A comparative study of algorithms for generating switch cover test sets

Matheus Monteiro Mariano[1,2], **Érica Ferreira Souza**[3], **André Takeshi Endo**[3], **Nandamudi L. Vijaykumar**[1]

[1]Laboratório de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais, INPE
[2] Faculdade de Tecnologia, FATEC
São José dos Campos – SP – Brasil

[3]Departamento de Computação
Universidade Tecnológica Federal do Paraná, UTFPR
Cornélio Procópio – PR – Brasil

matheus.mariano2@fatec.sp.gov.br, ericasouza@utfpr.edu.br
andreendo@utfpr.edu.br, vijay.nl@inpe.br

Abstract. *Test case generation based on Finite State Machines (FSMs) has been extensively investigated due to its accuracy and simplicity. Several test criteria have been proposed in the literature to generate test cases based on FSMs. One of the oldest criteria is the Switch Cover. As a main feature, the Switch Cover criterion defines that all transition pairs of an FSM must be covered. The classical Switch Cover algorithm converts the FSM into a graph (known as Dual Graph); this graph is balanced, and, finally, traversed based on an Eulerian Cycle algorithm. In this context, considering the stage where an FSM is converted into a graph, this study investigates other search algorithms on graphs, namely Depth-First Search (DFS) and Breadth-First Search (BFS), for generating test sets from a Dual Graph. We presented an experimental study that compares the DFS, BFS algorithms with the Eulerian Cycle. The study was conducted with a set of random and real-world machines, taking into account the number of test cases, the test suite size, the average length of sequences and generation time.*

1. Introduction

Advances in technology and the emergence of increasingly complex and critical systems require using improved test strategies in order to achieve high quality software products. Formal methods have been used for the software development and verification, however, software testing activities are still needed to complement these techniques by executing the system and comparing the obtained behavior with the expected one. During the testing activity, a model to guide the process of testing a system can be created. According to El-Far and Whittaker [El-Far and Whittaker 2001], software testing requires the use of a model to guide such efforts as support for test selection and verification. In this context, Model-Based Testing (MBT) has drawn lot of attention in both industrial and academic areas since it has proved effective by using models to represent system behavior in order to guide the generation/selection of test cases [Santiago 2011]. The sound adoption of MBT can offer some benefits, such as high fault detection ratio, reduced cost and time, traceability, and ease of handling requirements evolution [Utting and Legeard 2006].

One of the main features of MBT is the automated generation of test cases based on a formal representation, for example, Finite State Machines (FSMs) [Gill 1962, Sidhu and Leung 1989]. FSM is a formal modeling technique commonly used for testing due to its rigor and simplicity. FSM has been adopted, mainly, for modeling reactive systems and protocol implementations for a long time [Pinheiro et al. 2014, Pontes et al. 2014]. The published literature shows efforts of the scientific community in analyzing the efficiency of test criteria for several techniques like FSMs. Great effort has been spent on the development of criteria and methods that select/generate effective test sets, i.e., criteria that generate a test set capable of revealing all faults from a given fault domain. There are several test criteria and methods in the literature to generate test cases based on FSM, such as Switch Cover [Pimont and Rault 1976], Transition Tour (TT) [Naito and Tsunoyama 1981], Distinguishing Sequence (DS) [Guney 1970], Unique Input/Output (UIO) [Sabnani 1988], State Counting [Petrenko and Yevtushenko 2005] and, recently, H [Dorofeeva et al. 2005], SPY [Simão et al. 2009], P [Simão et al. 2010], Diagnosable Input/Output (DIO) [Zhang et al. 2011] and H-Switch Cover [Souza et al. 2015].

In particular, Switch Cover is an old criterion and has been investigated for a long time by some research groups [Arantes et al. 2008, Martins et al. 1999, Pimont and Rault 1976, Souza et al. 2008]. The main objective of the Switch Cover criterion is to cover all pairs of transitions of an FSM [Pimont and Rault 1976]. The Switch Cover algorithm performs a more refined test coverage where a switch is basically a branch-to-branch pair, and test cases consist of every branch-to-branch pair from a graph [Arantes et al. 2014]. One of the main characteristics of Switch Cover criterion is to generate a graph from an FSM, known as Dual Graph. The Dual Graph is then balanced and traversed based on an algorithm that generates test cases. A new version of Switch Cover criterion was created recently, called H-Switch Cover [Souza et al. 2015]. The fundamental characteristic of H-Switch Cover criterion is the ability to deal with complex FSMs, the performance improvement, and the use of the *Hierholzer* [Neumann 2004] algorithm as heuristics to guarantee that all edges are visited exactly once. The *Hierholzer* algorithm is based on Euler's theorem [Naito and Tsunoyama 1981] which produces a Eulerian Cycle.

Concepts and algorithms of graph theory can be applied in order to generate test cases, since state-transition diagrams are directed graphs that can be traversed [Arantes et al. 2014]. Considering the main features of the Switch Cover and H-Swich Cover criteria, they could be further explored and validated with respect to other algorithms for graphs, besides the Eulerian Cycle. While covering the same criterion, specific algorithms would produce different test sets with particular characteristics. Such diversity could be beneficial to the fault detection effectiveness [Shi et al. 2016]. Given this context, this study investigates other algorithms for graph to generate test cases from a Dual Graph. We present an experimental study that compares the Depth-First Search (DFS) and Breadth-First Search (BFS) algorithms with the Eulerian Cycle. In the analysis, we considered the number of test cases, the test suite size, the average length of sequences and generation time. In this evaluation, we adopted randomly generated complete machines, as well as real-world FSMs representing embedded software in space applications that fall into the category of reactive systems.

