and end of the graph. The method closest to the Test Suite Length value of the F method belongs to the F_w method. In the graph, there is a significant difference between the F and F_w methods and the W and UIO methods in the maintenance of Test Suite Length values. This difference was further increased at the end of the graph. The Test Suite Length value of the F method was slightly increased until the end of the graph. According to the Test Suite Length -state graph, the highest performance belongs to the F method. F_w method follows F method in terms of performance.

Table 5.7. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

As a result, in Table 5.7 the highest performance according to T-Test and Hedge's g values belongs to F method. The F_w method follows the F method in terms of performance rate. Therefore, the performance of F and F_w methods in Test Suite length - States yielded better results than W and UIO methods. See Section A.3.1 in Appendix for more information.

The target in the Figure 5.8 is that while the input increases, Test Suite Length remains low. When the graph is examined for this purpose, it is seen that UIO and W

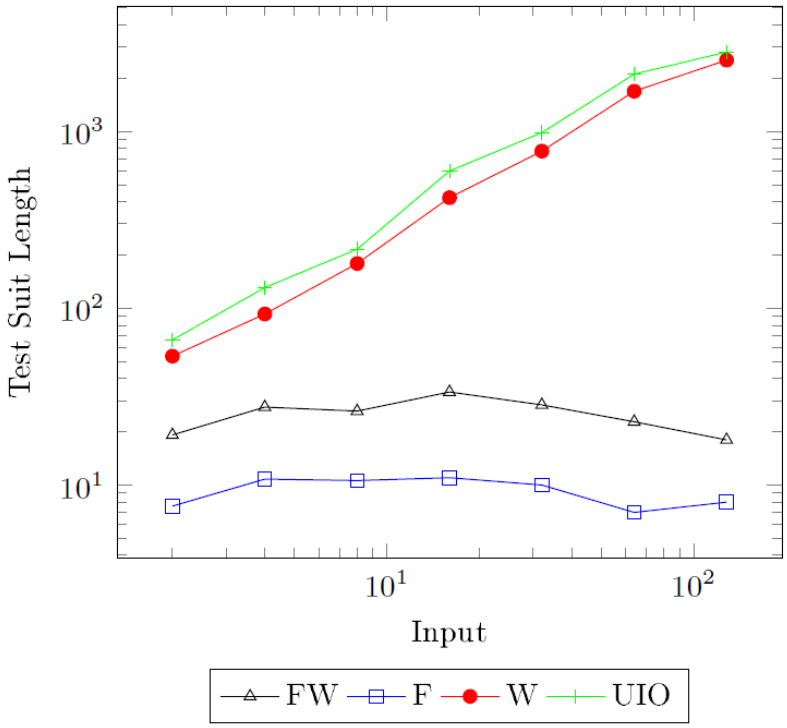

Figure 5.8. Input and Test Suite Length Relation Representation

methods start to graph near each other. Subsequently, the Test Suite Length values of UIO and W methods increased rapidly in a similar manner. At the end of the graph, the UIO method, whose Test Suite Length values increased rapidly, reached very high values. This is far above the intended value. In general, the UIO method reached close to the W method but higher than it reached the Test Suite Length. In this respect, the lowest performance compared to all other methods belongs to the UIO method.

The W method has similar values to the UIO from beginning to end of the graph. In general, the W method has progressed at similar Test Suite Length values with the UIO method. However, W method has lower values than UIO method with little difference. At the end of the graph, the W method was at the same Test Suite Length value as the UIO method. In this respect, the W method's performance is very close to the UIO method but slightly higher than the UIO method.

F_w method, in contrast to UIO and W methods, has started to graph at much lower Test Suite Length values. In contrast to the UIO and W methods, the Test Suite Length values of the F_w method did not increase. It is even possible to say that the decrease. In this respect, F_w method performance is higher than UIO and W methods, but it is lower

than F method.

The F method has started at the lowest Test Suite Length values. The graph continued with slight fluctuations. However, in the entire graph, the initial Test Suite Length value was maintained. That is, the Test Suite Length values of the F method have not increased. This is the desired result. The Test Suite Length values of the F method are similar to those of the F_w method. However, the F method gave better results than the F_w method. On the other hand, UIO method and W method have much higher values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

Table 5.8. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

As a result, in Table 5.8, the highest performance according to T - Test and Hedge's g values belongs to the F method. The F method follows the F_w method in terms of performance rate. Therefore, the performance of F and F_w methods in Test Suite Length - Input yielded better results than W and UIO methods. See Section A.3.2 in Appendix for more information.

Targeted in the Figure 5.9 is that Test Suite Length is low while output increases. When the graph is examined for this purpose, it is seen that UIO method starts to graph at the highest Test Suite Length value. In the continuation of the graph, the Test Suite Length values of the UIO method decreased slowly, then quickly. In the average output time, the values of the UIO method are balanced. From the middle of the graph, the values of the UIO method have become stable. Until the end of the graph, the UIO method retained its Test Suite Length values in the middle of the graph.

W method also showed similar progress in UIO method. However, the W method has started at a slightly lower Test Suite Length value than the UIO method. The characteristic of the W method curve is similar to that of the UIO method. From the start value, the slow, then rapidly falling W method progressed at the same Test Suite Length values from the middle of the graph. At the end of the graph, it maintained its Test Suite Length value. The W method was combined with the same Test Suite Length values as the UIO method after the first quarter of the graph. Until the end of the graph, the UIO method and W method proceeded at the same Test Suite Length values.

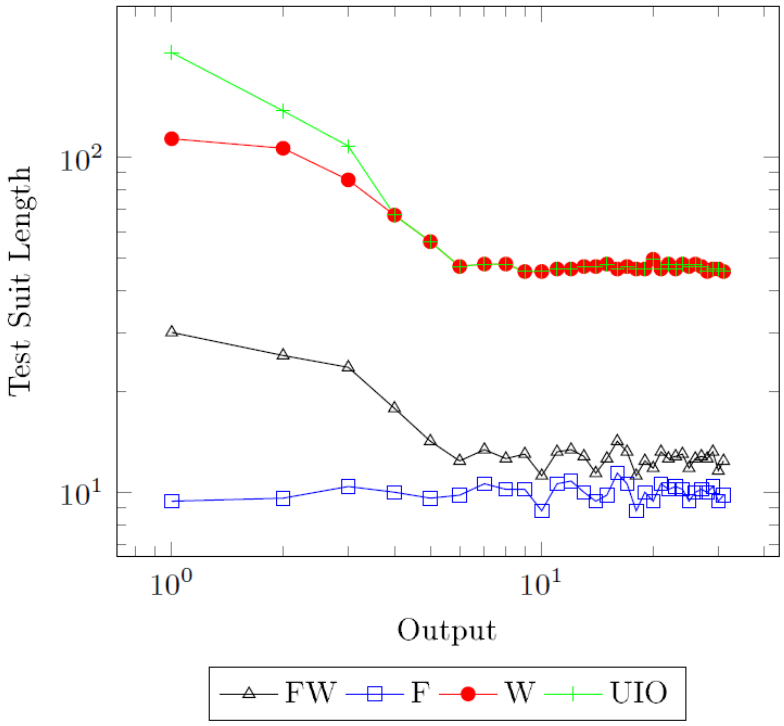

Figure 5.9. Output and Test Suite Length Relation Representation

At the beginning of the graph, the Test Suite Length values of the F_w method are much lower, as opposed to UIO and W methods. Similarly to the UIO and W methods, the Test Suite Length values of the F_w method decreased. Similar to the UIO and W methods, the F_w method met the F method at near Test Suite Length values in the half of the graph. In this respect, F_w method performance is higher than UIO and W methods, but it is lower than F method.

The F method has started at the lowest Test Suite Length values. The chart continued with slight fluctuations. However, it retained the initial Test Suite Length value in the entire graph. That is, the Test Suite Length values of the F method did not show a significant increase. This is the desired result. From half of the graph, the F values of the F method are similar to those of the F_w method. However, the F method gave better results than the F_w method. On the other hand, UIO method and W method have much higher values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

As a result, in Table 5.9, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms

Table 5.9. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

of performance. Therefore, the performance of F and F_w methods in Test Suite Length - Output yielded better results than W and UIO methods. See Section A.3.3 in Appendix for more information.

5.3.4. Fault Detection Ratio

Mutant analysis was used to determine the effectiveness of the method. In mutation analysis, mutants were created using the four categories such as operation, transfer, extra state and missing state errors. Four methodologies run with these mutants and their results were evaluated [32][4][6].

Mutation Analysis is a powerful technique for assessment of the goodness test of tests. It provides a set of strong criteria for test assessment and enhancement. Since the determination of the complete requirement is not possible in the normal way, mutants have been established according to certain criteria. Mutant is the pattern of the model distorted by certain criteria. Tests were run on possible created faults (mutants). The number of mutants caught determines the performance of the test set.

The error model applied for the FSM is shown in Figure 5.10. This fault model consists of four categories:

1. **Operation Error**: Output errors that occur during the transition are entered into this category. This error is shown in the second FSM in the figure. Output 0 instead of 1 in FSM.

2. **Transfer Error**: Errors that occur when changing from one state to another into this category. This error is shown in the third FSM in the figure. The FSM went q0 to q1 instead of q0 to q0.

3. **Extra State Error**: Errors that occur when FSM adds a new state are in this category. This error is shown in the fourth FSM in the figure. FSM added an extra q2 state.

4. **Missing State Error**: If the state is deleted, the errors will occur in this category. This error is shown on the fifth FSM in the figure.

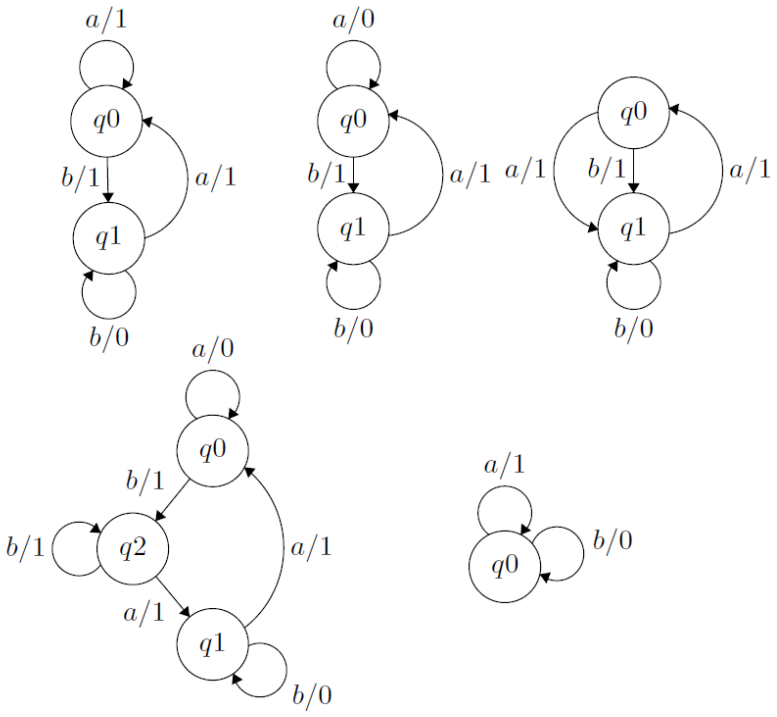

Figure 5.10. Operation Error, Transfer Error, Extra State Error, Missing State Error

In the Figure 5.11, the goal is to keep the Fault Detection Ratio high while the state increases. At the beginning of the graph, all methods are in the same Fault Detection Ratio values. The UIO method initially retained its Fault Detection Ratio value until the end of the graph. Furthermore, no change was observed in the Fault Detection Ratio values of the UIO method in all of the graph.

A similar characteristic is observed with the UIO method. The W method has also retained the value of Fault Detection Ratio, which the UIO method started, to the end of the graph. That is, the Fault Detection Ratio values of the W method remained stable throughout the graph. This shows that the W method has the same performance as the UIO method.

Similarly, the F_w method initially has the same Fault Detection Ratio value as the other methods. However, the F method did not progress with constant values in contrast to UIO and W methods. In general, the Fault Detection Ratio values of the F method fluctuated, albeit slightly. In the graph, the performance of F_w method is similar to UIO and W methods.

