Gray-Box Combinatorial Interaction Testing

by

Arsalan Javeed

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

Sabancı University

January 2015

GRAY BOX COMBINATORIAL INTERACTION TESTING

APPROVED BY:

Asst. Prof. Dr. Cemal Yılmaz
(Thesis Supervisor) ....

Prof. Dr. Erkay Savaş ....

Assoc. Prof. Dr. Tonguç Ünlüyurt ....

DATE OF APPROVAL: 05/01/2015

Gray-Box Combinatorial Interaction Testing

Arsalan Javeed

Computer Science and Engineering, MS Thesis, 2015

Thesis Supervisor: Asst. Prof. Cemal Yılmaz

Keywords: Software quality assurance, static analysis, combinatorial interaction testing, covering arrays, highly configurable softwares, configuration options

Abstract

The enourmous size of configuration spaces in highly configurable softwares pose challenges to testing. Typically exhaustive testing is neither an option nor a way. Combinatorial interaction techiques are a systematic way to test such enourmous configuration spaces by a systematic way of sampling the space, employed through covering arrays. A t-way covering array is a sampled subset of configurations which contains all t-way option setting combinations. Testing through t-way covering arrays is proven to be highly effective at revealing failures caused by interaction of t or fewer options. Although, traditional covering arrays are effective however, we've observed that they suffer in the presence of complex interactions among configuration options, referred as tangled options. A tangled configuration option is described as either a configuration option with complex structure and/or nested in hierarchy of configuration options. In this thesis, we conjecture the effectiveness of CIT in the presence of tangled options can greatly be improved, by analyzing the system's source code. The analysis of source code reveals the interaction of configuration options with each other, this information can be used to determine which additional option setting combinations and the conditions under which these combinations must be tested. Gray-box testing methods rely on partial structural information of the system dur-

ing testing. We've statically analyzed the source code of subject applications to extract the structure and hierachy of configuration options. Each configuration option has been structurally tested according to a test criterion against a t-way covering array and subsequently their t-way interactions. The criterion revealed the missing coverage of options which were employed to drive the additional testcase generation phase to acheive complete coverage. We present a number of novel CIT coverage criteria for t-wise interaction testing of configuration options. In this thesis, we've conducted a series of large scale experiments on 18 different real-world highly configurable software applications from different application domains to evaluate the proposed approach. We've observed that traditional t-way CAs can provide above 80% coverage for configuration options testing. However, they significantly suffer to provide interaction coverage under high t and tangling effects where coverage is dropped to less than 50%. Our work address these issues and propose a technique to acheive complete coverage.

Gri-Kutu Kombinatoryal Etkileşim Testi

Arsalan Javeed

Bilgisayar Bilimleri ve Mühendisliği, Yükseklisans Tezi, 2015

Tez Danışmanı: Yrd. Doç. Cemal Yılmaz

Anahtar Kelimeler: Yazılım kalite güvencesi, statik analiz, kombinatoryal etkilešim testi, kapsayan diziler, yapılandırılabilirliği yüksek yazılımlar, konfigürasyon seč̌enekleri

Özet

Čok fazla sayıda konfigürasyon seč̌eneği olan yapılandırılabilirliği yüksek yazılımların test edilmesinin zorlukları vardır. Kombinatoryal etkilešim teknikleri, kapsayan dizileri kullanarak yüksek düzeyde yapılandırılabilir sistemleri sistematik bir šekilde test etme yöntemidir. Bir t-yollu kapsayan dizi, bütün t-yollu konfigürasyon seč̌enek değerleri kombinasyonunu en az bir kere kapsayan bir konfigürasyon kümesidir. t-yollu kapsayan dizi kullanılarak test etmenin t veya daha az seč̌eneğin etkilešiminden kaynaklanan hataları ač̌ığa č̌ıkarmada yüksek etkisinin olduğu ampirik č̌alıšmalarla gösterilmištir. Geleneksel kapsayan diziler etkili olsa bile, konfigürasyonlarında seč̌enekleri arasında komplex etkilešimler olduğunda geleneksel kapsayan dizilerin zorlandıklarını gördük. Bu gibi durumlara dolašık (tangled) seč̌enekler diyoruz. Bir dolašık konfigürasyon seč̌eneği kompleks yapıda bir küme konfigürasyon seč̌eneği ile ve/veya ič̌ ič̌e geč̌miš konfigürasyon seč̌enekleri hiyeraršisi ile gösterilebilir. Bu tezde, dolašık seč̌eneklerin olduğu sistemlerin kaynak kodları incelenerek kombinatoryal etkilešim testlerinin etkisinin önemli bir bič̌imde gelištirilebileceği hipotezine sahibiz. Kaynak kodunun analiz edilmesi, konfigürasyon seč̌eneklerinin birbirleri arasındaki etkilešimin ač̌ığa č̌ıkartılmasında ve fazladan hangi seč̌enek kombinasyonlarının ve bu kombinasyonların hangi košullarda test

edileceğinin bulanmasında kullanılır. Gri kutu test metodları, test edilen sistemlerin yapısal bilgilerine ihtiyač duymaktadır. Konfigürasyon sečeneklerinin yapısını ve hiyeraršisini čıkarmak ičin statik olarak test edilecek sistemlerin kaynak kodlarını analiz ettik. Her konfigürasyon sečeneği bir test kriterine göre yapısal olarak bir kapsayan dizi tarafından ve ardından t-yollu etkilešimleri test edildi. Bu kriter, tam bir kapsama elde etmek yolunda eksik kalan konfigürasyon sečenekleri kombinasyonlarını belirlemede kullanılır. Daha sonrasında bu eksik kombinasyonlar ičin ek test durumları üretilir. Biz t-yollu kofigürasyon sečenekleri etkilešimi ičin bir dizi yeni kombinatoryal etkilešim kriterleri sunuyoruz. Bu tezde, sunduğumuz metodu ölčmek ičin yapılandırılabilirliği yüksek 18 gerček yazılım üzerinde geniš čapta deneysel čalıšmalar gerčekleštirdik. Geleneksel t-yollu kapsayan dizilerin konfigürasyon sečenekleri testinde sadece %80'ler civarında kapsama sağlayabildiğini gözlemledik. Ayrıca, t'nin yüksek değerlerinde ve dolašıklığın fazla olduğu yerlerde kapsama %50'nin altına düštü. Bu tezde önerilen metod, bu tarz sorunları hedef almaktadır ve tam bir kapsama elde etmek ičin bir teknik sunar.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Prof. Cemal Yılmaz for his continuous support of my master study and research, support, patience, motivation, enthusiasm and indepth knowledge. His guidance helped me in through out time of research and writing of this thesis.