The remainder of this paper is organized as follows: Section 2 presents the main

definitions of FSMs, as well as related works. Section 3 presents the algorithms to generated Switch Cover test cases. Section 4 shows experiments we conducted. Section 5 shows the results with respect to experiments. Section 6 presents a general discussion to highlight some points of our research. Finally, Section 7 presents conclusions and future directions for this research.

2. Background

In this section, the main concepts of this study and related works are discussed briefly.

2.1. Definitions

This paper is based on deterministic Mealy machine models that consist of states and transitions. Each transition consumes an input symbol and, as a consequence, produces an output symbol. An FSM may also be modeled by a state diagram, which is a directed graph so that vertices are states and edges are transitions. The edges are labeled with inputs and outputs associated with the transition. Figure 1 shows an example of a state diagram representation of an FSM.

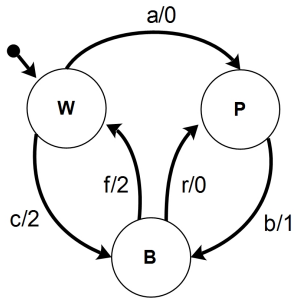


Figure 1. Example of an FSM.

The formal notations herein are based on [Dorofeeva et al. 2005, Endo and Simao 2013]. Formally, an FSM M is a 7-tuple $(S, s_0, I, O, D, \delta, \lambda)$, where:

- S is a finite set of states with initial state s_0,
- I is a finite set of inputs,
- O is a finite set of outputs,
- $D \subseteq S \times I$ is a specification domain,
- $\delta : D \to S$ is a transition function, and
- $\lambda : D \to O$ is an output function.

Tuple $(s, x) \in D$ is a *defined transition* in state s that consumes input symbol x. Form (s_i, x, y, s_j) may also be adopted to describe a transition from s_i to s_j that consumes input x and produces output y. In transition $t = (s_i, x, y, s_j)$, s_i is known as head state and s_j tail state; t is an incoming transition for s_j and t is an outgoing transition for s_i. An FSM which has defined transitions for each input symbol in all states, i.e., $D = S \times I$, is complete. Otherwise, the FSM is partial. A sequence $\alpha = x_1...x_k \in I^*$ is an input sequence defined for state $s \in S$, if there exist $s_1, ..., s_{k+1}$ such that $s = s_1$ and $\delta(s_i, x_i) = s_{i+1}$ for all $1 \leq i \leq k$. We say that α is a *transfer sequence* from s to s_{k+1} and that s_{k+1} is *reachable* from s. An FSM is strongly connected if every state is reachable

from all states. An FSM is initially connected if every state is reachable from initial state s_0.

Notation $\Omega(s)$ is used to denote all input sequences defined for state s and Ω_M as an abbreviation for $\Omega(s_0)$. Therefore, Ω_M represents all defined sequences for the FSM M. In this work, we assume that the FSM has a reset operation that brings the machine to its initial state. The reset operation is denoted by r. The transition and output functions are extended for defined input sequences, including the empty sequence ϵ, as follows. For a state $s_i \in S$, $\delta(s_i, \epsilon) = s_i$ and $\lambda(s_i, \epsilon) = \epsilon$, given an input sequence $\alpha x \in \Omega(s_i)$, we have $\delta(s_i, \alpha x) = \delta(\delta(s_i, \alpha), x)$ and $\lambda(s_i, \alpha x) = \lambda(s_i, \alpha)\lambda(\delta(s_i, \alpha), x)$.

A test case of M is an input sequence $\alpha \in \Omega_M$ starting with symbol r. A test set of M is a finite set of test cases of M.

This paper is focused on the Switch Cover criterion. Switch Cover is known as "*all combinations*", i.e. all pairs of transitions of an FSM must be covered [Pimont and Rault 1976]. The basis for Switch Cover is the "*De Bruijn*" sequence [De Bruijn 1946]. The criterion is formally defined as follows:

Given that $x', x'' \in I$ and $\alpha, \beta \in I^*$, a test set SW is a *switch cover* of M if, for each pair of transitions $(s_i, x'), (s_j, x'') \in D \mid \delta(s_i, x') = s_j$, there exists at least one test case $\alpha x' x'' \beta \in SW$ such that $\delta(s_0, \alpha) = s_i \wedge \delta(s_i, x') = s_j$.

2.2. Related Work

In the last years, the research in criteria/methods to generate test cases based on FSM has been focused on the comparison and improvement of existing methods in the literature.

Souza et al. [Souza et al. 2008] presented an empirical evaluation of cost and efficiency between two test criteria of the Statechart Coverage Criteria Family (FCCS) all-transitions and all-simple-paths criteria, and the Switch Cover criterion for FSMs. Mutation analysis was used to evaluate the criteria. Cost is defined according to the size of test suite and the time needed by a test suite to kill a mutant. Efficiency is related to the ability a test suite has to kill mutants (mutation score). In Simao et al. [Simão et al. 2009], the authors describe an experimental comparison among different coverage criteria for FSMs, such as state, transition, initialization fault, and transition fault coverage. Dorofeeva et al. [Dorofeeva et al. 2010] presented an overview and an experimental evaluation of FSM methods: DS, W, Wp, HSI, UIO, and H. They analyzed the criteria on different aspects, such as test suite length and derivation time. The experiments are conducted on randomly generated specifications and on two real-world protocols.