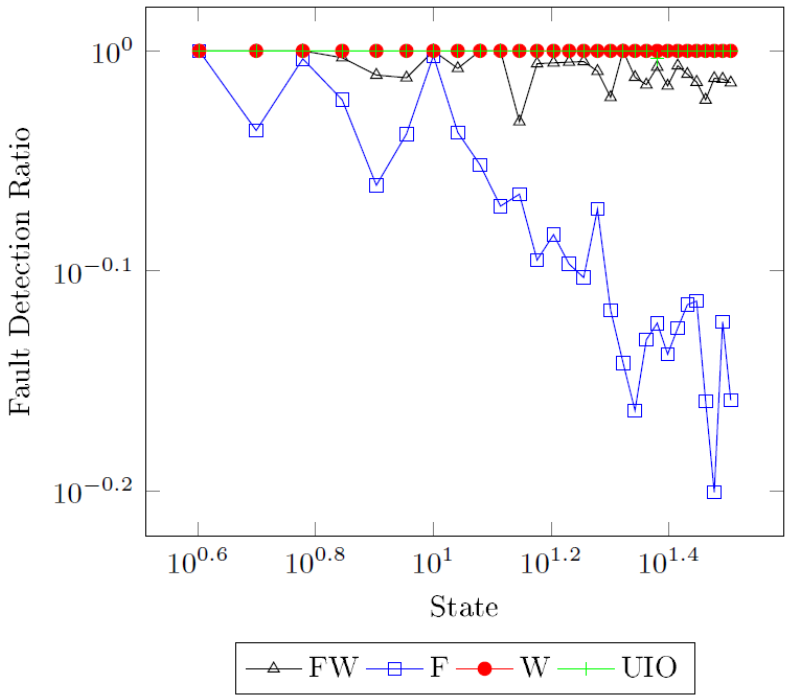

Figure 5.11. State and Fault Detection Ratio Relation Representation

The F method has started to graph from the same point as other methods. However, the F method progressed with zigzags across the graph. The Fault Detection Ratio values of the F method fluctuated. This progression is in the direction of decreasing the Fault Detection Ratio values, albeit slightly. Although the Fault Detection Ratio values of the F method continued to fluctuate in the last quarter of the graph, they made linear progress.

Table 5.10. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	lower than	lower than

As a result, in Table 5.10, the highest performance according to the T - Test and Hedge's g values belongs to the W method. The W method is followed by the UIO method in terms of performance. However, it is necessary to underline that the values of all four methods are very close to each other in terms of Fault Detection Ratio - State values. See Section A.4.1 in Appendix for more information.

In the Figure 5.12, Fault Detection Ratio is expected to remain high while the input increases. When the graph is analyzed, it is noteworthy that all three methods start

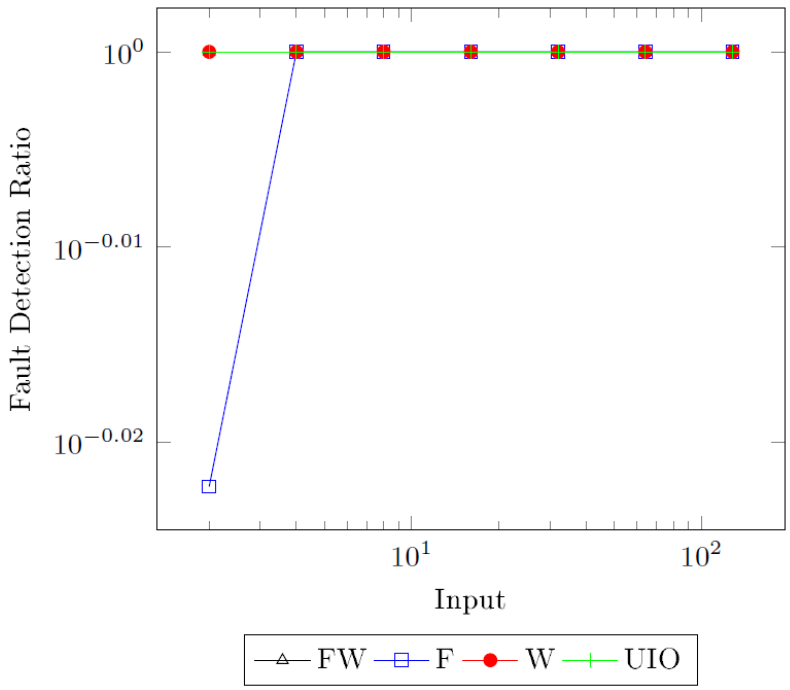

Figure 5.12. Input and Fault Detection Ratio Relation Representation

at similar values. Similarly, it is seen that all three methods are stable from their initial values to the last input range. The UIO method retained its initial Fault Detection Ratio value until the end of the graph. Furthermore, there was no change in the Fault Detection Ratio values of the UIO method in the entire graph.

A similar characteristic is observed with the UIO method. The W method has also retained the value Fault Detection Ratio as the UIO method, until the end of the graph. That is, the Fault Detection Ratio values of the W method remained stable throughout the graph. This shows that the W method has the same performance as the UIO method.

Similarly, the F_w method initially has the same Fault Detection Ratio value as the other methods. Moreover, the F method has progressed with fixed values such as UIO and W methods. The Fault Detection Ratio values of the F method across the graph are stable. In the graph, the F_w method's performance is exactly the same as the UIO and W methods.

The F method, unlike other methods, started with a low Fault Detection Ratio value. However, when approaching the first quarter of the graph, the F method rapidly increased and reached the same Fault Detection Ratio value as the other methods. The Fault Detection Ratio values of the F method have the same characteristics as the other

methods starting from the first quarter. Thus, F_w, W and UIO methods gave the same Fault Detection Ratio -Input result. The F method gave the same Fault Detection Ratio -Input result as the other methods except for the first quarter of the graph.

Table 5.11. T-Test and Hedge's g results

	W	UIO
F is	lower than	lower than
F_w is	same with	same with

As a result, in Table 5.11, we can say that the highest performance belongs to F_w, W and UIO methods. In terms of performance, the F method is slightly behind these three methods. However, it is necessary to underline that the values of all four methods are very close to each other in terms of Fault Detection Ratio - Input values. See Section A.4.2 in Appendix for more information.

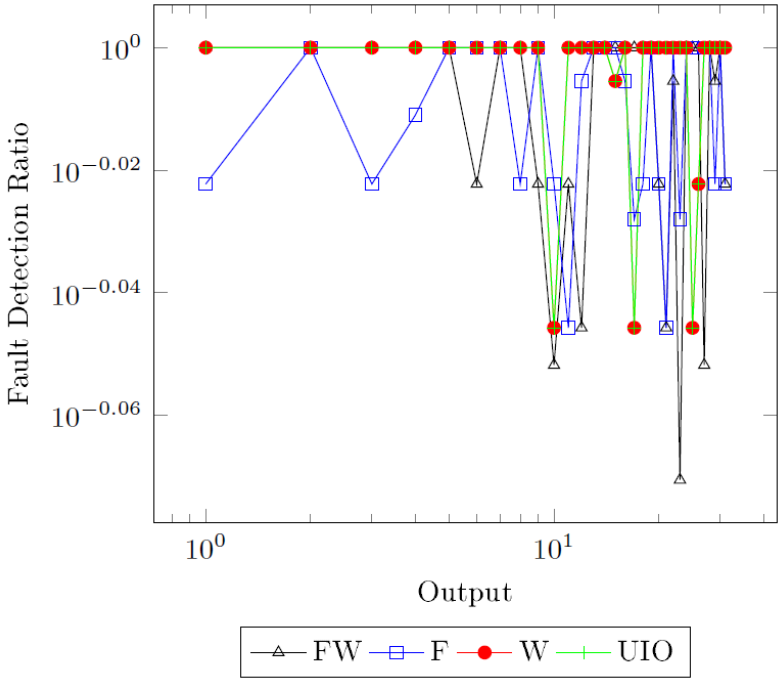

Figure 5.13. Output and Fault Detection Ratio Relation Representation

In the Figure 5.13, Fault Detection Ratio is expected to remain high while the output increases. At the beginning of the graph, all methods except the F method are in the same Fault Detection Ratio values. The UIO method maintained the initial Fault

Detection Ratio value to almost half of the graph. The Fault Detection Ratio values of the UIO method then proceeded with sharp zigzags.

The W method also has similar characteristics with the UIO method. The W method has also retained its initial Fault Detection Ratio values, such as the UIO method, to almost half of the graph. The Fault Detection Ratio values of the W method then proceeded with sharp zigzags. According to the Fault Detection Ratio -output graph, the W method and the UIO method are almost identical.

Similarly, the F_w method initially has the same Fault Detection Ratio value as the other methods. However, the Fault Detection Ratio value of the F method has changed earlier than the UIO and W methods. In the graph, the F method has also progressed with zigzags, like all other methods. In the graph, the performance of F_w method is similar with UIO, W and F methods.

The F method did not start from the same point as the other methods. However, the F method also progressed with zigzags across the graph, like other methods. This fluctuation occurred from the beginning of the graph, unlike other methods. In the last quarter of the graph, very similar ups and downs were observed in all methods. Hence, all methods have similar performances compared to the Fault Detection Ratio -output graph.

Table 5.12. T-Test and Hedge's g results

	W	UIO
F is	same with	same with
F_w is	same with	same with

As a result, in Table 5.12 the performances of the F_w and UIO methods are the same. According to the Fault Detection Ratio - Output graph, we can say that the performance of the four methods is exactly the same. See Section A.4.3 in Appendix for more information.

5.3.5. Killed Mutant / Test Suite Size

The target in the Figure 5.14 is that the Killed Mutant / Test Suite Size remains high while the states increase. When the graph is examined for this purpose, it is seen that UIO method starts to graph at the lowest Killed Mutant / Test Suite Size value. In the continuation of the graph, the Killed Mutant / Test Suite Size values of the UIO method

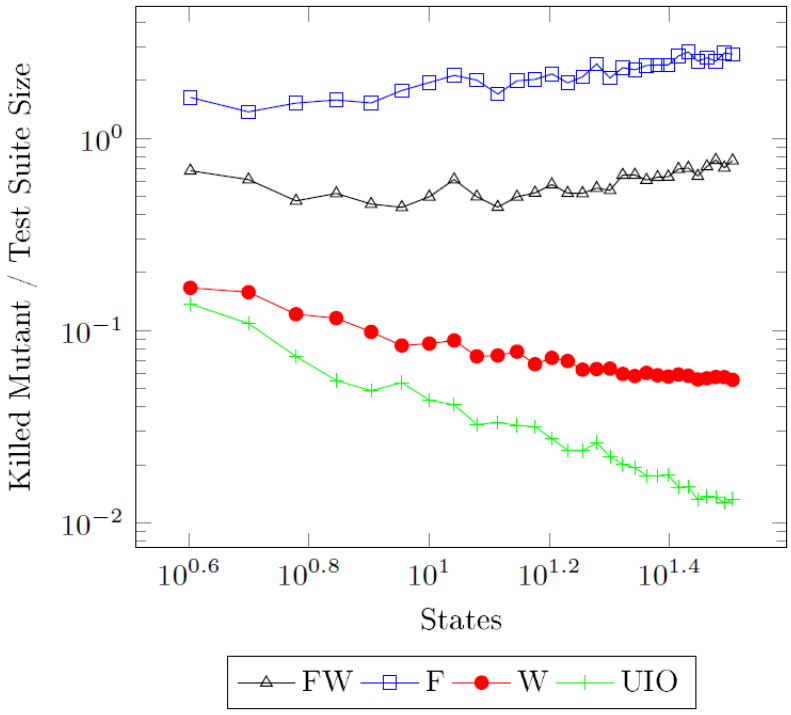

Figure 5.14. State and Killed Mutant / Test Suite Size Relation Representation

decreased regularly. This decline continued until the end of the graph. The decrease in the Killed Mutant / Test Suite Size values of the UIO method is quite fast. Thus, the UIO method has the lowest performance among the methods in terms of Killed Mutant / Test Suite Size -State.