I am deeply indebted and grateful for him being a great advisor and my research mentor for this master's study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Erkay Savaş and Prof. Tonguc Unluyurt for their encouragement, critical comments and feedback.

I thank my fellow labmates in Software Research Group: Hanefi Mercan, Gulsen Demiroz and Ugur Koc for their insightful comments, discussions and co-operations. Also I thank my friends in Sabanci University: Ahmed Nouman, Rahim Dherkhani, Ece Egemen and Emre Inal for some of the memorable moments during this span of time.

Last but not the least, I would like to thank my mother and my family members for their constant love, support and best wishes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CS	Computer Science.
CA	Traditional Covering Array.
CIT	Combinatorial Interaction Testing.
CCA	Complemented Covering Array.
FIT	Full Interaction Coverage Test suite
VO	Virtual Option.
AO	Actual Option.
HCS	Highly Configurable Software.
SUT	Software Under Test.
SQA	Software Quality Assurance.
CC	Condition Coverage.
DC	Decision Coverage.
MC/DC	Multiple Condition and Decision Coverage.
Cyclo	Cyclometic Complexity.
Conf. Opt	Configuration Option.

1

INTRODUCTION

Over the last few years software development has seen a trend shift from the production of individual programs to the production of families of related programs. The driving reason for the shift was the convenience of design and implementation of multiple software systems, equipped with a core set of common capabilities but often have significant differences, achieved by significant reuse of core and optional components through the implementation of configurable features as one unified system, often referred as, *highly configureable software systems (HCS)*. The process of configuration is referred as process of weaving of the optional features to the actual end-user software realization [9].

The notion of highly-configurable software systems has emerged in many different contexts spanning across the spectrum of hardware and software systems such as power distribution systems through OS kernels. The points of variations in a HCS allows the developers to insert different feature variations within the bounds of the subject software architecture. In HCS the high configurability can be either static or dynamic reconfigurability. Static reconfigurability is a way to configure a system at compile time where the system is configured as a part of build process. In constrast, dynamic reconfigurability is a way of configuring the system at runtime.

Among some of challenges in HCS development paradigm, one significant challenge is its testing. The testing effort consist of testing each configuration in a set of derived systems. Although, its desireable to achieve 100% test coverage, testing each configuration is infeasible in practice, since the configuration space is too vast to test. One of the main reason being the number of all possible configurations given a set of configuration features, observes an exponential growth. A similar challenge is, a single test case might run without failing in one derived system but might fail in some other derivation.

An acceptable cost and time constrained solution is desirable, aiming to provide a confidence in test coverage. One of the approach called combinatorial interaction testing (CIT) is employed in many domains and also supported by a range of available tools. The CIT systematically samples the configuration space of the software and test only the selected configurations. The CIT approach is employed by first defining the configuration space model of the subject system i.e. set of valid ways it can be configured. The configuration model includes a set of configuration options that can take a defined number of option settings in addition to system wide interoption constraints, which eliminates invalid configurations. The CIT technique based on this model generates a small set of valid configurations referred as a *t-way covering array*. The t-way covering array contains each possible combination of option settings for every combination of t options, at least once. The system is tested by running its testsuite for each configuration in the covering array [40].

In configurable software systems there are different variability mechanisms employed for adding configurability, for instance c-pre-processor(cpp) for C based systems. In our work we've taken C based HCS as our subject applications whose configurability is based on cpp macros. The cpp macros based configurability is implemented through *if-else* based constructs.

Kuhn et.al [21] observed most of the faults are caused by interaction between a small set of options, which can be revealed by testing the interactions of these options, which led to the notion of t-wise testing. They have observed over 80% of interaction faults can be revealed by 2-way and 3-way interactions of options whereas 6-way interactions can reveal 99% of such faults. However, traditional CIT techniques assume independence of these options but in real world scenarios although if not all most options can be independent but the effect of rest of the options are dependent on specific settings of other options.

Hierarchy	**cOpt**	
	id.	structure
if(o1||o2){	1	*(o1||o2)*
if(o2){	2	*(o2)*
}else{		
if(o3&&!o4){	3	*(o3&&!o4)*
}else{		
if(o5||o2){	4	*(o5||o2)*
}}}}		
if(o3&&o5){	5	*(o3&&o5)*
}		
if(o4){	6	*(o4)*
}		
Options	*o1,o2,o3,o4,o5*	

Table 1.1: Code Listing and Configuration Options

Consider a hypothetical code listing in Table1.1, inspired from one of our subject application, comprising a set of if-else constructs based on c-preprocessor macro implementations. Each if construct can either contain, a feature code or contains one or more nested if-else structures nesting feature code or *both*. The execution of any if or else block depends upon the evaluation of conditional boolean expression comprised one or more options. The actual configuration options for this listing are *o1*,*o2*,*o3*,*o4*,*o5*. These binary options can only take either *true* or *false* values. However, the configuration options are boolean expressions comprised on these binary options. For instance, configuration option *cOp1=(o1||o2)*, as presented in Table.1.1 so and so forth.

Table 1.2 and Table 1.3 presents a 2- and 3-way covering array (CA) comprised on 5 binary options for this example, having no inter-option constraints. Each row of the covering array represent a test configuration.

Each configuration option (cOpt) can be tested against one of the configuration of the t-way covering array. Here the test objective is to exercise each cOpt to its both possible outcomes *true* and *false*.

Option tangling is the effect where one configuration option is nested inside another configuration option. If a configuration option is not tangled it can be effectively exercised during testing. For instance, in Table1.1, cOpt1 can be exercised to both *true* and *false* against the following configurations: (FTxxx,TFxxx,FFxxx,TTxxx) of 2-way or 3-way CAs of Table 1.2 and Table 1.3. Since, cOpt1 has been effectively exercised to its all possible outcomes, in this case it is completely tested.

However, on the contrary, cOpt4=(o5||o2) is tangled inside cOpt1,cOpt2 and cOp3. cOpt4 can only be exercised if there exists such a configuration in the covering array that can 'set' the guarding configuration options to a specific setting so that, cOpt4 can be accessed. To test cOpt4 the guarding configuration options must be set to the following values cOpt1=*true*, cOpt2=*false*, cOpt3=*false*. In this case any configuration that has the following structure *TFFFx* can provide this setting combination. However in both 2- and 3-way CAs has one such configuration *TFFFF* which can only exercise cOpt5 to *false* in effect only testing it 50%. For complete testing configuration *TFFFT* is missing. In effect, CA is being suffered to provide complete coverage under the effect of option tangling.