More recently, Endo and Simao [Endo and Simao 2013] presented an experimental study that compared the complete test sets generated automatically using the methods W, HSI, SPY, H, and P. They analyzed the number of test cases and their length, the total cost (i.e., the length) of each test suite, and the effectiveness of the methods using mutation testing for FSMs. In order to conduct the study, FSMs were randomly generated varying number of states, inputs, outputs, and transitions. In Souza et al. [Souza et al. 2015], an investigation of cost and efficiency was conducted comparing the FSM criteria: H-Swith Cover, UIO and DS. In order to analyze the test cases' efficiency, mutation analysis (mutation score) was adopted. With respect to cost, the size of the test set generated (number of events) was considered. The investigation was conducted considering two embedded

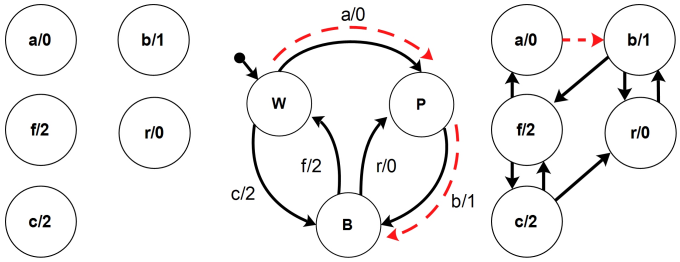
software products (space applications).

Analogously to these studies, we conducted an experimental comparison among different algorithms for graph to generate test cases from a Dual Graph obtained from a FSM.

3. Algorithms to Generate Switch Cover Test Sets

One of the main characteristics of Switch Cover criterion is to generate a graph from an FSM, known as Dual Graph. Formally, a graph is an ordered pair $G = (V, E)$ consisting of a set V of vertices (or nodes) and a set E of edges. Each edge in a graph connects two vertices [Bondy and Murty 2008].

From the Dual Graph, algorithms for graphs can be used to generate test sequences. The classical Switch Cover algorithm, for example, after obtaining the Dual Graph, has to balance the graph. Then the balanced graph is traversed, generating the test cases always starting at an initial vertex and returning to it. Using the FSM shown in Figure 1, the dual graph creation is described as follows:



(a) Constructing the graph (b) Edges creation from original FSM (c) New edges and vertices

Figure 2. Creation of the dual graph from the original FSM.

1. **Create vertices from the original FSM.** The transitions of the FSM must be converted into vertices (or nodes) (Figure 2(a)), in which W is the initial state. So, in this new graph, the new vertices "a" and "c" become the initial vertices for the algorithm, because they were the transitions leaving state W.
2. **Add vertices.** Based on the transitions of the original FSM, the graph vertices must be created. For example, in the original FSM, there is a transition *a* from state *W* to state *P* (Figure 2(b)), and there is a transition *b* leaving state *P*. Therefore, in the new graph that is being created, an edge is added connecting the new vertices *a* and *b* (Figure 2(c)). The same procedure is applied to all other pairs of transitions of the original FSM. Figure 2(c) shows the complete graph. After this process a directed graph is obtained. A directed graph or digraph is formed by vertices connected by directed edges.

Next, three algorithms, to generate switch cover test sets that are analyzed in this study, are presented.

(1) Eulerian Cycle from H-Switch Cover

In 2015, Souza et al. [Souza et al. 2015] created a new test criterion for FSM based on

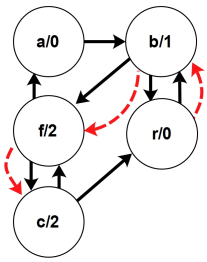


Figure 3. Balanced graph

original Switch Cover, H-Switch Cover. In H-Switch Cover criterion the main changes in the original criterion were the graph balancing and generation of test case steps.

When the graph is constructed, the polarities of the vertices must be balanced by constructing an Eulerian graph (Figure 3). In a Eulerian Graph there is a path where each edge is visited only once and a graph is Eulerian if there exists a closed trail (Eulerian Tour) containing all edges of E [Bondy and Murty 2008]. Balancing is obtained by duplicating the edges in such a way that the number of edges arriving is equal to the number of edges leaving the vertex (i.e. the degree of the vertex is zero). After the graph is balanced, it becomes a multigraph. A multigraph or pseudograph is a graph which is permitted to have multiple directed edges between the same pair of end nodes [Bang-Jensen and Gutin 2009].

After the graph balancing, the last step refers to the generation of test case based on the balanced graph (directed multigraph). This step of H-Switch Cover algorithm reflects the improvement of the quality of test cases in terms of coverage. H-Switch Cover uses *Hierholzer* heuristic to guarantee that all the edges are visited exactly once (Eulerian path). *Hierholzer* algorithm constructs an Eulerian path suggested from Euler's theorem proof. Euler's theorem says that a connected graph is Eulerian if and only if each vertex (transition arc) has the same degree [Neumann 2004, Bondy and Murty 2008]. Once a graph is balanced, one can apply *Hierholzer* algorithm and generate an Eulerian Cicle. The main steps for implementing the algorithm are:

1. Start with any edge of the initial vertex and select edges not yet visited until a cycle is closed;
2. If there are still any unvisited edges, start with an edge that is a part of an already existing cycle and create a new cycle as in the first step; and
3. If there are no more edges to be visited, an Eulerian Cycle must be constructed from the existing cycles, joining them from a common edge.