W method also showed similar progress in UIO method. However, at the beginning of the graph, W method is found at slightly higher Killed Mutant / Test Suite Size values than UIO method. The characteristic of the W method curve is similar to that of the UIO method. In the continuation of the graph, the Killed Mutant / Test Suite Size values of the W method dropped regularly, such as the UIO method. However, the decrease in the Killed Mutant / Test Suite Size values of the W method was slower than the UIO method.

At the beginning of the graph, the F_w method is found at higher Killed Mutant / Test Suite Size values than the UIO and W methods. In the rest of the graph, the Killed Mutant / Test Suite Size values of the F_w method did not decrease, as opposed to UIO and W methods. The F_w method has almost the same Killed Mutant / Test Suite Size values in the entire graph. In this respect, F_w method performance is higher than UIO and W methods, but it is lower than F method.

The F method started at the highest Killed Mutant / Test Suite Size values. The F method proceeded in the graph with very slight fluctuations. On the other hand, the Killed Mutant / Test Suite Size values of the F method increased in the continuation of the graph. In other words, F method has the highest Killed Mutant / Test Suite Size values in the graph. This is the desired result. The Killed Mutant / Test Suite Size values of the F method are similar to those of the F_w method. However, the F method gave better results than the F_w method. On the other hand, UIO method and W method have much lower values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

Table 5.13. T-Test and Hedge's g results

	W	UIO
F is	higher than	higher than
F_w is	higher than	higher than

As a result, in Table 5.13, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. Therefore, the performance of F and F_w methods in Killed Mutant / Test Suite Size - State yielded much better results than W and UIO methods. See Section A.5.1 in Appendix for more information.

In the Figure 5.15, the target is that the Killed Mutant / Test Suite Size values remain high while the input increases. When the graph is examined for this purpose, it is seen that initially UIO and W methods are close to each other. Although there are similarities between UIO and W method, the UIO method has the lowest Killed Mutant / Test Suite Size values. In the continuation of the graph, the Killed Mutant / Test Suite Size values of UIO and W methods were not increased. In this respect, the UIO method has the lowest performance compared to the graph.

The W method has similar values to the UIO method from the beginning to the end of the graph. The W method, which proceeds at similar Killed Mutant / Test Suite Size values with the UIO method throughout the graph, has higher values than the UIO method with little difference. At the end of the graph, the W method has the same Killed Mutant / Test Suite Size value as the UIO method. In this respect, the W method's performance is very close to the UIO method but slightly higher than the UIO method.

At the beginning of the graph, the F_w method has a higher Killed Mutant / Test Suite Size value than the UIO and W methods. In contrast to the UIO and W methods,

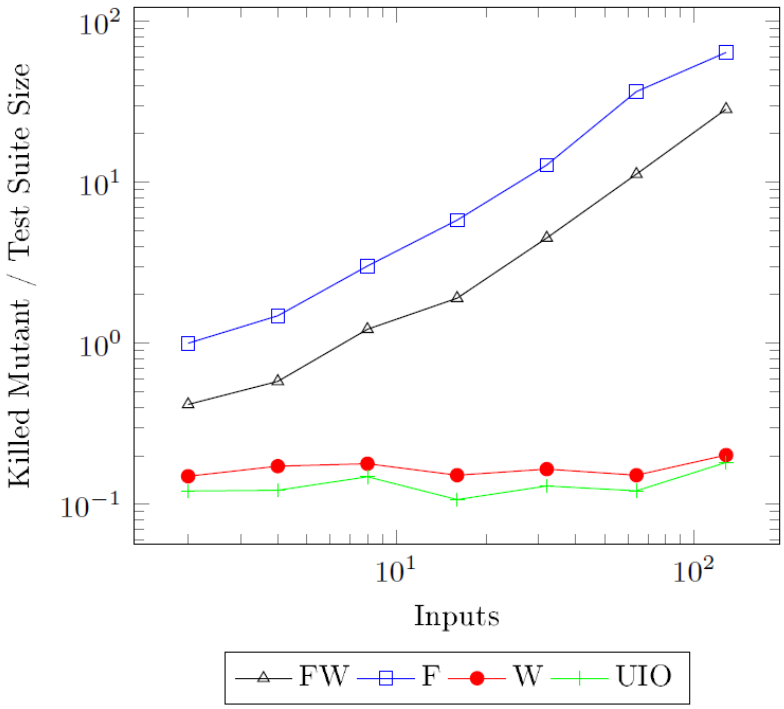

Figure 5.15. Input and Killed Mutant / Test Suite Size Relation Representation

the Killed Mutant / Test Suite Size values of the F_w method continued to increase. The Killed Mutant / Test Suite Size values of the F_w method increased rapidly and regularly throughout the graph. In this respect, F_w method performance is higher than UIO and W methods, but it is lower than F method.

The F method started at the highest Killed Mutant / Test Suite Size level. Similar to the F_w method, the F method's Killed Mutant / Test Suite Size values continued to increase. The Killed Mutant / Test Suite Size values of the F_w method increased rapidly and regularly throughout the graph. This is the desired result. The Killed Mutant / Test Suite Size values of the F method are similar to those of the F_w method. However, the F method gave better results than the F_w method. On the other hand, UIO method and W method have much lower values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

Table 5.14. T-Test and Hedge's g results

	W	UIO
F is	higher than	higher than
F_w is	higher than	higher than

As a result, in Table 5.14, the performance of the four methods can be considered the same by T-Test. See Section A.5.2 in Appendix for more information.

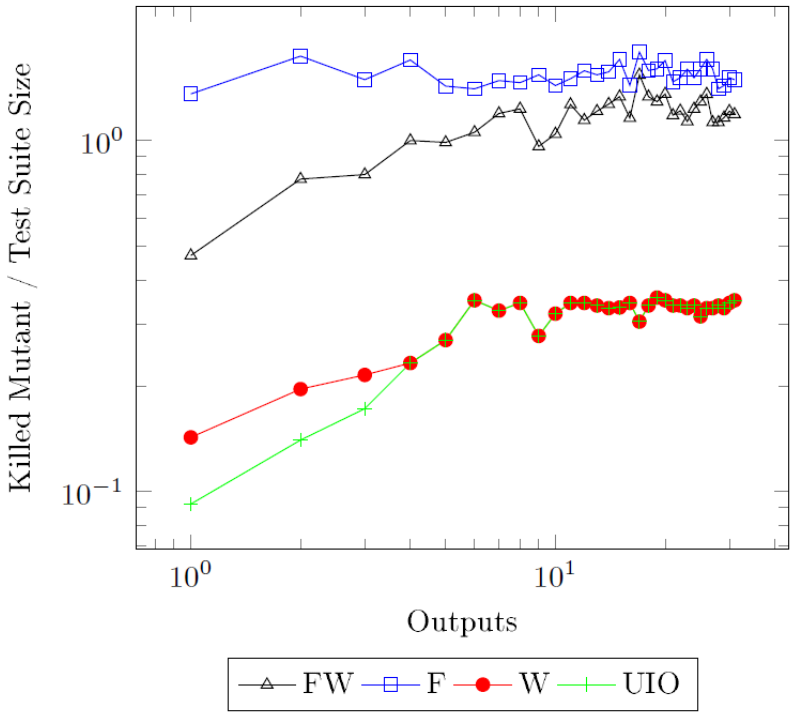

Figure 5.16. Output and Killed Mutant / Test Suite Size Relation Representation

In the Figure 5.16, the target is that the Killed Mutant / Test Suite Size remains at high values while the output increases. When the graph is examined for this purpose, it is seen that the UIO method starts to graph the lowest Killed Mutant / Test Suite Size. In the continuation of the graph, the Killed Mutant / Test Suite Size values of the UIO method increased rapidly first. It has stabilized from half of the graph. In other words, the values of the UIO method are balanced in half of the graph. Until the end of the graph, the UIO method retained the Killed Mutant / Test Suite Size values reached in the middle of the graph.

The W method also proceeded in a similar manner to the UIO method. However, at the beginning of the graph, W method is found at Killed Mutant / Test Suite Size values slightly higher than UIO method. The characteristic of the W method curve is similar to that of the UIO method. After the initial value, the lightweight W method, which increased rapidly, proceeded at the same Killed Mutant / Test Suite Size value from the middle of the graph. At the end of the graph, this Killed Mutant / Test Suite Size value was maintained. The W method has been combined with the same Killed Mutant / Test

Suite Size values as the UIO method from almost half of the graph. The UIO method and W method proceeded at the same Killed Mutant / Test Suite Size values until the end of the graph.

In contrast to UIO and W methods, the F_w method has started to graph much higher Killed Mutant / Test Suite Size. Similar to the UIO and W methods, the F_w method's Killed Mutant / Test Suite Size values increased. Similar to the UIO and W methods, the F_w method also met the F method at close Killed Mutant / Test Suite Size with half of the graph. In this respect, F_w method performance is higher than UIO and W methods, but it is lower than F method.

The F method started at the highest Killed Mutant / Test Suite Size values. The graph continued with slight fluctuations. However, in the entire graph, the initial Killed Mutant / Test Suite Size value was maintained. That is, the T method values of the F method proceeded approximately stable. This is the desired result. The Killed Mutant / Test Suite Size values of the F method are similar to those of the F_w method. However, the F method gave better results than the F_w method. On the other hand, UIO method and W method have much lower values than F method and F_w method. In this case, it is seen that the highest performance belongs to F method.

Table 5.15. T-Test and Hedge's g results

	W	UIO
F is	higher than	higher than
F_w is	higher than	higher than

As a result, in Table 5.15, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. Therefore, the performance of F and F_w methods in Killed Mutant / Test Suite Size - Output yielded much better results than W and UIO methods. See Section A.5.3 in Appendix for more information.

CHAPTER 6

THREATS TO VALIDITY

FSMs were randomly generated using certain criteria to prevent the results from being biased. These criteria were created by changing the state, output and input numbers of the FSM separately. Thus, it is examined how the algorithm produces the results according to the change of the criteria. The investigated features are the average length of test cases, length of test suites, fault detection ratio, reset amount and performance.

The number of randomly generated FSMs is about 640 for 15 graphs. Looking at the literature, this number seems to be sufficient for research. Algorithms were running four times to test the graphs. That is, the study has been investigated with about 2560 random FSMs in order to generalize the results.

While analyzing the subject, many similar academic papers have been investigated. Methods of these academic papers were adopted in the study. The algorithm created within the scope of the research is examined in light of these methods. In addition, the algorithm is written by a programmer who has known the subject, so the error rate of the program has been tried to be reduced. After the program has been written, the program has been run with manually created FSMs during the test phase. Thus, the program was checked for consistency with the calculated results.

W and UIO algorithms are used for comparison. Therefore, the result of the created algorithm is limited by these two algorithms compared. However, there are many algorithms in this area. Since it is not possible to compare with all of the existing algorithms, the most basic algorithms have been chosen for comparison. Most of the methods come from the W UIO method. What's more, there is little difference between W UIO and other methods.

This study focuses on output testing. The state verification part is taken from W method in F_w. Hence, verification part in W or UIO is little effect on test suite. For example, in the state verification part of the UIO, the test cases are parallel to the state number. However, Using fourier transform of boolean function instead of State verification part in W, UIO, the transitions may be reduced. This need to be checked.

The recommended method is determining the paths that most affect the outcome.

However, it should not be forgotten that the importance of a path that has low affect the outcome may be high.

In this study, it is recommended to reduce a test suite in accordance with the specific target. However, there may be cases where this reduction in the test suite cannot yield the desired result. In other words, the concept of ”the path that affects the function most is important” may not be valid in all cases. On the other hand, the deficiencies of the system can be eliminated by means of specific addons that can be added to the proposed system. With the parts taken from other models, the proposed method can be improved. Thus, better results can be achieved.

As a result, the proposed method uses the specific approach. Therefore, although there may cases where the approach is not covered, it can possible to say that the developed method has achieved the desired success.

CHAPTER 7

CONCLUSIONS

In this study, two new test production methods F and F_w have been proposed. The proposed test creation methods are based on the Fourier analysis of Boolean functions, unlike other test generation methods. With the Fourier analysis of Boolean functions, discrete structures were converted into polynomials. Thus, how much input combinations affect output can be found.