The testing effort in this scenario gets more challenging if the requirements of test criterion are complex. For instance, if the test criterion demands to completely exercise each option in a configuration option, there may not be the full set of configurations in the CAs that can meet those requirements. Usually, CAs can't meet complete coverage requirements for all configuration options and referred as coverage sufferings. In such, scenario CA will even suffer more.

For a set of configuration options in a test subject, testing against a t-way covering array under a test criterion, we've observed that CAs suffers. The sufferings are directly proportional to the stuructural complexity of configuration options, tangling of configuration options and complexity of test criterion. Lack of test coverage, cannot completely

o1	o2	o3	o4	o5
F	T	T	T	T
T	F	F	F	F
F	F	F	T	F
T	T	T	F	F
T	T	F	F	T
F	F	T	F	T
T	F	T	T	T

Table 1.2: A 2-way covering array for 5 binary options

o1	o2	o3	o4	o5
F	T	T	T	T
T	F	F	F	F
F	F	F	T	F
T	T	T	F	F
T	T	F	F	T
F	F	T	F	T
T	F	T	T	T
F	T	F	F	F
T	T	F	T	F
F	F	F	T	T
F	F	T	T	F

Table 1.3: A 3-way covering array for 5 binary options

exercise all the interactions of configuration options which do not expose all faulty interactions a system can possess. Often critical failures remain masked inside the system. This can be especially a serious issue for application used in safety critical domains such as health.

In this work we've developed a gray-box based t-way combinatorial interaction testing (CIT) approach to cover structural and t-way interaction testing of configuration options, under the effect of tangling, structural complexity on a set of existing real world highly configurable software systems for three adequacy criterion, based on static analysis of the software systems under study. Our motivation behind this work was the investigation and remedy of the suffering of coverage provision in traditional covering arrays under the effects of structural complexity and option tangling. Our initial hyphotheses was traditional coverage arrays can provide adequate coverage for structural testing of configuration options and they can provide significant test coverage, when options are untangled. However, CAs can not provide adequate coverage for testing interactions of tangled options. The results that we've obtained in this study strongly supports our inititial hypothesis. The experiments we've performed and the results that we've obtained strongly support our proposed approach.

In this work we've made the following contributions:

- Empirical demonstration of coverage suffering by t-way traditional CAs for the structural and t-way interaction testing of configuration options in the presence of tangling.
- Introduced a number of novel interaction coverage criteria that can test *structure* and *interaction* of tangled options, based on static analysis of subject application.
- Developed a gray-box approach to acheive complete test coverage under the guidance of our novel criterion.
- Performed large scale experiments on highly configurable real-world applications to investigate and remedy this problem.

The remainder of the article is organized as follows: Chapter 2 provides *Background Information*, Chapter 3 provides information about *Related Work*, Chapter 4 provides information about *Approach*, Chapter 5 about *Experiments*, Chapter 6 discusses *Threats to Validity* and Chapter 7 discussed about *Conclusions and Future Works*.

2

BACKGROUND INFORMATION

This chapter provides information about traditional covering arrays, interaction testing, gray-box testing and structural coverage criterions.

2.1. Combinatorial Interaction Testing (CIT)

Combinatorial Interaction Testing (CIT) is a software testing technique aimed to reveal interaction-faults which are exposed through the interaction of various configuration options of the subject system. Most modern softwares often typically employ tens to hundreds of configuration options and exhaustive testing of such systems is infeasible. For instance a moderate system having 64 binary options have 2^{64} possible combinations to test which is clearly impractical. Even if there are resources available to test the system exhaustively, it is inefficient because only a small proportion of the option-value combinations trigger the failure [39]. CIT is a systematic way which provide a practical way to have acceptable trade off between cost and efficiency while triggering failure combinations. The CIT employs special combinatorial object termed "Covering Arrays (CAs)" to systematically cover certain key option setting combinations for testing purposes. [28]

2.2. Tradtional Covering Arrays (CAs)

A t-way covering array is defined as a set of configurations for a given input space, in which each possible combination of t options appear at least once. The parameter *t* is referred as *coverage strength*. [39] The Table 1.2 demonstrates a 2-way covering array for five binary configuration options. The configuration space model consists of 5 binary options, with no interoption constraints. Exhaustive testing of such model requires 2^5 configurations, but 2-way CA for this configuration space model only comprise on 9 configurations to test, which is way lower than size of configurations space model. For a fixed t, as the value of number of configuration options increase the overall size of the covering array is increased in smaller proportion, in constrast to the size of whole configuration space. Thus, very large configuration spaces can be covered efficiently. Typically higher the coverage strength t, higher the interaction fault revealing ability of the covering array. A study suggests that 70-88% of such faults can be revealed using t strengths 2 and 3 while 99% of such faults can be revealed employing t=6 [21]. For a given size of a configuration space, increase in t can escalate the size of CA by a significant factor. Typically t=2,3 are commonly used [39].

2.3. Virtual Option

A virtual option(Vopt) is described as the outer most *decision* statement in a hierarchy of an if-else configuration blocks. A virtual option has a set of settings under a coverage criterion. A virtual option can take any of the possible settings to exercise various control flow paths in its structure. Table 2.1 presents an example of virtual options and its settings under decision coverage (DC) for a hierarchy of if-else structures. The coverage criterion are discussed in subsequent sections.

<table>
<tr><th>Vopt id.</th><th>Nested If-blocks</th><th>Vopt</th><th>Settings (DC)</th></tr>
<tr><td>1</td><td>if(o1||o2){</td><td>(o1||o2)</td><td>{(o1||o2),!(o1||o2),(o1&&o2&&o3),(o1&&o2&&!o3)}</td></tr>
<tr><td></td><td>if(o3){</td><td colspan="2" rowspan="2"></td></tr>
<tr><td></td><td>}}</td></tr>
<tr><td>2</td><td>if(o5||o2){</td><td>(o5||o2)</td><td>{(o5||o2),!(o5||o2)}</td></tr>
<tr><td></td><td>}</td><td colspan="2"></td></tr>
</table>

Table 2.1: Illustration of a virtual option and associated settings

2.4. Cyclometic Complexity

Cyclometic complexity (cyclo) is a graph-theoretic complexity metric and used to manange and control program complexity. The cyclo complexity depends only on the decision structure of the program and irrelevant of its physical size. The cyclometic complexity is defined as a number of a control flow graph G with n vertices, e edges and p connections are $cyclo=v(G)=e-n+p$

In a strongly connected graph, the cyclo is equal to maximum number of linearly independent paths [25].

Higher the cyclometic complexity, higher probability of errors and thus greater the testing effort needed [25]. The cyclo levels between 2 and 4 are considered low, while 5-7 are considered moderate and 7+ are considered high [18].