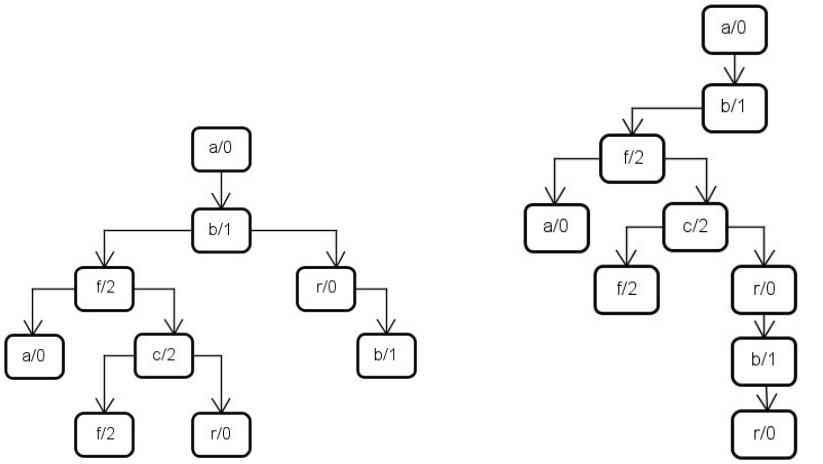
In the example, considering the original FSM, the initial state was *W*, then, in the graph created, when the transitions *a/0* and *c/2* are transformed into vertices, these become initial vertices in the new graph. The following test cases are generated: *abrbfcrbf* and *crbrbfabf*. Test cases, in this example, is presented only input data in the test case.

H-Switch Cover relies on the traditional Switch Cover test criterion, but H-Switch Cover uses new heuristics to improve its performance, for example, adoption of rules to optimize graph balancing and traverse the graph for test cases generation. With the implementation of the new approach (rules) to balance the vertices, there was an improvement of the quality of test cases in terms of size, since it optimizes the balancing.

(2) Breadth-First Search (BFS) and (3) Depth-First Search (DFS)

In graph theory, BFS is a search algorithm in graphs used to perform a search or traverse a graph in spanning tree format. It starts at a root vertex and explores all the neighboring vertices before moving to the next level neighbors. Similar to BFS, DFS is also an algorithm for traversing a graph. DFS visits each vertex of a graph, starting at a root vertex. Then, it traverses as far as possible along each branch before backtracking, that is, the algorithm moves along a spanning tree of that graph.

Considering the graph of Figure 2(c), trees are generated to represent the walk of the BFS and DFS algorithms from vertex *a/0*. Figures 4(a) and 4(b) present the trees for BFS and DFS, respectively. The generated test sequences from the **BFS** are: *abfa*, *abfcf*, *abfcr*, *abrb*. Test sequences from the **DFS** are: *abfa*, *abfcf*, *abfcrbr*.



(a) BFS (b) DFS

Figure 4. Trees generated from BFS and DFS algorithms

4. Experimental Study

As mentioned, this study investigates algorithms for graph to generate test cases from a Dual Graph obtained from FSMs. We present an experimental study that compares the approaches based on the DFS and BFS algorithms with the H-Switch Cover (Eulerian Cycle). We evaluated the three algorithms with 900 randomly generated FSMs and twenty one machines of real-world applications. Figure 5 presents the overview of the conducted experiment.

We adopted reduced and deterministic FSMs randomly generated in [Endo and Simao 2013]. We selected machines with four inputs, four outputs and the number of states ranging from four to 20. For each FSM configuration, 100 different machines were adopted and the average measures were calculated aiming to reduce the influence of particular FSM characteristics on the results. In total, 900 randomly generated FSMs were analyzed in the study.

With respect to the real-world FSMs, two software products embedded into computers of space projects under development at the *Instituto Nacional de Pesquisas Espaciais* - INPE (National Institute for Space Research) were used in this experiment. The

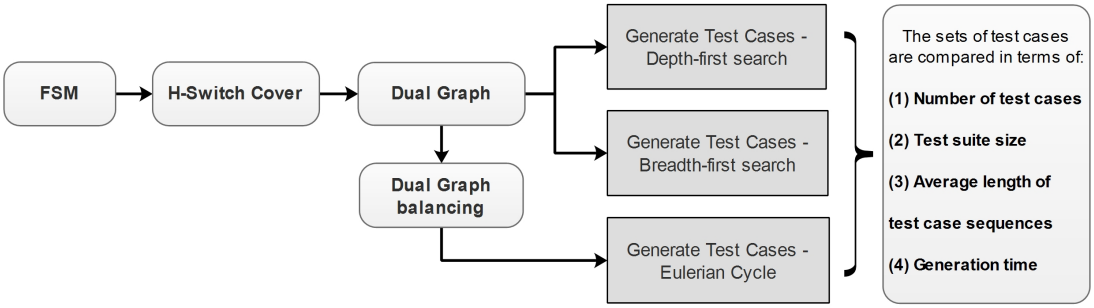


Figure 5. Experiment Overview

first software product is *Alpha, Proton and Electron monitoring eXperiment in the magnetosphere* (APEX). APEX is the software product embedded into an astrophysical experiment computer of a Brazilian space project. The second software product is *Software for the Payload Data Handling Computer* (SWPDC), which was developed in the context of the Quality of Space Application Embedded Software) (QSEE) research project [Santiago et al. 2007]. This software has been currently adapted to on-board computer of a balloon-borne high energy astrophysics experiment under development at INPE.