The proposed methods differ in terms of the points tested. The F method only tests outputs; The F_w method also tests the next state with the outputs. In this context, the proposed methods are compared with UIO and W methods in terms of "characteristic, cost, fault detection ratio and performance". The results were analyzed by T-Test and Hedges' g.

As a result, the highest of performance per test was determined as F method. The F method for performance per test is followed by F_w, W and UIO methods, respectively. According to the results, the F_w method has a higher performance than the F method in terms of fault detection ratio. The fault detection ratio of the F_w method is very close to the W and UIO methods.

As a result, with the proposed methods, the test suite length is shortened significantly and high performance have been obtained.

REFERENCES

[1] KISS Generator 0.8. https://ddd.fit.cvut.cz/prj/Circ_Gen/index.php?page=kiss, Accessed: 2019-04-03.

[2] Mahsa Abbasian. On efficiency and effectiveness of model-based test case generation techniques by applying the HIS method. 2016.

[3] Andris Ambainis. Polynomial degree vs. quantum query complexity. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 230–239. IEEE, 2003.

[4] Fevzi Belli and Mutlu Beyazit. A formal framework for mutation testing. pages 121–130, 2010.

[5] Fevzi Belli, Mutlu Beyazıt, Andre Takeshi Endo, Aditya Mathur, and Adenilso Simao. Fault domain-based testing in imperfect situations: a heuristic approach and case studies. *Software Quality Journal*, 23(3):423–452, 2015.

[6] Fevzi Belli, Mutlu Beyazit, Tomohiko Takagi, and Zengo Furukawa. Model-based mutation testing using pushdown automata. *IEICE TRANSACTIONS on Information and Systems*, 95(9):2211–2218, 2012.

[7] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of computer and system sciences*, 47(3):549–595, 1993.

[8] Aline Bonami. Ensembles λ (p) dans le dual de d. *Ann. Inst. Fourier*, 18(2):193–204, 1968.

[9] Bengt ve Katoen Joost-Pieter ve Leucker Martin ve Pretschner Alexander Broy, Manfred ve Jonsson. *Model-Based Testing of Reactive Systems*. 2005.

[10] B.S.Ainapure. *Software testing and quality assurance*. 2009.

[11] Tsun S. Chow. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering*, SE-4(3):178–187, may 1978.

[12] Paulo Cesar ve Simao Adenilso Damasceno, Carlos Diego Nascimento ve Masiero. Evaluating test characteristics and effectiveness of fsm-based testing methods on rbac systems. *Proceedings of the 30th Brazilian Symposium on Software Engineering - SBES '16*, 2016.

[13] Ronald De Wolf. A brief introduction to fourier analysis on the boolean cube. *Theory of Computing, Graduate Surveys*, 1(1-20):15, 2008.

[14] Khaled ve Maag Stephane ve Cavalli Ana R. ve Yevtushenko Nina Dorofeeva, Rita ve El-Fakih. Fsm-based conformance testing methods: A survey annotated with experimental evaluation. *Information ve Software Technology*, 2010.

[15] Margarita Dorofeeva and Irina Koufareva. Novel modification of the w-method. *Bulletin of the Novosibirsk Computing Center. Series: Computer Science*, (18):69–80, 2002.

[16] R Dorofeeva, K El-Fakih, and N Yevtushenko. An improved fsm-based conformance testing method. In *Proc. of the IFIP 25th International Conference on Formal Methods for Networked and Distributed Systems*, pages 204–218.

[17] Rita Dorofeeva, Khaled El-Fakih, and Nina Yevtushenko. An improved conformance testing method. In *International Conference on Formal Techniques for Networked and Distributed Systems*, pages 204–218. Springer, 2005.

[18] Adenilso Endo, Andre Takeshi ve Simao. Evaluating test suite characteristics, cost, and effectiveness of fsm-based testing methods. *Information and Software Technology*, 2013.

[19] A.T. Endo and A. Simao. Experimental comparison of test case generation methods for finite state machines. *Proceedings - IEEE 5th International Conference on Software Testing, Verification and Validation, ICST 2012*, pages 549–558, 2012.

[20] Nathan Jacob Fine. On the walsh functions. *Transactions of the American Mathematical Society*, 65(3):372–414, 1949.

[21] Vanderson Hafemann Fragal, Adenilso Simao, Mohammad Reza Mousavi, and Uraz Cengiz Turker. Extending HSI Test Generation Method for Software Product Lines. *The Computer Journal*, 2018.

[22] Susumu Fujiwara, G v Bochmann, Ferhat Khendek, Mokhtar Amalou, and Abderrazak Ghedamsi. Test selection based on finite state models. *IEEE Transactions on software engineering*, 17(6):591–603, 1991.

[23] Angelo Gargantini. *4 Conformance Testing*. Springer, 2005.

[24] Arthur Gill. *Introduction to the theory of finite-state machines.* McGraw-Hill, New York, 1962.

[25] Solomon Golomb. On the classification of boolean functions. *IRE transactions on circuit theory*, 6(5):176–186, 1959.

[26] Guney Gonenc. A method for the design of fault detection experiments. *IEEE transactions on Computers*, 100(6):551–558, 1970.

[27] Antonio Grasselli and Fabrizio Luccio. A method for minimizing the number of internal states in incompletely specified sequential networks. *IEEE Transactions on Electronic Computers*, (3):350–359, 1965.

[28] FC Hennine. Fault detecting experiments for sequential circuits. In *Switching Circuit Theory and Logical Design, 1964 Proceedings of the Fifth Annual Symposium on*, pages 95–110. IEEE, 1964.

[29] Robert M Hierons. Testing from a nondeterministic finite state machine using adaptive state counting. *IEEE Transactions on Computers*, 53(10):1330–1342, 2004.

[30] Robert M. Hierons, Paul Krause, Gerald Lüttgen, Anthony J. H. Simons, Sergiy Vilkomir, Martin R. Woodward, Hussein Zedan, Kirill Bogdanov, Jonathan P. Bowen, Rance Cleaveland, John Derrick, Jeremy Dick, Marian Gheorghe, Mark Harman, and Kalpesh Kapoor. Using formal specifications to support testing. *ACM Computing Surveys*, 41(2):1–76, feb 2009.

[31] Robert M Hierons and Hasan Ural. Optimizing the length of checking sequences. *IEEE Transactions on Computers*, 55(5):618–629, 2006.

[32] Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE transactions on software engineering*, 37(5):649–678, 2011.

[33] Jeff Kahn, Gil Kalai, and Nathan Linial. *The influence of variables on Boolean functions*. IEEE, 1988.

[34] MG Karpovsky. Finite orthogonal series in the design of digital devices (monograph), 1976.

[35] Tali Kaufman, Simon Litsyn, and Ning Xie. Breaking the e-soundness bound of the linearity test over gf (2). *SIAM Journal on Computing*, 39(5):1988–2003, 2010.

[36] Konrad Kiener. *Uber Produkte von quadratisch integrierbaren Funktionen endlicher Vielfalt*. PhD thesis, Dissertation, Universität Innsbruck, 1969.

[37] R. Lai. A survey of communication protocol testing. *Journal of Systems and Software*, 62(1):21–46, may 2002.

[38] Robert J Lechner. Harmonic analysis of switching functions. In *Recent developments in switching theory*, pages 121–228. Elsevier, 1971.

[39] Robert Joseph Lechner. *Affine equivalence of switching functions*. PhD thesis, Harvard University, 1963.

[40] Mihalis Lee, David ve Yannakakis. Principles ve methods of testing finite state machines, 1996.

[41] Gang Luo, Alexandre Petrenko, and Gregor v Bochmann. Selecting test sequences for partially-specified nondeterministic finite state machines. In *Protocol Test Systems*, pages 95–110. Springer, 1995.

[42] Aditya P. Mathur. *Foundations of software testing*. 2005.

[43] DE Muller. Boolean algebras in electric circuit design. *The American Mathematical Monthly*, 61(7):27–28, 1954.

[44] Sachio Naito and Masahiro Tsunoyama. Fault detection for sequential machines by transitions tours. 1981.

[45] Akira Nakashima. The theory of relay circuits composition. *Nippon Electrical Communication Engineers*, (3), 1936.

[46] Ichizo Ninomiya. A theory of the coordinate representation of switching functions. 1960.

[47] Noam Nisan and Avi Wigderson. On rank vs. communication complexity. *Combinatorica*, 15(4):557–565, 1995.

[48] Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

[49] REAC Paley. A remarkable series of orthogonal functions. *Proc. London Math. Soc*, 34:241–264, 1931.

[50] Jorge M Pena and Arlindo L Oliveira. A new algorithm for exact reduction of incompletely specified finite state machines. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 18(11):1619–1632, 1999.

[51] Alexandre Petrenko. Nondeterministic state machine in protocol conformance testing. *Protocol Test Systems*, pages 363–378, 1994.

[52] Alexandre Petrenko and Nina Yevtushenko. Testing from partial deterministic fsm specifications. *IEEE Transactions on Computers*, 54(9):1154–1165, 2005.

[53] Alexandre Petrenko and Nina Yevtushenko. Adaptive testing of nondeterministic systems with fsm. In *High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on*, pages 224–228. IEEE, 2014.

[54] Codrin Pruteanu and Cristian-gyözö Haba. GenFSM: A finite state machine generation tool. *Proc. 9th Int. Conf. Dev. Applicat. Syst*, pages 165–168, 2008.

[55] Ali Rezaki and Hasan Ural. Construction of checking sequences based on characterization sets. *Computer Communications*, 18(12):911–920, 1995.

[56] Klaus F Roth. On certain sets of integers. *Journal of the London Mathematical Society*, 1(1):104–109, 1953.

[57] Krishan Sabnani and Anton Dahbura. A protocol test generation procedure. *Computer Networks and ISDN systems*, 15(4):285–297, 1988.

[58] Sven Sandberg. 1 homing and synchronizing sequences. In *Model-based testing of reactive systems*, pages 5–33. Springer, 2005.

[59] Claude E Shannon. A symbolic analysis of relay and switching circuits. *Electrical Engineering*, 57(12):713–723, 1938.

[60] VI Shestakov. Some mathematical methods for construction and simplificaton of two terminal electrical networks of class a. *Dissertation, Lomonosov State Univ., Moscow*, 1938.

[61] Deepinder P. Sidhu and T-K Leung. Formal methods for protocol testing: A detailed study. *IEEE transactions on software engineering*, 15(4):413–426, 1989.

[62] Adenilso Simao and Alexandre Petrenko. Checking completeness of tests for finite state machines. *IEEE Transactions on Computers*, 59(8):1023–1032, 2010.

[63] Adenilso Simão, Alexandre Petrenko, and Nina Yevtushenko. Generating reduced tests for fsms with extra states. In *Testing of Software and Communication Systems*, pages 129–145. Springer, 2009.

[64] Sam Spiro. Fourier analysis of boolean functions. 2014.

[65] H. Ural. Formal methods for test sequence generation, 1992.

[66] MP Vasilevskii. Failure diagnosis of automata. *Cybernetics*, 9(4):653–665, 1973.

[67] Naum Vilenkin. On a class of complete orthonormal systems. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya*, 11(4):363–400, 1947.

[68] Son T Vuong. The uiov-method for protocol test sequence generation. In *Proc. 2nd IFIP Int. Workshop on Protocol Test Systems (IWPTS'89)*, pages 161–175, 1989.

[69] Joseph L Walsh. A closed set of normal orthogonal functions. *American Journal of Mathematics*, 45(1):5–24, 1923.

[70] N Yevtushenko and A Petrenko. Test derivation method for an arbitrary deterministic automaton, automatic control and computer sciences, 1990.

[71] Fan Zhang and To-yat Cheung. Optimal transfer trees and distinguishing trees for testing observable nondeterministic finite-state machines. *IEEE Transactions on Software Engineering*, 29(1):1–14, 2003.

APPENDIX A

STATISTICAL EVALUATION

The equality of variance is examined before starting the T-Test. For this, Levene's Test for Equality of Variances is used. Here, the null hypothesis refers to the equality of variances. The alternative hypothesis is that the variances are not equal. If the null hypothesis is accepted, the first line of the T-Test is checked. If the null hypothesis is not accepted, the second line is looked at. In order to accept the null hypothesis, the significance level should be greater than 0.05. Otherwise, the alternative hypothesis is accepted.