For a tangled if-then else hierarchy, the cyclometic complexity of associated virtual option will be higher. Tangled virtual options can be located through cyclo values. For example in Table 2.1, Vopt1, Vopt2 has corresponding cyclo values of 3 and 2. So, Vopt1 is more tangled than Vopt2.

2.5. Testing Approaches

At a high level, there are three different approaches for sofware testing termed: black-box, white-box and gray-box testing [3].

Blackbox testing targets the software's external behaviour and attributes from an end-user's point of view. In contrast whitebox-testing often referred as glass-box testing is based on the internal structure of software such as the architecture of source code, control flow and internal data structures and algorithms. Informally white-box testing is often described as testing from a developer's point of view. Both white-box and black-box testing complements each other for a complete testing effort. White box testing is effective revealing granular low-level faults such as data-flow or boundary conditions whereas, black box methods are effective at revealing high-level faults such as system's usability faults.

Gray-box testing features the characteristics of both black and white box testing. The graybox approach focuses testing of components for their functionality and inter-operability in the context of system design. The gray-box testing consists of internal knowledge of software and the operating enviornment. In certain application domain such as Web applications the gray-box methods have proved to be quite effective. Gray-box testing is defined as *"The tests designed based on the knowledge of algorithms, internal states, architectures or high-level descriptions of program behaviour"* [24]

This work is termed as gray-box combinatorial interaction testing, due to the fact that the subject applications have been statically analyzed to figure out how configuration options interact with each other. Our proposed testing technique is guided by the coverage criterions and missing coverage are covered using uncovered settings of configuration option combinations.

2.6. Structural Coverage Criterion

Structural coverage criterion are broadly classified into two categories: control flow and data flow. The data flow criterion are based on measuring the flow of data between variable assignments and subsequent references aka def-use. The metrics measuring data-flow are based on analysis of paths from variable definition to its use.

The control flow criterion are based on measuring control-flow between block of statements. Typically control flow criterion are more common than data flow criterion. The extent of structural coverage acheived for control flow criterion is measured in terms of statements executed, exercising of control-constructs and associated logical expression evaluations. Some of the well known structural coverage criterions [17] are: Statement Coverage (SC), Decision Coverage (DC), Condition Coverage (CC), Condition and Decision Coverage (CDC), Modified Condition and Decision Coverage (MC/DC), Multiple Condition Coverage (MCC) etc. Each of the control flow criterion has different level of coverage detail, scope and strength.

2.6.0.1. Condition vs Decision

The discrimination between condition and decision is as follows: *A condition is defined as a boolean expression that doesn't have any logical operators such as and(&&), or(||), not(!). Whereas, a decision on the other hand has more than one conditions connected by logical operators.*

2.6.1. Statement Coverage(SC)

Statement Coverage is described as, all statements in the program must be invoked at least once during testing. 100% statement coverage implies the execution of all statements. The notion of SC is verification that all statements in a program are reachable. Among control flow criterion, SC is considered the weakest.

The criterion employed for our proposed work are described as follows.

2.6.2. Decision Coverage(DC)

Decision coverage is employed for testing of control constructs a.k.a decision statements that alter the control-flow of the program and it is fulfilled by requirement of two-testcases one for a *true* and one for *false* outcome. Each decision statement can be comprised on one or more than one conditions. For example Table 2.2 presents an example on decision coverage. The decision *o1&&o2||o3* comprised on the three conditions *o1,o2,o3* and two test cases are suffice to exercise the decision to *true* and *false*. However, the effect of conditions *o2, o3* is not tested, that is the testsuite can't distinguish between the decision *o1&& o2||o3* and decision *o3*.

Decision coverage can ensure complete testing of control constructs only for simple decisions. i.e. decisions comprising a single condition e.g. o3

<table>
<tr><th rowspan="2">Decision</th><td rowspan="2">o1&&o2||o3</td><th colspan="3">DC Testcases</th><th rowspan="2">Outcome</th></tr>
<tr><td>o1</td><td>o2</td><td>o3</td></tr>
<tr><th rowspan="2">Conditions</th><td rowspan="2">o1,o2,o3</td><td>T</td><td>T</td><td>F</td><td>T</td></tr>
<tr><td>F</td><td>T</td><td>F</td><td>F</td></tr>
</table>

Table 2.2: An example of Decision Coverage

2.6.3. Condition Coverage(CC)

Condition Coverage is also employed for testing control constructs with the purpose to exercise each condition in a decision. In CC, each condition is required to take all possible outcomes at least once. Note that this doesn't necessarily mean that the respective decision is fully exercised for all possible outcomes. For instance, Table 2.3 presents an example on CC, where each condition *o1,o2,o3* is being exercised to *true* and *false* however, the decision *o1&&o2||o3* has only been exercised for *true*. For that reason CC coverage doesn't subsume DC.

The Condition Decision Coverage (CDC) combines both CC and DC and requires that test cases should also exercise decision for all possible outcomes.

Decision	*o1&&o2‖o3*	**CC Testcases**			**Outcome**
		o1	*o2*	*o3*	
Conditions	*o1,o2,o3*	T	*F*	T	**T**
		F	*T*	*T*	**T**
		T	T	*F*	**T**

Table 2.3: An example on Condition Coverage

2.6.4. Modified Condition and Decision Coverage (MC/DC)

The MC/DC criterion primarily augments condition decision coverage (CDC) and has following requirements:

- Each decision in the program has taken all possible outcomes at least once
- Each condition in a decision in the program has taken all possible outcomes at least once
- Each condition in a decision has been shown to independently affect that decision's outcome

The independence effect is described that each condition when tested relative to other conditions should independently affect the outcome. Typically MC/DC testsuites require n+1 testcases for a decision comprising of n conditions. For full MC/DC coverage the testsuite must be carefully crafted based on n+1 testcases. Table 2.4 presents and example on MC/DC coverage on decision with 4 testcases, acheiving complete coverage. The test cases are exercising complete condition and decision coverage by exercising the decision and all conditions to both *true* and *false*. However, testcases *(1,3), (2,4), (1,2)* demonstrate the independence effect of conditions *o2,o3,o1* .In comparison to CC and DC, MC/DC significantly requires more testing effort and test cases. Generally, MC/DC is employed for testing of safetly critical softwares to comply stringent certification requirements [7].

Decision	*o1&&o2‖o3*	**MCDC Testcases**				**Outcome**
		Id	*o1*	*o2*	*o3*	
Conditions	*o1,o2,o3*	*1*	T	T	F	**T**
		2	*F*	*T*	*F*	**F**
		3	*T*	F	*F*	**F**
		4	F	T	T	**T**

Table 2.4: An example on MC/DC

2.7. Interaction Coverage Criterion

We have proposed the following two criterion, which are the extensions of the corresponding structural coverage criterion, to test the *interaction* of virtual options i.e. Decision Coverage (DC) and Condition Coverage (CC). However, we didn't proposed MC/DC for a valid reason which will be discussed in this section.