Twenty one scenarios developed for the two software products were evaluated. The first model refers to the first project, while the other twenty models refer to the second project. The latter is considered large and more complex, therefore, it implied in creating several use cases that addressed the scenarios it contains. Details for the two software products used and their FSMs can be found in [Souza et al. 2015].

With respect to the evaluation of test methods, usually, the size of test suites has been adopted as a measure. Therefore, the cost is proportional to test suite size that can be measured by counting the number of test cases in a test set. For this work, in order to analyze the test cases generated with respect to cost, we analyze measures concerning the test suites generated for each method and the generation time, defined as follows:

- **Number of test cases**. We consider the number of test cases as the number of resets in a test suite. This also represents the number of test cases since there is a reset at the beginning of each test case.
- **Test suite size**. We consider the number of input events as the test suite size. When an FSM has a state activated as a response to an input event, this event may cause a change of state, and may produce an output action.
- **Average length of sequences**. This considers the average length of test cases sequence, that is, average number of input events in a test case sequence.
- **Generation time**. This refers to the time (in milliseconds) needed to generate a test suite. Since many external factors can affect time spent to generate test cases, each configuration was executed 100 times and the average generation time was taken.

5. Analysis of Results

5.1. Results for randomly generated FSMs

Number of test cases. Figure 6 shows how the number of test cases varies as a function of the number of states. The Eulerian cycle algorithm from H-Switch Cover criterion

always presents a smaller number of test cases since it generates a single path in a graph starting at an initial vertex and returning to it. More test cases will be generated if there exists other initial vertices in the graph. On the other hand, considering the BFS and DFS algorithms, we observed that the number of test cases grows as the number of states increases. The BFS and DFS algorithms show an approximate linear growth.

In the construction of BFS spanning tree, as shown in Figure 4, the number of leaf nodes increases as graph size increases. Therefore, there is always a higher number of leaf nodes than in the DFS spanning tree, and, consequently the number of test cases for BFS will be relatively larger than the DFS.

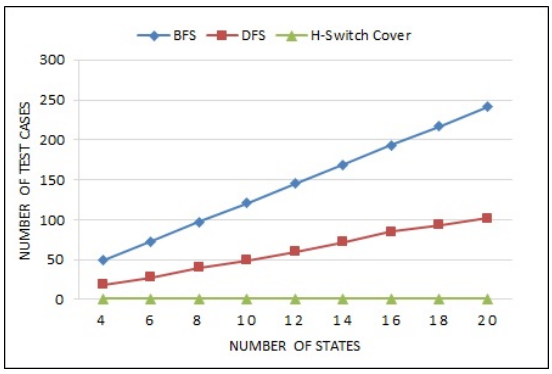


Figure 6. Number of test cases varying the number of states.

Test suite size. Figure 7 shows how the test suite size varies in function of the number of states. For the three algorithms, the test suite size grows as the number of states increases. However, while BFS and the H-Switch Cover criterion grow linearly, the DFS algorithm has shown a polynomial growth according to the number of states. As expected the H-Switch Cover criterion achieved the best results, and this is due to the fact the criterion has recently undergone improvements in optimization of graph balancing step and reducing the number of input events, that is, the test suite size [Souza et al. 2015].

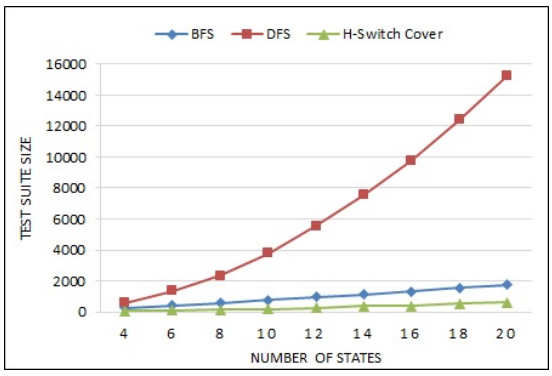


Figure 7. Test suite size varying the number of states.

Average length of sequences. Figure 8 shows how the average length of test case sequences varies in function of the number of states. Since the H-Switch Cover criterion generates a unique sequence, the sequence size tends to grow as it has more states (same

behavior shown in Figure 7). As BFS and DFS do not restrict the number of sequences, the average test case size grows slowly as the number of states increases. Based on the how both the algorithms traverse in the graph (as viewed in the trees in Figure 4), BFS produces the shortest sequences, while DFS generates a slightly longer sequences.

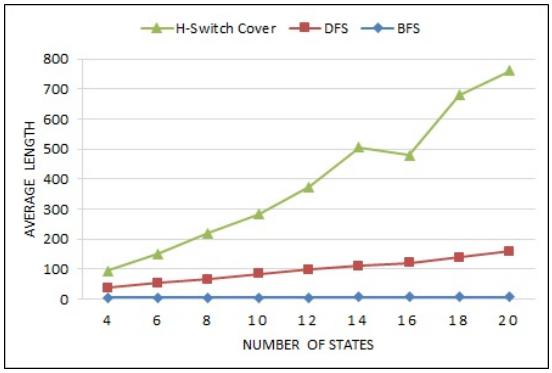


Figure 8. Average size sequence varying the number of states.