After looking at Levene's Test for Equality of Variances, it is understood which line should be directed. Once the line is checked, the "t-test for Equality of Means" section of that line is examined.

In the T-Test, the null hypothesis is that the averages are equal. The alternative hypothesis is that the averages are different. If the sig. (2-tailed) value is greater than 0.05, the null hypothesis is accepted. If the value of Sig. (2-tailed) is less than 0.05, the Alternative hypothesis is accepted.

If the alternative hypothesis is accepted, the Hedges' g value is activated. Hedges' g value indicates how different averages are. Hedges' g is about 0.2, which is too small to be seen with the naked eye. Hedges' g is around 0.5, which means that there is a moderate difference. Hedges' g value greater than 0.8 indicates the differences that can easily be seen with the naked eye.

A.1. Reset Numbers

A.1.1. Reset and State

According to Table A.1, the average value of W method is 273,213. The average value of the F method is 6,503. The average F_w method is 24,131. The average of UIO

Table A.1. General Statistics

	Mean	N	Std. Deviation
W	273,213	29	150,349
F	6,503	29	1,817
F_w	24,131	29	9,112
UIO	717,124	29	544,305

method is 717,124. According to these results, the highest average value belongs to UIO method. The lowest average value belongs to the F method. Moreover, there is a huge difference between the highest average and the lowest average. Thus, the UIO method has the lowest performance. The F method has clearly the highest performance.

Table A.2. T-Test for W and F methods at Reset-State

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	94,258	0	9,552	56	0	266,710	27,921	210,777	322,643
Equal variances not assumed			9,552	28	0	266,710	27,921	209,516	323,903

When examined closely, the T-Test results of the F method and the W method showed that there was a significant difference between the two methods (Table A.2). On the other hand, Hedge's g value (2.508) also supports this difference. As a result of this difference, the F method has a significantly higher performance than the W method.

Table A.3. T-Test for UIO and F methods at Reset-State

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	81,340	0	7,030	56	0	710,620	101,075	508,141	913,099
Equal variances not assumed			7,030	28	0	710,620	101,075	503,576	917,664

When we look at the T-Test values of the UIO method with the F method, there is also a significant difference between the two methods(Table A.3). Hedge's g value

(1.846), on the other hand, supports this difference. As a result of this difference, the F method's performance is significantly higher than the UIO method.

Table A.4. T-Test for F_w and F methods at Reset-State

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	41,491	0	10,215	56	0	17,627	1,725	14,170	21,084
Equal variances not assumed			10,215	30,225	0	17,627	1,725	14,104	21,150

When the T-Test values of the F_w method were examined by F method, a significant difference was found between the two methods (Table A.4). On the other hand, Hedge's g value (2.683) also supports this difference. As a result of this difference, F method performance is significantly higher than the F_w method.

Table A.5. T-Test for W and F_w methods at Reset-State

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	85,195	0	-8,905	56	0	-249,082	27,970	-305,114	-193,051
Equal variances not assumed			-8,905	28,205	0	-249,082	27,970	-306,358	-191,806

A significant difference was also found between the T-Test values of the W and F_w methods (Table A.5). On the other hand, Hedge's g value (2.338) also supports this difference. As a result of this difference, F_w method performance is significantly higher than W method.

Table A.6. T-Test for UIO and F_w methods at Reset-State

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	79,189	0	-6,855	56	0	-692,993	101,089	-895,499	-490,487
Equal variances not assumed			-6,855	28,015	0	-692,993	101,089	-900,059	-485,926

A significant difference was found between the two methods in the T-Test values of the UIO and F_w methods (Table A.5). On the other hand, the Hedge's g values (1,800) supports this difference. As a result of this difference, F_w method performance is significantly higher than UIO method.

As a result, the highest performance in the Reset-State graph according to the T-Test and Hedge's g values belongs to the F method. The F method follows the F_w method in terms of performance rate. The performance of F and F_w methods yielded better results than W and UIO methods.

A.1.2. Reset and Input

Table A.7. General Statistics

	Mean	N	Std. Deviation
W	297,6	7	345,239
F	2,571	7	0,390
F_w	5,457	7	1,129
UIO	358,142	7	392,537

According to Table A.7, the average value of W method is 297,6000. The average value of the F method is 2,5714. The average of F_w method is 5,4571. The average UIO method is 358,1429. According to these results, the highest average value belongs to UIO method. The lowest average value belongs to the F method. Moreover, there is a huge difference between the highest average and the lowest average. Thus, the UIO method has the lowest performance. The F method has clearly the highest performance.

Table A.8. T-Test for W and F methods at Reset-Input

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,460	0,002	-2,260	12	0,043	-295,028	-295,028	-579,338	-10,719
Equal variances not assumed			-2,260	6	0,064	-295,028	-295,028	-614,321	24,264

When examined closely, it was seen that there was a difference between T-test values of F and W methods. On the other hand, Hedge's g value (2.208) also supports this difference. As a result of this difference, we can say that the performance of the F method is higher than the W method.

Table A.9. T-Test for UIO and F methods at Reset-Input

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	15,063	0,002	-2,396	12	0,033	-355,571	148,365	-678,831	-32,311
Equal variances not assumed			-2,396	6	0,053	-355,571	148,365	-718,607	7,464

When we look at the T-Test values of the UIO and F method, there is also a significant difference between the two methods. On the other hand, the Hedge's g value (1.281) supports this difference. As a result of this difference, we can say that the F method's performance is significantly higher than the UIO method.

Table A.10. T-Test for F_w and F methods at Reset-Input

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	4,794	0,049	-6,387	12	0	-2,885	0,451	-3,870	-1,901
Equal variances not assumed			-6,387	7,412	0	-2,885	0,451	-3,942	-1,829

When the T-Test values of the F_w and F methods were examined, a significant difference was found between the two methods. On the other hand, the hedge's g value (3.416) supports this difference. As a result of this difference, we can say that the F method's performance is significantly higher than the F_w method.

In the T-Test results of the W and F_w methods, it was seen that there was a difference between the reset-input values of both methods. On the other hand, the Hedge's g value (1.196) of the F_w and W methods also supports this difference. As a result of this difference, we can say that the performance of F_w method is higher than W method.

The T-Test values of the UIO and F_w methods differed between the reset-input values of both methods. On the other hand, the Hedge's g value (1.270) also supports

Table A.11. T-Test for W and F_w methods at Reset-Input

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,401	0,002	-2,238	12	0,044	-292,142	130,488	-576,453	-7,831
Equal variances not assumed			-2,238	6	0,066	-292,142	130,488	-611,436	27,150

Table A.12. T-Test for UIO and F_w methods at Reset-Input

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	15,009	0,002	-2,377	12	0,034	-352,685	148,365	-675,946	-29,424
Equal variances not assumed			-2,377	6	0,054	-352,685	148,365	-715,721	10,350

this difference. As a result of this difference, we can say that F_w method performance is higher than UIO method.

As a result, the highest performance in the Reset-Input graph according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. The performance of F and F_w methods in Reset-Input yielded better results than W and UIO methods.

A.1.3. Reset and Output

Table A.13. General Statistics

	Mean	N	Std. Deviation
W	18,064	31	5,637
F	2,541	31	0,194
F_w	2,858	31	0,893
UIO	19,141	31	9,178

According to Table A.13, the average value of W method is 18,0645. The average

value of F method is 2,5419. The average of F_w method is 2,8581. The average of the UIO method is 19,6516. According to these results, the highest average value belongs to UIO method. The lowest average value belongs to the F method. Moreover, there is a huge difference between the highest average and the lowest average. Thus, the UIO method has the lowest performance. The F method has clearly the highest performance.

Table A.14. T-Test for W and F methods at Reset-Output

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	17,460	0	-15,322	60	0	-15,522	1,013	-17,548	-13,496
Equal variances not assumed			-15,322	30,071	0	-15,522	1,013	-17,591	-13,453

In close examination, T-Test results of the W and F methods showed a significant difference between the reset-output values of both methods. On the other hand, Hedge's g value (3.892) also supports this difference. As a result of this difference, we can say that the performance of the F method is much higher than the W method.

Table A.15. T-Test for UIO and F methods at Reset-Output

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,586	0	-10,067	60	0	-16,6	1,648	-19,898	-13,301
Equal variances not assumed			-10,067	30,026	0	-16,6	1,648	-19,967	-13,232

When looking at the T-Test values of the UIO and F methods, a significant difference was found between the reset-output values of both methods. The Hedge's g value (2.557) of the F and UIO methods also supports this difference. As a result of this difference, we can say that the performance of the F method is significantly higher than the UIO method.

When the T-Test values of the F and F_w methods were considered, no difference was found between the reset-output values of the methods. Therefore, there is no need to refer to the Hedge's g value. As a result, we can say that there is no significant difference between F method performance and F_w method performance.

Table A.16. T-Test for F_w and F methods at Reset-Output

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	7,329	0,008	-1,925	30	0,058	-0,316	0,164	-0,644	0,012
Equal variances not assumed			-1,925	32,839	0,062	-0,316	0,164	-0,650	0,017

Table A.17. T-Test for W and F_w methods at Reset-Output

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	13,541	0	-14,834	60	0	-15,206	1,025	-17,256	-13,155
Equal variances not assumed			-14,834	31,505	0	-15,206	1,025	-17,295	-13,117

In the T-Test results of the W and F_w methods, a significant difference was determined between the reset-output values of both methods. On the other hand, the Hedge's g value (3,767) of the F_w and W methods also supports this difference. As a result of this difference, we can say that F_w method performance is quite high compared to W method.

Table A.18. T-Test for UIO and F_w methods at Reset-Output

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	12,508	0	-9,831	60	0	-16,283	1,656	-19,596	-12,970
Equal variances not assumed			-9,831	30,568	0	-16,283	1,656	-19,663	-12,903

The T-Test values of the UIO and F_w methods differed between the reset-output values of both methods. Similarly, Hedge's g value (2.497) supports this difference. As a result of this difference, we can say that F_w method performance is higher than UIO method.

As a result, in the Reset-Output graph, the highest performance with respect to T-Test and Hedge's g values belongs to the F_w method. In other words, the performance

of F and F_w methods in Reset-Output yielded better results than W and UIO methods.

A.2. Average Test Case Length

A.2.1. Average Test Case Length and State

Table A.19. General Statistics

	Mean	N	Std. Deviation
W	3,939	29	0,432
F	3,922	29	0,315
F_w	4,922	29	0,315
UIO	4,817	29	0,730

According to Table A.19, the average value of W method is 3,9394. The average value of the F method is 3,9229. The average of F_w method is 4,9229. The average UIO method is 4,8172. According to these results, the highest average value belongs to F_w method. The lowest average value belongs to the F method. However, there is no big difference between the highest average and the lowest average. Thus, the F_w method has the highest performance.

Table A.20. T-Test for W and F methods at Average Test Case Length-State

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	3,933	0,052	-0,166	56	0,868	-0,016	0,099	-0,215	0,182
Equal variances not assumed			-0,166	51,239	0,868	-0,016	0,099	-0,216	0,182

According to the T-Test results of W and F methods, no significant difference was found between the Average Test Case Length-State values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result of this situation, we can say that there is no significant difference between F and W methods performance.

Table A.21. T-Test for UIO and F methods at Average Test Case Length-State

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	17,526	0	-6,053	56	0	-0,894	0,147	-1,190	-0,598
Equal variances not assumed			-6,053	38,098	0	-0,894	0,147	-1,193	-0,595

The T-Test values of the F and UIO methods showed a slight difference between the Average Test Case Length-State values of both methods. The Hedge's g value (1.591) of the F and UIO methods also supports this difference. As a result of this difference, we can say that the performance of the UIO method is slightly higher than the F method.

Table A.22. T-Test for F_w and F methods at Average Test Case Length-State

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0	1	-12,068	56	0	-1	0,082	-1,165	-0,834
Equal variances not assumed			-12,068	56	0	-1	0,082	-1,165	-0,834

When the T-Test values of the F and F_w methods were considered, a difference was found between the Average Test Case Length-State values of the methods. The Hedge's g value (3.169) of the F and F_w methods also supports this difference. As a result of this situation, we can say that F_w method performance is higher than F method.