2.7.1. Decision Coverage(DC) for Interaction Coverage

Decision coverage for interaction testing is defined as, *each t-way interaction of virtual options should be exercised to its all possible outcomes.* Table 2.5 demonstrates DC for 2-way interaction of two virtual options and presents a full coverage interaction testsuite for testing. For complete t-way interaction coverage of participating virtual options in an interaction the testsuite should exercise the interaction to both *true* and *false*. However, it should be noted complete interaction DC doesn't guarantee full exercise of each virtual option in the interaction.

Vopt id.	**Vopt**	**Settings**	
1	(o1&&o2)	[(o1&&o2),!(o1&&o2)]	
2	(o3)	[o3,!o3]	
2-way interaction	*Vopt1 && Vopt2*		
Test suite			
o1	*o2*	*o3*	*outcome*
T	T	T	*T*
T	T	F	*F*

Table 2.5: Illustration of interaction DC for virtual options

2.7.2. Condition Coverage(CC) for Interaction Coverage

Condition coverage for interaction testing is defined as, *each virtual option in an interaction should be exercised to its all possible outcomes.* Table 2.6 demonstrates CC for 2-way interaction of two virtual options against a given test suite, which exercise each virtual option to its both possible outcomes *true* and *false* but doesn't exercise the 2-way interaction to *true* and *false*. Thus, interaction CC doesn't subsume interaction DC and complete interaction CC doesn't guarantee complete interaction DC.

Vopt id.	**Vopt**	**Settings**	
1	(o1&&o2)	[(o1&&o2),!(o1&&o2),o1,!o1,o2,!o2]	
2	(o3)	[o3,!o3]	
2-way interaction	*Vopt1 && Vopt2*		
Testsuite			
Vopt1	*Vopt2*	*Testcases*	*outcome*
F	***F***	[T,T,F]	***F***
T	***T***	[T,F,T]	***F***

Table 2.6: Illustration of interaction CC for virtual options

2.7.3. MC/DC for Interaction Coverage

MC/DC can't be defined for interaction coverage for virtual options because its illogical against the original definition of MC/DC and violates the interaction of virtual options for a subset of settings of individual virtual option. MC/DC for interaction testing would have the following definition in the context of interaction testing, which is not valid, i.e. *Each setting in a t-way virtual option interaction should be shown to independently affect the outcome of interaction*. For instance, the virtual option setting *!o3* violates the whole interaction of *Vopt1&&Vopt2* of Table 2.6

3

RELATED WORK

Combinatorial interaction testing (CIT) is way of testing huge configuration spaces where exhaustive testing are not an option. CIT is a black-box technique indicated by large body of literature. CIT is usually performed through employing t-way CAs. Different CA generation techniques, catering for different constrained option and configuration space models are discussed and variety of construction techniques have been proposed. This chapter describes some of the related work in different categories as follows:

There are a variety of t-way testing and CA generation approaches, which are mostly AI based and require complex computations. Thus, in effect are limited to small configuration spaces and interaction strengths [10,20,35,37,44]. Nie et al. [29] broadly classify CA generation techniques to four main categories, random search based methods [33], mathematical models [16], heuristic search based methods [8] and greedy methods [36]. Ahmed et al. [1] proposed a novel CA generation strategy based on particle swarm optimizations, which can cater for complex configuration models and high interaction strengths upto t>=6. Their approach supports uniform and variable strength CAs, however, it lacks to handle inter-option constraints and support of seeding. Our approach takes a different coarse, our CA configuration generation is based on constraints satisfaction. Under a given test criterion we've a pool of unsatisfied constraints that represent the missing coverages. We create sub pools of constraints in such a way each pool contains those constraints which can be satisfied together. For each pool of constraing we've generated a missing test configuration.

Optimal test suites have always been desired especially, which can provide complete test coverage. In literature, there have been works [31, 32, 34, 38]. addressing minimization strategies of test suites through program analysis. Arlt et al. [2] work targets GUI testing based on event sequence testing using sequence covering arrays. Their approach is based on static analysis of application's code to figure out and eliminate redundant event-sequences or invalid sequences of events. Their technique discovers the causality among event sequences, which is used to eliminate redundant and invalid test cases during test suite generation. However, our approach differs in the sense that instead eliminating the redundant test configurations we generate only essential configurations which provide complete test coverage. Thus, our test suites are comprised on all essential test cases.

A number of works [15,27,42] are based on configuration space exploration of program to guide test suite generation, based on domain partioning to meet given objectives of testing activity. Yu et al. [41] proposed a novel combinatorial interaction test generation algorithm based on IPOG-C, which performed better in terms of test case generation, time and size of test suite. The test generation employs a novel constraint handling strategy termed minimum invalid tuples, in contrast to existing constraint solving techniques. The test generation process generate only such test cases that are validated by the specified valid tuples. The valid tuples are derived from feature model of subject application. In contrast, we take the coarse of filtering of invalid t-tuples of option settings and employ only valid tuples to guide the test generation process. We've partioned t-tuples of option settings into pools and each pool is responsible for generating a valid configuration.

Barret et al. [4] proposed a combinatorial testsuite generation tool based IPOG algorithm aimed to ensure specified degree of configuration space coverage. Their proposed approach is gray-box approach where they integrate the application specific knowledge, in the form of constraints that guide test suite generation process. The test suite generation process is customized according to application's requirement by partial or full inclusion of seeds through customized combinations, giving the ability to enforce certain test cases either included or excluded through *nesting factors*. The concept of nesting factors enabled to exercise the required degree of *path coverage* in hierarchical structures, generating inter-option constraints or specifying invalid combination of option values. In contrast, our approach which is also gray-box performs static analysis of code, obtains configuration space model, inter-option constraints and the structural hierarchy of configuration options. We exercise different paths of hierarchical structures based on the constrained settings obtained under given criteria. Barret's approach only cater for numerical or catergorical values of options and exercise limited degree of hierarchical structures, in the sense of restraining certain option values that violate the invalid exercise paths. In contrast, our approach aimed to target both structural and interaction coverage which is guided by test criterion. The test criterion defines the scope of testing which is represented by set of option settings under a given criterion. The set of option settings determines the scope of testing, more detailed testing have more settings. Thus we can adjust the resolution of testing from low to high as specified by test criterion. Constraint solving techinques to enforce scope of particular test configurations have been studied in [5, 9, 11, 14] .