Generation time. Figure 9 shows how the generation time of test cases varies in function of the number of states. Both BFS and DFS algorithms have a low variation with respect to the test cases generation time ($\approx$ 1.7 milliseconds). However, H-Switch Cover criterion has an ever-increasing execution time, as the number of states grows. This happens because the H-Switch Cover criterion has a graph balancing step which is traversed using a Eurelian cycle algorithm to generate the test cases. This is an expensive process, both the balancing process and Eulerizing the graph, since new edges can be created in the graph. Furthermore, the new edges can extend primarily for complex FSMs, which can cause a significant computational effort.

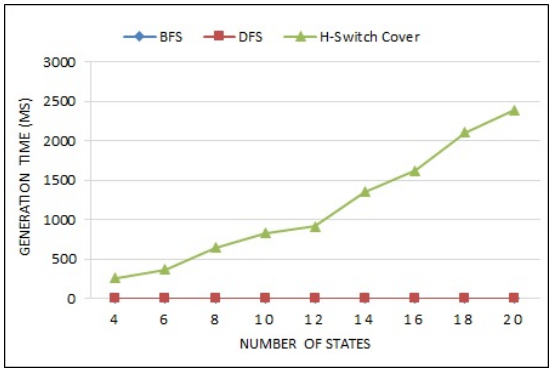


Figure 9. Generation time varying the number of states.

5.2. Results of real-world FSMs

Table 1 shows the analyzed results for the first software products used as a case study, APEX. Comparing with the random FSMs results, APEX results are very similar. H-Switch Cover criterion achieved the best results in terms of number of test case and test suite size since it has only one test case. Another similar result is the average length of sequences, where BFS algorithm has a better result than others, showing the lower

average of sequences. Futhermore, considering the generation time to generate the test case, the H-Switch Cover continues much higher than the BFS and DFS algorithms.

Table 1. Analysis results for APEX

	BFS	DFS	H-Switch Cover
Number of test case	87	85	1
Test suite size	554	886	370
Average length of sequences	5,4	9,4	369
Generation time	16,8	19,5	305,5

Regarding the second case study, Figure 10 shows the results for SWPDC. We generated an average of twenty FSMs that represent the different SWPDC scenarios. For all measures analyzed, the H-Switch Cover criterion continues to show results similar to the experiment with random FSMs and the software APEX. On the other hand, the BFS and DFS algorithms presented few differences. Regarding the number of test cases, BFS continues to present a larger number relative to the others. However, unlike the random FSMs, in a real scenario the BFS algorithm showed the highest value for the test suite size. Regarding the time for generation of test cases, there was not much difference between the algorithms analyzed since the number of FSMs is small to infer in this analysis.

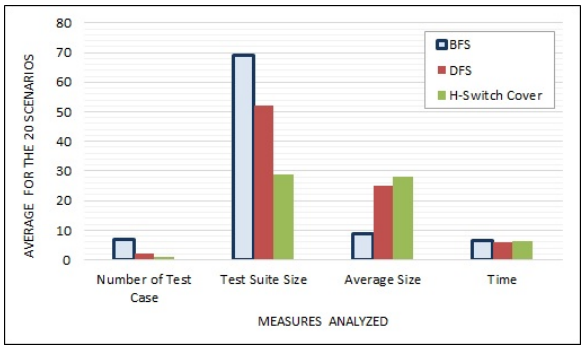
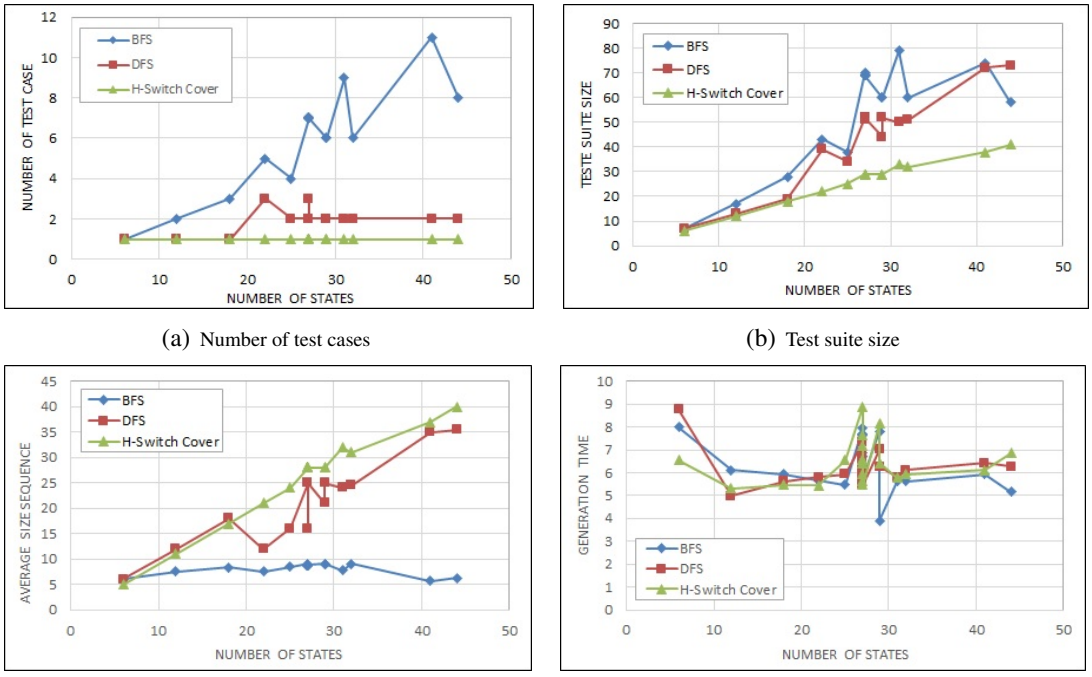


Figure 10. Measures for SWPDC.