Table A.23. T-Test for W and F_w methods at Average Test Case Length-State

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	3,933	0,052	9,896	56	0	0,983	0,099	0,784	1,182
Equal variances not assumed			9,896	51,239	0	0,983	0,099	0,784	1,182

In the T-Test results of the W and F_w methods, a significant difference was found between the Average Test Case Length-State values of both methods. On the other hand,

the Hedge's g value (2.600) of the F_w and W methods supports this difference. As a result of this difference, we can say that F_w method performance is quite high compared to W method.

Table A.24. T-Test for UIO and F_w methods at Average Test Case Length-State

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	17,526	0	0,715	56	0,477	0,105	0,147	-0,190	0,401
Equal variances not assumed			0,715	38,098	0,478	0,105	0,147	-0,193	0,404

No significant difference was found between the UIO and F_w methods in the T-Test values. Therefore, there is no need to refer to the Hedge's g value. As a result of this situation, we can say that there is no significant difference between F_w and UIO methods performance.

As a result, in the Average Test Case Length -State graph, the highest performance according to the T-Test and Hedge's g values belongs to the F_w method.

A.2.2. Average Test Case Length and Input

Table A.25. General Statistics

	Mean	N	Std. Deviation
W	2,864	7	0,221
F	3,6	7	0,350
F_w	4,6	7	0,350
UIO	2,920	7	0,342

According to Table A.25, the average value of the W method is 2,8643. The average value of the F method is 3,6000. The average of the F_w method is 4,6000. The average of the UIO method is 2,9206. According to these results, the highest average value belongs to F_w method. The lowest average value belongs to the W method. Thus, the W method has the lowest performance. The F_w method has the highest performance.

Table A.26. T-Test for F and W methods at Average Test Case Length-Input

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0,914	0,357	4,694	12	0	0,735	0,156	0,394	1,077
Equal variances not assumed			4,694	10,122	0	0,735	0,156	0,387	1,084

When examined closely, the T-Test results of W and F methods significantly differed between Average Test Case Length -Input values of both methods. On the other hand, Hedge's g value (2.514) supports this difference. As a result of this difference, we can say that the F method's performance is quite high compared to the W method.

Table A.27. T-Test for UIO and F methods at Average Test Case Length-Input

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0,021	0,885	3,666	12	0	0,679	0,185	0,275	1,083
Equal variances not assumed			3,666	11,993	0	0,679	0,185	0,275	1,083

When the T-Test values of the F and UIO methods were examined, a difference was found between the Average Test Case Length -Input values of both methods. The Hedge's g value (1.965) of the F and UIO methods also supports this difference. As a result of this difference, we can say that the F method's performance is significantly higher than the UIO method.

Table A.28. T-Test for F_w and F methods at Average Test Case Length-Input

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0	1	-5,335	12	0	-1	0,187	-1,408	-0,591
Equal variances not assumed			-5,335	12	0	-1	0,187	-1,408	-0,591

When the T-Test values of the F and F_w methods were examined, a difference was

found between the Average Test Case Length -Input values of the methods. Hedge's g value of F and F_w methods (2.857) also supports this difference. As a result, we can say that the F method has a higher performance than the F_w method.

Table A.29. T-Test for W and F_w methods at Average Test Case Length-Input

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0,914	0,357	11,076	12	0	1,735	0,156	1,394	2,077
Equal variances not assumed			11,076	10,122	0	1,735	0,156	1,387	2,084

In the T-Test results of the F_w and W methods, a significant difference was found between the Average Test Case Length -Input values of both methods. On the other hand, the Hedge's g value (5.931) of the F_w and W methods also supports this difference. As a result of this difference, we can say that F_w method performance is quite high compared to W method.

Table A.30. T-Test for UIO and F_w methods at Average Test Case Length-Input

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0,021	0,885	9,064	12	0	1,679	0,185	1,275	2,083
Equal variances not assumed			9,064	11,993	0	1,679	0,185	1,275	2,083

The T-Test values of the F_w and UIO methods also differed between the Average Test Case Length -Input values of both methods. Similarly, Hedge's g value (4.855) also supports this difference. As a result of this difference, we can say that F_w method performance is higher than UIO method.

As a result, in the Average Test Case Length -Input chart, the highest performance F method according to T-Test and Hedge's g values belongs to F_w method. In other words, the performance of F and F_w methods in Average Test Case Length -Input yielded better results than W and UIO methods.

A.2.3. Average Test Case Length and Output

Table A.31. General Statistics

	Mean	N	Std. Deviation
W	2,938	31	0,058
F	4,027	31	0,317
F_w	5,027	31	0,317
UIO	2,965	31	0,139

According to Table A.31, the average value of W method is 2,9387. The average value of the F method is 4.0280. The average F_w method is 5.0280. The average of the UIO method is 2,9652. According to these results, the highest average value belongs to F_w method. The lowest average value belongs to the W method. On the other hand, there is no big difference between the highest average and the lowest average. Thus, the W method has the lowest performance. The F_w method has the highest performance.

Table A.32. T-Test for W and F methods at Average Test Case Length-Output

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	36,195	0	18,794	60	0	1,089	0,057	0,973	1,205
Equal variances not assumed			18,794	32,054	0	1,089	0,057	0,971	1,207

In close examination, the T-Test results of W and F methods differed between Average Test Case Length - Output values of both methods. On the other hand, Hedge's g value (4.783) also supports this difference. As a result of this difference, we can say that the performance of the F method is much higher than the W method.

Considering the T-Test values of the F and UIO methods, a significant difference was found between the two methods. Hedge's g value (4.343) also supports this difference. As a result of this difference, we can say that the F method's performance is significantly higher than the UIO method.

When the T-test values of the F and F_w methods were considered, a difference was found between the Average Test Case Length - Output values of both methods. The

Table A.33. T-Test for UIO and F methods at Average Test Case Length-Output

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	19,697	0	17,064	60	0	1,062	0,062	0,938	1,187
Equal variances not assumed			17,064	41,243	0	1,062	0,062	0,937	1,188

Table A.34. T-Test for F_w and F methods at Average Test Case Length-Output

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	0	1	-12,408	60	0	-1	0,080	-1,161	-0,838
Equal variances not assumed			-12,408	60	0	-1	0,080	-1,161	-0,838

Hedge's g value (3.154) also supports this difference. As a result of this difference, we can say that the F_w method's performance is slightly higher than the F method.

Table A.35. T-Test for W and F_w methods at Average Test Case Length-Output

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	36,195	0	36,048	60	0	2,089	0,057	1,973	2,205
Equal variances not assumed			36,048	32,054	0	2,089	0,057	1,971	2,207

In the T-Test results of the F_w and W methods, there was a significant difference between the Average Test Case Length - Output values of both methods. On the other hand, the Hedge's g value (9.171) of the F_w and W methods supports this difference. As a result of this difference, we can say that the performance of the F_w method is much higher than the W method.

A significant difference was found between the Average Test Case Length - Output values of both methods in T - Test values of F_w and UIO method. On the other hand, the Hedge's g value (8.428) of the UIO method with the F_w method also supports this

Table A.36. T-Test for UIO and F_w methods at Average Test Case Length-Output

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	19,697	0	33,119	60	0	2,062	0,062	1,938	2,187
Equal variances not assumed			33,119	41,243	0	2,062	0,062	1,937	2,188

difference. As a result of this difference, we can say that F_w method performance is quite high compared to UIO method.

As a result, in the Average Test Case Length - Output graph, the highest performance according to T - Test and Hedge's g values belongs to F_w method. The F method follows the F_w method in terms of performance. Therefore, the performance of F_w and F methods in Average Test Case Length - Output yielded better results than W and UIO methods.

A.3. Test Suite Length

A.3.1. Test Suite Length and State

Table A.37. General Statistics

	Mean	N	Std. Deviation
W	1136,744	29	693,191
F	24,931	29	6,246
F_w	116,179	29	42,097
UIO	3795,441	29	3135,607

According to Table A.37, the average value of W method is 1136,7448. The average value of the F method is 24,9310. The average of F_w method is 116,1793. The average of the UIO method is 1268,3241. According to these results, the highest average value belongs to UIO method. The lowest average value belongs to the F method. On

the other hand, there is a huge difference between the highest average and the lowest average. Thus, the UIO method has the lowest performance. The F method has the highest performance.

Table A.38. T-Test for W and F methods at Test Suite Length-State

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	93,934	0	-8,636	56	0	-1111,813	128,727	-1369,686	-853,941
Equal variances not assumed			-8,636	28,004	0	-1111,813	128,727	-1375,498	-848,129

When examined closely, in the T-Test results of the W and F methods, a significant difference is observed between the Test Suite length-States values of both methods. The Hedge's g value (2.2681), on the other hand, supports this difference. As a result of this difference, we can say that the performance of the F method is much higher than the W method.

Table A.39. T-Test for UIO and F methods at Test Suite Length-State

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	82,174	0	-6,475	56	0	-3770,510	582,268	-4936,934	-2604,085
Equal variances not assumed			-6,475	28	0	-3770,510	582,268	-4963,233	-2577,787

Considering the T-Test values of the F and UIO methods, a significant difference was found between the two methods. On the other hand, the Hedge's g value (1,700) supports this difference. As a result of this difference, we can say that the F method's performance is significantly higher than the UIO method.

When the T-Test values of the F and F_w methods were examined, a difference was found between the Test Suite length - States values of both methods. The Hedge's g value (3.032) also supports this difference. As a result of this difference, we can say that the F method's performance is higher than the F_w method.

In the T-Test results of the F_w and W methods, it was seen that there was a difference between the Test Suite length-States values of both methods. On the other hand, the

Table A.40. T-Test for F_w and F methods at Test Suite Length-State

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	50,646	0	-11,546	56	0	-91,248	7,902	-107,079	-75,416
Equal variances not assumed			-11,546	29,232	0	-91,248	7,902	-107,405	-75,090

Table A.41. T-Test for W and F_w methods at Test Suite Length-State

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	84,323	0	-7,913	56	0	-1020,565	128,959	-1278,902	-762,228
Equal variances not assumed			-7,913	28,206	0	-1020,565	128,959	-1284,640	-756,490

Hedge's g value (2.078) of the F_w and W methods also supports this difference. As a result of this difference, we can say that the F_w method performance is quite high compared to W method.

Table A.42. T-Test for UIO and F_w methods at Test Suite Length-State

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	80,317	0	-6,318	56	0	-3679,262	582,320	-4845,789	-2512,734
Equal variances not assumed			-6,318	28,010	0	-3679,262	582,320	-4872,071	-2486,452

The T-Test values of the F_w and UIO methods also differed between the Test Suite length-States values of both methods. On the other hand, the Hedge's g value (1.659) of the UIO method with the F_w method also supports this difference. As a result of this difference, we can say that the F_w method performance is higher than UIO method.

As a result, the highest performance according to T-Test and Hedge's g values belongs to F method. The F_w method follows the F method in terms of performance rate. Therefore, the performance of F and F_w methods in Test Suite length - States yielded

better results than W and UIO methods.

A.3.2. Test Suite Length and Input

Table A.43. General Statistics

	Mean	N	Std. Deviation
W	820,914	7	947,131
F	9,285	7	1,692
F_w	25,114	7	5,494
UIO	989,314	7	1075,243

According to Table A.25, the average value of W method is 820,9143. The average value of F method is 9,2857. The average of F_w method is 25,1143. The average of UIO method is 989,3143. According to these results, the highest average value belongs to UIO method. The lowest average value belongs to the F method. Also, there is a huge difference between the highest average and the lowest average. Thus, the UIO method has the lowest performance. The F method has the highest performance.

Table A.44. T-Test for W and F methods at Test Suite Length-Input

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,480	0,002	-2,267	12	0,042	-811,628	357,982	-1591,605	-31,651
Equal variances not assumed			-2,267	6	0,063	-811,628	357,982	-1687,578	64,321

In close examination of the T-Test results of W and F method, significant difference is observed between Test Suite Length - Input values of both methods. The Hedge's g value (1.211), on the other hand, supports this difference. As a result of this difference, we can say that the performance of the F method is higher than the W method.