Our test subjects were compile time configurable where the configuration mechanisms was implemented through c-preprocessor(cpp) macros. We've parsed those macros to establish configuration space model of the subject application, option interactions and inter-option constraints. In literature, there have been many related works analyzing different aspects of cpp usage and contributing various techniques. Cpp usage patterns for codebases were studied for various real world applications in [12, 23, 30]. F. Medeiros et al. [26] studied the variability mechanisms by cpp, they empircially studied the fault-proneness and fault caused by this variability implementing mechanism. Lei et al. [22] proposed a generalization to the IPO test generation strategy from pairwise (2-way) to in general t-way testing. This work reports on the design choices in terms of horizontal and vertical growth and optimizations they've used for their approach to avoid the challenge of combinatorial grownth of coverage space while emphasizing acceptable testsuite generation time. In this regard, [6, 19] emphasize a mixed strength based coverage approaches for pseudo-exhaustive coverage for critical applications. Combinatorial explosion can be a significant problem during tackling large configuration spaces, for instance lookup time and memory management can be major issues. Our interaction test suite generation approach addressed these issues by maintaining a hierarchy of dictionary based lookup tuple caches at various levels, while memory managment has been efficiently implemented, memory cleanups have been run on critical points during computations.

Yu et al. [43] proposed a comparison of traditional coverage criterion and proposed MUMCUT criterion, an extention of MC/DC. They compared the studied criterion emperically and formally to establish the fact in 1-way coverage MC/DC suites are effective but they can miss some faults, in a given suite, which can certainly be detected by MUMCUT, based on critical testpoints of the logical expressions. [13] empirically report the granularity of coverage criteria is always effective and efficient in revealing more faults but has its own costs. Fault-based logic coverage is comparable to MC/DC in effectiveness. However, in literature as per our knowledge interaction test criterion are not discussed for interaction testing. We've addressed this issue and proposed interaction coverage criterion which are in fact extentions of corresponding structural coverage criterion as discussed in chapter 2 in detail, in our belief the proposed interaction coverage criterion will effectively reveal interaction faults.

4

APPROACH

In the last chapters we have provided some of the preliminary background of the approach. In this section we put those notions into practice.

Given a configuration space model and a coverage criterion CC, DC or MC/DC we figure out what needs to be covered and under which conditions they need to be covered. Then, we specify everything to be covered as a constraint. Finally, we aim to cover everything using a minimal number of configurations.

The source code of subject application is statically analyzed to figure out configuration options and their interactions. Since, the subjects we've analyzed were all C/C++ based applications where the configurations options were embedded in the source code through c-preprocessor macros. Extraction of configuration options and their interactions have been performed through parsing of the c-preprocessor(cpp) code. The static analysis phase is comprised through the following steps.

The source code of the subject application is processed through a source code preprocessing phase which involves formatting the source code to a standard and subsequently to a parsing phase which extracts all the actual binary configuration options and virtual options and its various settings under a criteria.

The various options of a virtual options in parsed source code are transformed to corresponding guard expressions. A guard expressions is decribed as *2-tuple {guard, expression}* where expression can be only processed when the guard is set to *True*. The notion behind the introducing the guard expressions is capturing the hierarchy of the tangled virtual options. For instance, the Table 4.1 presents an example of source code listing and corresponding guard expressions, virtual options(Vopt) and actual configuration options(aOpt).

V opt.	**Nested If-blocks**	**Guard Expressions**
1	if(o1‖o2){	*{True,o1‖o2}*
	if(o2){	*{o1‖o2,o2}*
	}else{	
	if(o3&&o4){	*{!o2,o3&&o4}*
	}else{	
	if(o5‖o2){	*{!(o3&&o4),(o5‖o2)}*
	}}}}	
2	if(o6){}	{True,o6}
aOpt	*o1,o2,o3,o4,o5,o6*	

Table 4.1: Virtual Options, IfBlocks and Guard Expressions

G_{sut}
{{{True,o1‖o2},{o1‖o2,o2},{!o2,o3&&o4},{!(o3&&o4),(o5‖o2)}},{{True,o6}}}

Table 4.2: Guard Expressions of Table 4.1

Given a set of guard expressions G_{sut} the goal is to perform interaction testing of virtual options. The interaction testing is performed under a given criterion. The criterion used in our approach are CC, DC and MC/DC. The goal is to acheive full coverage under a given criterion. To this end, it is figured out what combinations need to be tested and under which conditions they must be tested. The t-way interaction testing means, testing the t-way interaction of virtual options. Thus, 1-way testing means testing the structure of a given virtual option under a criterion. Whereas, 2-way and 3-way interaction testing of virtual option means performing 2-way or 3-way interactions of virtual options. The interaction testing is performed under a given strength of t either 2 or 3, which is infact testing against a corresponding 2-way or 3-way CA configurations.

For 1-way interaction testing of virtual options i.e. structural testing of virtual options, been performed under CC, DC and MC/DC. Whereas, for 2-way interaction testing of virtual options has been only performed under CC and DC but not MC/DC for the reason discussed in chapter 2.

The reason for choosing DC, CC and MC/DC as testing criterion in our approach are due to the fact each subsequent criterion perform involved degree of detailed coverage. For instance, DC can only exercise the outcomes of virtual option to both *true* and *false*, where as it doesn't exercise each actual option in a virtual option. CC coverage does the job of exercising each actual option in a virtual option but doesn't necessarily exercise the virtual option to both possible outcomes. MC/DC address both of these issues but require much more testing effort and larger number of testcases. Safety critical applications that require stringent testing requirement for certification requirements usually rely on MC/DC coverage.

4.1. 1-way Testing of Virtual Options

The subject application is analyzed for 1-way coverage of virtual options for each of the used criterion (DC,CC,MC/DC) against 2-way and 3-way covering arrays (CAs). The set of guard expressions of the subject application G_{sut} are analyzed for coverage provision and subjected to the Algorithm-1 for coverage measurment. The Algorithm-1 takes a set of guard expressions G_{sut} and a t-way CA as seed and test criteria. The Algorithm-1 proceed along the following lines, the set of guard expressions are converted to corresponding *Regular Constraints* if the criterion is CC or DC otherwise, for MCDC the guard expressions are converted to *Observability Constraints*. The notion behind this conversion is the taking into account the level of details to be covered for each guard expression according to the criterion requirements.

The notion of Regular and Observability Constraint are boolean satisfiability expressions, that are considered satisfied iff any configuration in seed satisfies them. However, if not, during the stage of generating tests for missing coverage, their SAT solution is generated to meet complete coverage requirement. Each of the virtual option is treated as a constraint to satisfy during interaction testing but the type of constraint is determined by test criteria. For CC and DC the constraints are treated as Regular Constraints and as Observability Constraints for MC/DC.