In order to visualize and better understand the variation of values of the each measure in function of the number of states in the 20 FSMs of SWPDC, the Figure 11 shows the graphics of analysis.

In Figure 11(a) considering the BFS algorithm, the number of test cases grows as the number of states increase more than other algorithms, but this does not occur linearly as in the experiment conducted with the random FSMs. This occurs because the number of states in each FSM of the SWPDC do not follow the same growth pattern, for example, the ascending order for the first six states in SWPDC is: 6, 12, 18, 22, 25, 27. Furthermore, in the Dual Graph creation in the real FSMs, there is a greater number of initial vertices, which can influence the number of test cases.

In relation to the test suite size, even with a different results of the random FSMs, in the Figure 11(b) we observed a similar increase of proportionality between BFS and DFS algorithms. The H-Switch Cover criterion showed the better results with lower set of test suite size and it increasing proportionally to the increase in number of states. Similarly



(a) Number of test cases

(b) Test suite size

(c) Average size sequence

(d) Generation time of the test case

Figure 11. Variation of values of the measures in function of the number of states in the 20 FSMs of SWPDC

to the number of test cases, a greater number of initial vertices can influence the results increasing the test suite size.

In the analysis of the average length of sequence (Figure 11(c)), the result is the same as the random FSMs, but the DFS algorithm presents a higher growth as increase number of states when compared with the results of DFS algorithm in random FSMs. With respect to the time for generating test cases (Figure 11(d)), the results of the three algorithms are very close, as already mentioned, due to the small number of FSMs in SWPDC which makes it difficult to infer an accurate result.

6. Discussion

FSMs are important to represent reactive systems so that applied and observed events model transitions that modify the states of a system. The graph theory techniques also allow us to use the behavioral information stored in models. Models are an excellent way to represent and understand system behavior, and they provide an easy way to generate test case(s).

As previously described, among the existing methods/criteria to generate test cases from a model, Switch Cover is an old criterion and it has been investigated for a long time. The main feature of the Switch Cover is the conversion of FSM in a Dual Graph to ensure that every branch-to-branch pair from the graph is covered. The latest version criteria created from the Switch Cover is the H-Switch Cover, which showed improvement in the algorithm, mainly in the test case generation that produces the Eulerian Cycle. However, specific algorithms would produce different test sets with particular characteristics. Given this context, this study investigated different algorithms applied to

a graph to generate test cases from a Dual Graph.

The results presented in this paper can assist a test expert to identify new approaches to generate test cases by means of graphs. Among the main results we identified that in terms of number of test case and test suite size, H-Switch Cover criterion had the best results (Figures 6 and 7). Note that (the) DFS algorithm has a very different result from BFS and H-Switch Cover when comparing the test suite size. DFS provides a polynomial growth as the number of states increases. However, considering the average size of test case sequence (Figure 8), the DFS and BFS algorithms can be considered better, mainly the BFS. On the other hand, considering the time to generate test cases (Figure 9), H-Switch Cover represents the worst results.

Many studies in the literature do not use FSMs of real systems to compare the test sets generated by different methods. In this experiment beyond the different FSMs randomly generated, we also used FSMs of real-world systems. Comparisons like we showed in this research, where complex and real applications are assessed, are valuable toward the improvement of the state of the art.

The different analyses about the methods' behavior can support the choice of the best method to fit in a given project. The test expert can identify the trade offs between test case characteristics. For instance, the tester may want a method that produces a shorter number of test case(s). In this case, H-Switch Cover criterion would be the best choice. On the other hand, it has a higher cost compared to the test case generation time. Therefore, it is at the discretion of the testing specialist to decide which algorithm can be used considering the experiment results and characteristics of the system being tested.

7. Conclusion and Future Work

This paper has presented an experimental evaluation of test case generation methods for FSMs. DFS, BFS and Eulerian Cycle from H-Swith Cover criterion were analyzed. In this analysis, we adopted 900 randomly generated complete machines and twenty one real-world FSMs representing embedded software in space applications that fall into the category of reactive systems. Overall, the three algorithms studied showed similar results when compared to real-world FSMs and random FSMs. In terms of numbers of test cases and test suite size, the H-Switch Cover criterion provides the best results. However, considering the average test case length, the BFS algorithm presents the best result. Finally, in terms of time for generating test cases, the BFS and DFS algorithms have the best performance.

Switch Cover is considered an important criterion, as it has been used in critical domains. This is shown in the case studies on embedded software in scientific equipment on board satellites, presented in this paper. In a domain where test models can become large and complex, it is important for the tester to know other options for generating test cases. In the case of Switch Cover or H-Switch Cover, there are several forms (algorithms) to generate test sets covering such criteria. The results in this work may provide support for a test expert to make a decision on which is the best (suitable) option to be employed based on the quality attributes of the system under test.

Conducting research on the behavior of methods for FSMs is necessary and has been the object of many investigations. However, there is still a lot to be done on this

aspect. Future directions include evaluation of other graph algorithms, use of other FSMs with different configurations and the conduction of others analyses, for example, the efficiency analysis of test cases for defect detection.