Considering the T-Test values of the F and UIO method, a significant difference was found between the two methods. On the other hand, Hedge's g value (1.288) also supports this difference. As a result of this difference, we can say that the performance of the F method is significantly higher than the UIO method.

Table A.45. T-Test for UIO and F methods at Test Suite Length-Input

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	15,114	0,002	-2,411	12	0,032	-980,028	406,404	-1865,507	-94,549
Equal variances not assumed			-2,411	6	0,052	-980,028	406,404	-1974,462	14,405

Table A.46. T-Test for F_w and F methods at Test Suite Length-Input

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	7,267	0,019	-7,284	12	0	-15,828	2,172	-20,562	-11,094
Equal variances not assumed			-7,284	7,128	0	-15,828	2,172	-20,947	-10,709

When the T-Test values of the F and F_w methods were considered, a difference was found between the Test Suite Length - Input values of both methods. In addition, Hedge's g value (3,894) also supports this difference. As a result of this difference, we can say that the performance of F method is higher than F_w method.

Table A.47. T-Test for W and F_w methods at Test Suite Length-Input

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,366	0,002	-2,222	12	0,046	-795,8	357,987	-1575,788	-15,811
Equal variances not assumed			-2,222	6	0,067	-795,8	357,987	-1671,750	80,150

In the T-Test results of the F_w and W method, there was a difference between the Test Suite Length - Input values of both methods. On the other hand, the Hedge's g value (1.188) of the F_w and W methods supports this difference. As a result of this difference, we can say that the performance of F_w method is higher than W method.

The T-Test values of the F_w and UIO methods were also different between the Test Suite Length-Input values of both methods. On the other hand, the Hedge's g value

Table A.48. T-Test for UIO and F_w methods at Test Suite Length-Input

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	15,011	0,002	-2,372	12	0,035	-964,2	406,409	-1849,689	-78,710
Equal variances not assumed			-2,372	6	0,055	-964,2	406,409	-1958,634	30,234

(1.268) of the UIO method with the F_w method supports this difference. As a result of this difference, we can say that the F_w method's performance is quite high compared to the UIO method.

As a result, in the Test Suite Length - Input graph, the highest performance according to T - Test and Hedge's g values belongs to the F method. The F method follows the F_w method in terms of performance rate. Therefore, the performance of F and F_w methods in Test Suite Length - Input yielded better results than W and UIO methods.

A.3.3. Test Suite Length and Output

Table A.49. General Statistics

	Mean	N	Std. Deviation
W	53,212	31	17,098
F	10	31	0,572
F_w	14,148	31	4,326
UIO	57,903	31	33,589

According to Table A.31, the average value of W method is 53,2129. The average value of the F method is 10,0000. The average of the F_w method is 14,1484. The average of UIO method is 57,9032. According to these results, the highest average value belongs to UIO method. The lowest average value belongs to the F method. There is also a notable difference between the highest average and the lowest average. Thus, the UIO method has the lowest performance. The F method has the highest performance.

A close examination showed that there was a significant difference between the Test Suite Length - Output values of both methods in the T - Test results of W and F

Table A.50. T-Test for W and F methods at Test Suite Length-Output

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	17,570	0	-14,063	60	0	-43,212	3,072	-49,359	-37,066
Equal variances not assumed			-14,063	30,067	0	-43,212	3,072	-49,487	-36,938

method. On the other hand, Hedge's g value (3.572) also supports this difference. As a result of this difference, we can say that the performance of the F method is much higher than the W method.

Table A.51. T-Test for UIO and F methods at Test Suite Length-Output

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	13,031	0	-7,939	60	0	-47,903	6,033	-59,972	-35,833
Equal variances not assumed			-7,939	30,017	0	-47,903	6,033	-60,225	-35,580

Considering the T-Test values of the F and UIO method, a significant difference was found between the two methods. On the other hand, Hedge's g value (2.016) also supports this difference. As a result of this difference, we can say that the performance of the F method is significantly higher than the UIO method.

Table A.52. T-Test for F_w and F methods at Test Suite Length-Output

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	12,438	0	-5,292	60	0	-4,148	0,783	-5,716	-2,580
Equal variances not assumed			-5,292	31,051	0	-4,148	0,783	-5,746	-2,549

When the T-test values of the F and F_w methods were considered, a difference was found between the Test Suite Length - Output values of both methods. In addition, the

Hedge's g value (1.344) also supports this difference. As a result of this difference, we can say that the performance of F method is higher than F_w method.

Table A.53. T-Test for W and F_w methods at Test Suite Length-Output

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	10,140	0,002	-12,332	60	0	-39,064	3,167	-45,400	-32,728
Equal variances not assumed			-12,332	33,825	0	-39,064	3,167	-45,503	-32,625

In the T-Test results of the F_w and W method, there was a difference between the Test Suite Length - Output values of both methods. On the other hand, the Hedge's g value (3.132) of the F_w and W methods supports this difference. As a result of this difference, we can say that the performance of the F_w method is much higher than the W method.

Table A.54. T-Test for UIO and F_w methods at Test Suite Length-Output

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	9,932	0,002	-7,193	60	0	-43,754	6,082	-55,922	-31,587
Equal variances not assumed			-7,193	30,995	0	-43,754	6,082	-56,160	-31,349

The T-Test values of the F_w and UIO methods also differed between the Test Suite Length - Output values of both methods. In addition, the Hedge's g value (1.827) of the UIO method with the F_w method supports this difference. As a result of this difference, we can say that the F_w method's performance is higher than the UIO method.

As a result, in the Test Suite Length - Output graph, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. Therefore, the performance of F and F_w methods in Test Suite Length - Output yielded better results than W and UIO methods.

A.4. Fault Detection Ratio

A.4.1. Fault Detection Ratio and State

Table A.55. General Statistics

	Mean	N	Std. Deviation
W	1	29	0
F	0,816	29	0,100
F_w	0,979	29	0,017
UIO	0,999	29	0,001

According to Table A.55, the average value of W method is 1,0000. The average value of the F method is 0,8161. The average of F_w method is 0.9795. The average of the UIO method is 0.9488. According to these results, the highest average value belongs to W method. The lowest average value belongs to the F method. On the other hand, the difference between the highest average and the lowest average is quite low. However, according to these results, the highest performance belongs to W method.

Table A.56. T-Test for W and F methods at Fault Detection Ratio-State

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	74,156	0	-9,841	56	0	-0,183	0,018	-0,221	-0,146
Equal variances not assumed			-9,841	28	0	-0,183	0,018	-0,222	-0,145

When examined closely, the difference between the T-test values of F and UIO methods is remarkable. On the other hand, since the std. deviation of the W method is zero (0), the Hedge's g value cannot be calculated. However, according to the T-Test values, we can say that the performance of the W method, with little difference, is higher than the F method.

When the T-test values of F and UIO methods were examined, there was a difference between the two methods. On the other hand, Hedge's g value (2.587) of F and

Table A.57. T-Test for UIO and F methods at Fault Detection Ratio-State

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	73,128	0	-9,825	56	0	-0,183	0,018	-0,221	-0,146
Equal variances not assumed			-9,825	28,013	0	-0,183	0,018	-0,221	-0,145

UIO methods supports this difference. As a result of this difference, we can say that the performance of the F method is lower than the UIO method.

Table A.58. T-Test for F_w and F methods at Fault Detection Ratio-State

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	49,808	0	-8,616	56	0	-0,163	0,018	-0,201	-0,125
Equal variances not assumed			-8,616	29,704	0	-0,163	0,018	-0,202	-0,124

When the T-Test values of the F and F_w methods were examined, a difference was found between the Fault Detection Ratio - State values of both methods. In addition, the Hedge's g value (2,272) also supports this difference. As a result of this difference, we can say that the F_w method performance is higher than F method.

Table A.59. T-Test for W and F_w methods at Fault Detection Ratio-State

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	50,621	0	-6,272	56	0	-0,020	0,003	-0,026	-0,013
Equal variances not assumed			-6,272	28	0	-0,020	0,003	-0,027	-0,013

In the T-Test results of the F_w and W methods, it was seen that there was a difference between the Fault Detection Ratio - State values of both methods. On the other hand, since the std. deviation of the W method is zero (0), the Hedge's g value cannot be calculated. However, according to T-Test values, we can say that the W method performance

is higher than the F_w method with little difference.

Table A.60. T-Test for UIO and F_w methods at Fault Detection Ratio-State

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	45,781	0	-6,160	56	0	-0,020	0,003	-0,026	-0,013
Equal variances not assumed			-6,160	28,434	0	-0,020	0,003	-0,026	-0,013

The T-Test values of the F_w and UIO methods also differed between the Fault Detection Ratio - State values of both methods. In addition, the Hedge's g value (1.660) of the UIO method with F_w method supports this difference. As a result of this difference, we can say that the performance of the UIO method is slightly higher than the F method.

As a result, in the Fault Detection Ratio - State graph, the highest performance according to the T - Test and Hedge's g values belongs to the W method. The W method is followed by the UIO method in terms of performance. However, it is necessary to underline that the values of all four methods are very close to each other in terms of Fault Detection Ratio - State values.

A.4.2. Fault Detection Ratio and Input

Table A.61. General Statistics

	Mean	N	Std. Deviation
W	1	7	0
F	0,992	7	0,018
F_w	1	7	0
UIO	1	7	0

According to Table A.61, the average value of the W method is 1,0000. The average value of the F method is 0.9929. The average F_w method is 1,0000. The average of the UIO method is 1,0000. According to these results, the average of each of the four methods is very close to each other. Even, except for the F method, the Fault Detection Ratio - Input averages of the other three methods are exactly the same.

Table A.62. T-Test for W and F methods at Fault Detection Ratio-Input

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	5,759	0,033	-1	12	0,337	-0,007	0,007	-0,022	0,008
Equal variances not assumed			-1	6	0,355	-0,007	0,007	-0,024	0,010

A close examination reveals a slight difference between the T-Test results of W and F methods. On the other hand, since the std. deviation of the W method is zero (0), the Hedge's g value cannot be calculated. However, according to the T-Test values, the performance of the W method, with little difference, is higher than the F method.

When the T-test values of the F and UIO methods were considered, a slight difference was found between the two methods. On the other hand, since the std. deviation of the UIO method is zero (0), the Hedge's g value cannot be calculated. However, according to the T-Test values, the performance of the UIO method, with little difference, is higher than the F method.

When the T-Test values of the F and F_w methods were considered, a slight difference was found between the Fault Detection Ratio - Input values of both methods. On the other hand, since the std. deviation of the F_w method is zero (0), the Hedge's g value cannot be calculated. However, according to the T-Test values, the F_w method's performance is slightly higher than the F method, with little difference.

Since the values of both the F_w and W methods are exactly the same and std. deviation is zero (0), the T-Test values are zero (0). For the same reason Hedge's g values cannot be calculated. This means that the F_w method and the W method's Fault Detection Ratio-Input values are exactly the same.

Since the values of both the F_w and UIO method are exactly the same and the std. deviation is zero (0), the T-Test values are zero (0). For the same reason Hedge's g values cannot be calculated. This means that the F_w method and the W method's Fault Detection Ratio-Input values are exactly the same.

As a result, according to the Fault Detection Ratio - Input graph, we can say that the highest performance belongs to F_w, W and UIO methods. In terms of performance, the F method is slightly behind these three methods. However, it is necessary to underline that the values of all four methods are very close to each other in terms of Fault Detection

Ratio - Input values.

A.4.3. Fault Detection Ratio and Output

Table A.63. General Statistics

	Mean	N	Std. Deviation
W	0,988	31	0,030
F	0,975	31	0,031
F_w	0,972	31	0,043
UIO	0,988	31	0,030

According to Table A.63, the average value of W method is 0.9883. The average value of the F method is 0.9750. The average of F_w method is 0.9726. The average of UIO method is 0.9883. According to these results, the highest average value belongs to W and UIO methods. The lowest average value belongs to the F_w method. However, there is no significant difference between the highest average and the lowest average.