The Algorithm1 performs coverage measurement using a t-way CA as test suite Lines[7:27].

Table4.3 presents an example on regular and observability constraint for the guard expression *{True, o1 && o2}*. Lines[28:39] perform the actual boolean satisfiability testing on the constraints(regular/observability). The satisfiablity testing of regular constraints comparatively require less computation than observability constraints. For instance, a single regular constraint under DC coverage requires *2* satisfiablity tests, CC requires *2+2*noOfConditions* and MCDC requires *2+2*noOfConditions+2*noOfConditionsObservability* tests.

The algorithm returns the set of unsatisfied constraints S_{un} for the criteria and the measured percentage coverage P_c.

Guard Expression	{True, o1&&o2}	**Constraint Representation**
Constraint	**Criteria**	
Regular	*DC*	{(o1&&o2),!(o1&&o2)}
	CC	{(Regular DC Constraint),o1,!o1,o2,!o2}
Observability	*MCDC*	{(Regular CC Constraint),[o1&&o2,o1],[o1&&o2,o2]}

Table 4.3: Representation of Regular and Observability Constraint for a Guard Expression

Algorithm 1 Algorithm to perform coverage measurement on virtual options

Input: G_{sut} set of guarded expression of sut, CA_t t-way CA, *crit* coverage criteria
Output: P_c percent coverage, S_{un} set of unsatisfied constraints

```
if crit=="DC" or crit=="CC" then
    cons ← convertSetofguardedExprsintoRegularConstrs(G_sut)
else if crit=="MCDC" then
    cons ← convertSetofguardedExprsintoObsConstrs(G_sut)
end if
cvgInfo, S_un ← measureCvg(crit, cons, CA_t)
procedure measureCvg(crit,cons,testsuite)
    satisfied ← {}
    unsatisfied ← {}
    for all c in cons do
        for all testcase in testsuite do
            if isRegCons(c) then
                if isRegConsSatisfied(c,testcase) then
                    satisfied ← satisfied ∪ c
                    break
                end if
            else if isObsCons(c) then
                if isObsConsSatisfied(c,testcase) then
                    satisfied ← satisfied ∪ c
                    break
                end if
            end if
            unsatisfied ← unsatisfied ∪ c
        end for
    end for
    return {satisfied, unsatisfied}
end procedure
procedure isRegConsSatisfied(c,testcase)
    isSatisfied ← false
    constraint ← c
    isSatisfied ← isBooleanSatisfiable(constraint, conditions, testcase)
    return isSatisfied
end procedure
procedure isObsConsSatisfied(c,testcase)
    isSatisfied ← false
    constraint, obsVar ← c
    isSatisfied ← isBooleanSatisfiable(constraint, obsVar, conditions, testcase)
    return isSatisfied
end procedure
```

4.2. Generation of Missing Testcases for Complete 1-way Virtual Option Testing

The missing configurations in a t-way CA, for complete virtual option testing is generated through Algorithm-2. The missing coverage information is obtained from Algorithm-1 in the form of set of unsatisfied constraints S_{un}. Algorithm-2 takes S_{un} and desired criteria and generate additional configurations a.k.a test cases. These additional test cases in conjuction with t-way CA comprise full coverage test suite termed CCA (Complemented Covering Array).

This algorithm uses a greedy approach to generate a minimal number of additional test cases for the unsatisfied constraints S_{un}. Depends on the type of unsatisfied constraints the algorithm adjusts itself for either 1 or 2 steps. For regular constraints the algorithm uses 1 step and vice versa. The test generation process is greedy where the heuristic is to mutually group and satisfy the maximum possible number of constraints together in a single boolean satisfiablity instance.

A boolean satisfiablity solution is generated for each group of mutually satisfiable regular constraints. Similarly, for observability constraints the testsuite is first partially constructed for satisfying the observability constraints and then later for regular constraints.

4.3. 2-way and 3-way Virtual Option Interaction Coverage

At high level there is no major difference between the way we compute missing configurations of t-way CA for 1-way virtual option testing and 2- and 3-way virtual option interaction coverage. We take the configuration space model of subject application as a set of constraints. In this context, a constraint is satisfied if there is at least one configuration in the generated test suite that satifies the constraint. We've used a greedy algorithm to compute the full coverage interaction test suite termed FIT. The algorithm maintains a set of clusters. Initially, the set is empty. Then we iterate over all the constraints. For each constraint we try to find a cluster in the set where we can insert the constraint. If no

Algorithm 2 Algorithm for additional 1-way VO Coverage

Input: S_{un} set of unsatisfied constraints, *crit* coverage criteria
Output: T_{add} set of additional testcases, *CCA* complemented CA

```
T_add ← grdyObtFullCvgCons(S_un, {})
CCA ← T_add ∪ CA_t
procedure grdyObtFullCvgCons(cons,seed)
    obsCons ← {}
    regCons ← {}
    for all c in cons do
        if type(c)=="observability" then
            obsCons ← obsCons ∪ con
        else if type(c)=="regular" then
            regCons ← regCons ∪ con
        end if
    end for
    testsetObsCons ← {}
    testsetObsCons ← grdyObtTsForObsCons(obsCons)
    partialTs ← {}
    partialTs ← grdyFindSatisSubsForRegExprs(regCons, vars, testsetObsCons)
    return partialTs
end procedure
procedure grdyFindSatisSubsForRegExprs(cons,vars,seed)
    PartialTs ← {}
    for all satisSubset in satisSubsets do
        testcases ← genTestcase(satisSubset, getVars(satisSubset))
        partialTs ← partialTs ∪ testcases
    end for
    return partialTs
end procedure
procedure grdyObtTsForObsCons(cons,vars,seed)
    PartialTs ← {}
    for all satisSubset in satisSubsets do
        testcases ← genTestcaseForObs(satisSubset, getVars(satisSubset))
        partialTs ← partialTs ∪ testcases
    end for
    return partialTs
end procedure
procedure genTestcase(cons,vars)
    if cons!=false then
        return booleanSatisfiable(cons, vars)
    end if
    return { }
end procedure
```

such cluster is found, we create a new cluster. Each cluster represents a set of constraints that are solvable together. At the end each cluster is used to generate a configuration. All such configurations constitute the FIT test suite. Algorithm-3 performs this whole operation.

The Algorithm-3 uses the configuration model R_M of the subject application, which is comprised on a set of virtual options and its settings for a the given criteria.