Acknowledgment

The authors would like to thank CNPq (PIBIC) for the financial support. The author Andre Takeshi Endo is partially financially supported by CNPq (Grant Number 445958/2014-6).

References

Arantes, A., Santiago, V. A. J., Vijaykumar, N. L., and Souza, E. F. (2014). Tool support for generating model-based test cases via web. *International Journal of Web Engineering and Technology*, 9:62–96.

Arantes, A. O., Vijaykumar, N. L., Santiago, V. A. J., and Guimarães, D. (2008). Test Case Generation for Critical Systems through a Collaborative Web-based Tool. In *International conference on inovation in software engineering (ISE)*, pages 163–168.

Bang-Jensen, J. and Gutin, G. Z. (2009). *Digraphs: Theory, Algorithms and Applications*. Springer, 2 edition.

Bondy, J. and Murty, U. (2008). *Graduate Texts in Mathematics: Graph Theory*. Springer, USA.

De Bruijn, N. G. (1946). A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49:758–764.

Dorofeeva, R., El-Fakih, K., Maag, S., Cavalli, A., and Yevtushenko, N. (2010). FSM-based conformance testing methods: a survey annotated with experimental evaluation. *Information and Software Technology*, 52:1286–1297.

Dorofeeva, R., El-Fakih, K., and Yevtushenko, N. (2005). An improved conformance testing method. In *International Conference on Formal Techniques for Networked and Distributed Systems (FORTE)*, pages 204–218.

El-Far, I. K. and Whittaker, J. A. (2001). Model-based software testing. In *Encyclopedia on Software Engineering*, pages 825–837.

Endo, A. and Simao, A. (2013). Evaluating test suite characteristics, cost, and effectiveness of FSM-based testing methods. *Information and Software Technology*, 55(6):1045–1062.

Gill, A. (1962). *Introduction to the Theory of Finite-State Machines*. McGraw-Hill, New York.

Guney, G. (1970). A method for the design of fault detection experiments. *IEEE Transactions on Computers*, 19:551–558.

Martins, E., Sabiao, S. B., and Ambrosio, A. M. (1999). ConData: a tool for automating specification-based test case generation for communication systems. In *International Conference on System Sciences*, pages 303–319.

Naito, S. and Tsunoyama, M. (1981). Fault detection for sequential machines by transition tours. In *Fault Tolerant Computing Conference (FTCS)*, pages 238–243.

Neumann, F. (2004). Expected runtimes of evolutionary algorithms for the eulerian cycle problem. In *Congress on Evolutionary Computation*, pages 904–910.

Petrenko, A. and Yevtushenko, N. (2005). Testing from partial deterministic FSM specifications. *IEEE Transactions on Computers*, 54:1154–1165.

Pimont, S. and Rault, J. (1976). A software reliability assessment based on a structural and behavioral analysis of programs. In *International Conference on Software Engineering (ICSE)*, pages 486–491.

Pinheiro, A. C., Simão, A., and Ambrosio, A. M. (2014). FSM-Based Test Case Generation Methods Applied to Test the Communication Software on Board the ITASAT University Satellite: A Case Study. *Journal of Aerospace Technology and Management*, 6:447–461.

Pontes, R. P., Véras, P. C., Ambrosio, A. M., and Villani, E. (2014). Contributions of model checking and CoFI methodology to the development of space embedded software. *Empirical Software Engineering*, 19:39–68.

Sabnani, K. K. (1988). A protocol test generation procedure. *Computer networks and ISDN systems*, 15:285–297.

Santiago, V., Mattiello, F., Costa, R., Silva, W. P., and Ambrósio, A. M. (2007). QSEE project: An experience in outsourcing software development for space. In *Int. conference on software engineering and knowledge engineering*, pages 183–188.

Santiago, V. A. J. (2011). *SOLIMVA: A methodology for generating model-based test cases from natural language requirements and detecting incompleteness in software specifications*. PhD thesis, National Institute for Space Research (INPE), Doctorate at Post Graduation Course in Applied Computing. São José dos Campos, SP, Brazil.

Shi, Q., Chen, Z., Fang, C., Feng, Y., and Xu, B. (2016). Measuring the Diversity of a Test Set With Distance Entropy. *IEEE Transactions on Reliability*, 65(1):19–27.

Sidhu, D. P. and Leung, T. (1989). Formal methods for protocol testing: A detailed study. *Transactions on Software Engineering*, 15:413–426.

Simão, A., Petrenko, A., and Maldonado, J. (2009). Comparing finite state machine test coverage criteria. *IET Software*, 3:91–105.

Simão, A., Petrenko, A., and Maldonado, J. (2010). Fault coverage-driven incremental test generation. *Computer Journal*, 53:1508–1522.

Souza, E. F., Santiago, V. A. J., Guimar aes, D., and Vijaykumar, N. L. (2008). Evaluation of test criteria for space application software modeling in statecharts. In *International conference on innovation in software engineering*, pages 157–162.

Souza, E. F., Santiago, V. A. J., and Vijaykumar, N. L. (2015). H-Switch Cover: a new test criterion to generate test case from finite state machines. *Software Quality Journal*, pages 1–33.

Utting, M. and Legeard, B. (2006). *Practical Model-Based Testing*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

Zhang, X., M., Y., Geng, G., and Luo, W. (2011). A DFSM-Based Protocol Conformance Testing and Diagnosing Method. *Informatica*, 22:447–469.