Table A.64. T-Test for F_w, F and UIO, F methods at Fault Detection Ratio-Output

W-F UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	2,5109	0,118	-1,687	60	0,096	-0,013	0,007	-0,029	0,002
Equal variances not assumed			-1,687	59,983	0,096	-0,013	0,007	-0,029	0,002

When examined closely, no significant difference was found in the T-Test results of W and F method in terms of Fault Detection Ratio - Output values. Therefore, there is no need to refer to the Hedge's g value. As a result, we can say that the performances of the F and W methods are the same.

In the T-Test results of the F and UIO method, there was no significant difference between the Fault Detection Ratio-Output values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result, we can say that the performance of the F method and the performance of the UIO method are the same.

Table A.65. T-Test for F_w and F methods at Fault Detection Ratio-Output

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	2,527	0,117	0,250	60	0,802	0,002	0,009	-0,016	0,021
Equal variances not assumed			0,250	54,353	0,802	0,002	0,009	-0,016	0,021

The same is also available in the T-Test results of the F_w and F methods. There was no significant difference between Fault Detection Ratio - Output values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result, we can say that the performances of the F_w and F methods are the same.

Table A.66. T-Test for W, F_w and UIO, F_w methods at Fault Detection Ratio-Output

W-F_w **UIO-F_w**	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	6,800	0,011	-1,637	60	0,106	-0,015	0,009	-0,034	0,003
Equal variances not assumed			-1,637	53,877	0,106	-0,015	0,009	-0,034	0,003

In the T-Test results of the F_w and W methods, there was no significant difference between the Fault Detection Ratio - Output values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result, the performances of the F_w and W methods are the same.

The same is also present in the T-Test results of the F_w and UIO methods. We can say that there was no significant difference between Fault Detection Ratio - Output values of both methods. Therefore, there is no need to refer to the Hedge's g value.

As a result, the performances of the F_w and UIO methods are the same. As a result, according to the Fault Detection Ratio - Output graph, we can say that the performance of the four methods is exactly the same.

A.5. Killed Mutant / Test Suite Size

A.5.1. Killed Mutant / Test Suite Size and State

Table A.67. General Statistics

	Mean	N	Std. Deviation
W	0,076	29	0,029
F	2,152	29	0,412
F_w	0,588	29	0,097
UIO	0,034	29	0,028

According to Table A.67, the average value of W method is 0,0770. The average value of the F method is 2,1523. The F_w method has an average of 0.5883. The average of the UIO method is 0.0346. According to these results, the highest average value belongs to the F method. The lowest average value belongs to the UIO method. Thus, the UIO method has the lowest performance. The F method has the highest performance.

Table A.68. T-Test for W and F methods at Killed Mutant / Test Suite Size-State

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	61,445	0	27,012	56	0	2,075	0,076	1,921	2,229
Equal variances not assumed			27,012	28,283	0	2,075	0,076	1,917	2,232

When the T-test results of W and F methods are examined closely, significant difference is observed between Killed Mutant / Test Suite Size - State values of both methods. On the other hand, Hedge's g value (7.104) supports this difference. As a result of this difference, we can say that the performance of the F method is much higher than the W method.

Considering the T-Test values of the F and UIO methods, a significant difference was found between the Killed Mutant / Test Suite Size - State values of both methods.

Table A.69. T-Test for UIO and F methods at Killed Mutant / Test Suite Size-State

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	61,891	0	27,566	56	0	2,117	0,076	1,963	2,271
Equal variances not assumed			27,566	28,276	0	2,117	0,076	1,960	2,274

Hedge's g value (7.253) also supports this difference. As a result of this difference, we can say that the performance of the F method is significantly higher than the UIO method.

Table A.70. T-Test for F_w and F methods at Killed Mutant / Test Suite Size-State

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	38,464	0	19,864	56	0	1,563	0,078	1,406	1,721
Equal variances not assumed			19,864	31,094	0	1,563	0,078	1,403	1,724

When the T-Test values of the F and F_w methods were examined, a difference was found between the Killed Mutant / Test Suite Size - State values of both methods. The Hedge's g value (5.225) also supports this difference. As a result of this difference, we can say that the performance of F method is higher than F_w method.

Table A.71. T-Test for W and F_w methods at Killed Mutant / Test Suite Size-State

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	42,954	0	27,129	56	0	0,511	0,018	0,473	0,549
Equal variances not assumed			27,129	33,070	0	0,511	0,018	0,473	0,549

In the T-Test results of the F_w and W methods, a great difference is observed between the Killed Mutant / Test Suite Size - State values of both methods. On the other hand, the Hedge's g value (7.137) of the F_w and W methods supports this big difference. As a result of this difference, we can say that the performance of the F_w method is much

higher than the W method.

Table A.72. T-Test for UIO and F_w methods at Killed Mutant / Test Suite Size-State

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	44,180	0	29,410	56	0	0,553	0,018	0,516	0,591
Equal variances not assumed			29,410	32,947	0	0,553	0,018	0,515	0,592

The T-Test values of the F_w and UIO methods significantly differed between the Killed Mutant / Test Suite Size - State values of both methods. In addition, the Hedge's g value (7.760) of the UIO method with the F_w method also supports this huge difference. As a result of this difference, we can say that the F_w method's performance is quite high compared to the UIO method.

As a result, in the Killed Mutant / Test Suite Size - State graph, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. Therefore, the performance of F and F_w methods in Killed Mutant / Test Suite Size - State yielded much better results than W and UIO methods.

A.5.2. Killed Mutant / Test Suite Size - Input

Table A.73. General Statistics

	Mean	N	Std. Deviation
W	0,167	7	0,0190
F	17,812	7	23,882
F_w	6,900	7	10,229
UIO	0,132	7	0,0248

According to Table A.73, the average value of the W method is 0,1672. The average value of the F method is 17,8128. The average of F_w method is 6,9003. The average of the UIO method is 0,1330. According to these results, the highest average value belongs to the F method. The lowest average value belongs to the UIO method.

Therefore, the UIO method has the lowest performance. The F method has the highest performance.

Table A.74. T-Test for W and F methods at Killed Mutant / Test Suite Size-Input

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,265	0,002	1,954	12	0,074	17,645	9,026	-2,022	37,313
Equal variances not assumed			1,954	6	0,098	17,645	9,026	-4,442	39,733

When the T-test results of W and F method were examined closely, there was no significant difference between Killed Mutant / Test Suite Size-Input values. Therefore, there is no need to refer to the Hedge's g value. As a result, the performance of the method F and W can be considered the same.

Table A.75. T-Test for UIO and F methods at Killed Mutant / Test Suite Size-Input

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	14,260	0,002	1,958	12	0,073	17,679	9,026	-1,988	37,347
Equal variances not assumed			1,958	6	0,097	17,679	9,026	-4,408	39,767

In the T-Test results of the F and UIO method, there was no significant difference between the Killed Mutant / Test Suite Size - Input values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result of this, the performance of the F and UIO method can be considered the same.

The same is also available in the T-Test results of the F_w and F method. There was no significant difference between Killed Mutant / Test Suite Size - Input values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result, the F_w and F method's performance is the same.

In the T-Test results of the F_w and W method, there was no significant difference between the Killed Mutant / Test Suite Size - Input values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result, the performance of the F_w and W method can be considered the same.

Table A.76. T-Test for F_w and F methods at Killed Mutant / Test Suite Size-Input

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	4,162	0,063	1,111	12	0,288	10,912	9,820	-10,483	32,308
Equal variances not assumed			1,111	8,129	0,298	10,912	9,820	-11,669	33,494

Table A.77. T-Test for F_w and W methods at Killed Mutant / Test Suite Size-Input

W-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	9,318	0,010	1,741	12	0,107	6,733	3,866	-1,690	15,156
Equal variances not assumed			1,741	6	0,132	6,733	3,866	-2,727	16,193

The same is also present in the T-Test results of the F_w and UIO method. There was no significant difference between Killed Mutant / Test Suite Size - Input values of both methods. Therefore, there is no need to refer to the Hedge's g value. As a result, the performance of the F_w and UIO method can be considered the same.

As a result, according to the Killed Mutant / Test Suite Size - Input graph, the performance of the four methods can be considered the same.

A.5.3. Killed Mutant / Test Suite Size - Output

According to Table A.79, the average value of W method is 0.3153. The average value of the F method is 1,5431. The average of F_w method is 1,1499. The average of UIO method is 0,3105. According to these results, the highest average value belongs to the F method. The lowest average value belongs to the UIO method. Thus, the UIO method has the lowest performance. The F method has the highest performance.

In the T-test results of W and F methods, significant difference is observed between Killed Mutant / Test Suite Size - Output values. On the other hand, the Hedge's g value (14.431) also supports this huge difference. As a result of this difference, we can

Table A.78. T-Test for F_w and UIO methods at Killed Mutant / Test Suite Size-Input

UIO-F_w	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	9,308	0,010	1,750	12	0,105	6,767	3,866	-1,656	15,191
Equal variances not assumed			1,750	6	0,130	6,767	3,866	-2,692	16,227

Table A.79. General Statistics

	Mean	N	Std. Deviation
W	0,315	31	0,051
F	1,543	31	0,109
F_w	1,149	31	0,202
UIO	0,310	31	0,064

say that the performance of the F method is much higher than the W method.

When the T-test results of F and UIO methods are examined, a significant difference is found between Killed Mutant / Test Suite Size - Output values. On the other hand, the Hedge's g value (13,795) supports this big difference. As a result of this difference, we can say that the performance of the F method is significantly higher than the UIO method.

When the T-test values of the F and F_w methods were considered, a difference was found between the Killed Mutant / Test Suite Size - Output values of both methods. In addition, the Hedge's g value (2.427) also supports this difference. As a result of this difference, we can say that the performance of F method is higher than F_w method.

In the T-Test results of F_w and W methods, there is a great difference between Killed Mutant / Test Suite Size - Output values of both methods. On the other hand, the Hedge's g value (5.661) of the F_w and W methods supports this difference. As a result of this difference, we can say that the performance of the F_w method is much higher than the W method.

A significant difference was found between the Killed Mutant / Test Suite Size-Output values of both methods in T-Test values of F_w and UIO methods. In addition, the Hedge's g value (5.599) of the UIO method with the F_w method supports this difference. As a result of this difference, we can say that the F_w method's performance is quite high

Table A.80. T-Test for W and F methods at Killed Mutant / Test Suite Size-Output

W-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	17,145	0	56,639	60	0	1,227	0,021	1,184	1,271
Equal variances not assumed			56,639	42,776	0	1,227	0,021	1,184	1,271

Table A.81. T-Test for UIO and F methods at Killed Mutant / Test Suite Size-Output

UIO-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	10,669	0,001	54,083	60	0	1,232	0,022	1,187	1,278
Equal variances not assumed			54,083	48,812	0	1,232	0,022	1,187	1,278

compared to the UIO method.

As a result, in the Killed Mutant / Test Suite Size - Output graph, the highest performance according to the T-Test and Hedge's g values belongs to the F method. The F_w method follows the F method in terms of performance. Therefore, the performance of F and F_w methods in Killed Mutant / Test Suite Size - Output yielded much better results than W and UIO methods.

Table A.82. T-Test for F_w and F methods at Killed Mutant / Test Suite Size - Output

F_w-F	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	3,308	0,073	9,520	60	0	0,393	0,041	0,310	0,475
Equal variances not assumed			9,520	46,073	0	0,393	0,041	0,310	0,475

Table A.83. T-Test for F_w and W methods at Killed Mutant / Test Suite Size - Output

F_w-W	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	15,120	0	22,242	60	0	0,834	0,037	0,759	0,909
Equal variances not assumed			22,242	33,880	0	0,834	0,037	0,758	0,910

Table A.84. T-Test for F_w and UIO methods at Killed Mutant / Test Suite Size - Output

F_w-UIO	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	Lower	Upper
Equal variances assumed	12,504	0	21,987	60	0	0,839	0,0381	0,763	0,915
Equal variances not assumed			21,987	36,080	0	0,839	0,0381	0,761	0,916

VITA

Savaş Takan received the BSc degree (2009) in Computer Engineering from İzmir Institute of Technology, Turkey. From 2009 to 2019 he worked as a research assistant at İzmir Institute of Technology. He received the MS degree (2012) in Computer Engineering from İzmir Institute of Technology. His research interests include Formal Methods for Software Testing, Bioinformatics, Robotics, and Dependable Systems.

Permanent Address: İZMİR