The FIT test suite can be computed incrementally, if a seed of existing configurations is provided. For tj-way of FIT suite computation ti-way of unique combination of option settings are determined that are invalid, i.e. the combinations which result in constraints collision leading to no boolean satisfiablity solution.

Since during t-way FIT suite generation, the number of t-tuples grow factorially especially, for 3-way case they reach to millions in count. For the determination of valid t-tuples of virtual option settings, some optimizations have been used by maintaining some tuple caches. Those optimizations have proved to be quite effective and saved the lookup time by orders of magnitude. The first step is, maintaining a sorted 2-way cache of valid and invalid t-tuples as dictionaries, where each t-tuple is a key, that way we've acheived a constant lookup time for a given t-tuple to determine its validity. Those dictionaries are maintained along the way during computation. Secondly, a separate dictionary of *ti*-way tuples is maintained and used for higher $tj > ti$. To determine a tj tuple is valid or invalid the lookup is performed in the *ti* caches. For cache hits, no need of additional tuple validations is required which results a significant performance improvement.

Algorithm 3 Algorithm to compute FIT suite

Input: R_M real model of sut, t coverage strength, *incr* construct incrementally, *seed* optional seed

Output: *cores* set of cores, *testsuite* cit testsuite

```
vModel ← getVirtualConfigSpaceModel(R_M)
Ucores ← findAlltWayUnsatisfiableCores(vModel, 2)
if incr then
    for all t_i in range(3,t) do
        Ucores ← findAlltWayUnsatisfiableCores(Ucores, t_i)
        testsuite, seed ← computeCITTestsuite(vModel, t, seed)
    end for
end if
procedure computeCitTestsuite(vModel, t, seed)
    tWayComb ← findAlltWayCombinationOfSettingsInSeed(seed, t)
    satisCores ← findSatisfiableCores(vModel, t)
    testsuite ← generateTestsuite(satisCores)
    return satisCores,testsuite
end procedure
procedure findSatisfiableCores(vModel, t)
    optCombinations ← tWaySubsets(V_i ∈ {v_0, ..., v_n}, t)
    for all optCombination in optCombinations do
        allTtuples ← allTtuples ∪ crossProduct(optCombination)
    end for
    allTtuples ← removeAllInvalidTtuples(allTtuples)
    tuplesCoveredInSeed, unCoveredTuplesInSeed ←
checkTTuplesCoveredInSeed(seed, allTtuples)
    allconstraints ← maptTuplesToActualSettings(allTtuples)
    cores ← {}
    for all r in allConstraints do
        clusterFound ← false
        for all c in cores do
            if isConstraintPlaceableInCore(c,r) then
                c ← c ∪ r
                clusterFound ← True
            end if
            if !clusterFound then
                newCore ← createEmptyCore()
                newCore ← newCore ∪ r
                cores ← cores ∪ newCore
            end if
        end for
    end for
    return cores
end procedure
procedure generateTestsuite(satisatisfiableCores)
    testsuite ← {}
    for all core in satisfiableCores do
        testsuite ← satisfiabilitysolve(core)
    end for
    return testsuite
end procedure
```

5

EXPERIMENTS

This chapter provides information about the experimental setup, design and performance for the carried out work.

5.1. Test subjects

For the proposed work we have chosen a set of subject applications(suts) for experimentation. All of the subject applications possess a varying degree of configurability ranging from intermediate to high. The test subjects open source applications configurable through c-preprocessor macros. The test subjects ranged from all application domains, ranging from webserver, text and graphical editors to virtual machines and security applications. Table A.2 provides a brief summary on the profile on each of them.

The test subject (sut) is an independent variable to study the effects of varying configurability on actual software systems. Each of the test subjects (suts) possess a fix number of actual configuration options distributed in various if-then-else constructs to implement variability. The group of the subjects has a ranging degree of low to high configurabilty.

5.2. Experimental Setup

The real configuration model of subject is comprised on the actual structure and hierarchy of virtual options and corresponding settings. The experimental setup takes the physical configuration model of the subject application based on the static analysis of source code, a coverage criteria and a t-way covering array of strengths 2,3. For each phase of testing the experimental setup performs a coverage provision measurement by the t-way CA and reports the coverage statistics and optionally additional test cases for full coverage under that criteria.

5.3. Experimental Model

The experimental setup was broadly based around two major phases. In the first segment of phase I, the goal was structural testing i.e. investigation of 1-way coverage provision for virtual options by t=2 and t=3 way CA under the 3 structural coverage criterion (DC,CC and MC/DC). The lack of full coverage of CAs are covered through the generation of additional test cases to complement into 100% test suites termed *CCA (complemented covering arrays)*.

Secondly, the second segment of each phase of experiments was dedicated to investigation of t=2,3 way interaction testing of virtual options (VO) against the complemented covering arrays (CCA) and measuring their coverage provision for the interaction decision(DC) and condition coverage(CC). In addition the main objective of this segment was to generate the full interaction coverage test suites (FIT) for complete t-way interaction testing of virtual options.

In the second phase of experiments all of the subject applications were broken down to 5 different cyclometic complexity levels to investigate the effects of tangling on the coverage provision and expose the sufferings of covering arrays in correlation to cyclometic complexity.

In both phases we run the following number of experiments:

Phase1:

Segment 1:

{17 Suts} x {t=2,3} x {3 criterion (CC,DC,MC/DC)} x {Avg. 3 different t-way CA versions} x {3 runs of generating missing configurations} = 918

Segment 2:

{12 Suts} x {t=2,3} x {2 criterion (CC,DC)} x {2 coverage measurements}=96

Phase2:

Segment 1:

{17 Suts} x {t=2,3} x {3 criterion (CC,DC,MC/DC)} x {3 runs of missing test cases generation} x {5 Cyclo levels} = 1530

Segment 2:

{17 Suts} x {t=2,3} x {2 criterion (CC,DC)} x {2 coverage measurements} x {5 Cyclo levels} =680

The experiments were performed on a shared server machine with the following specs RAM: 126GiB Processor : Intel(R) Xeon(R) CPU E5-2690 2.90Ghz Cores:18 Disk Space: 1TiB OS: CentOs 6.4 Kernel: 2.6.32 GNOME: 2.28.2

5.4. Independent Variables:

Independent Variables:

Coverage Strength (t):

The coverage strength t is an independent variable in our experiments and the used values are 2,3. According to an earlier study [21] most of the faults are revealed by interaction among small set of options and they are effectively revealed by t-way interaction testing, the higher the t the more and more faults are revealed. The strength 2 and 3 reveal more than 80% of the faults and strengths upto 6 can reveal 99% interaction faults. The downside of increasing t is exponential increase in the computation and construction time with linear increase in a given number of options, due to a factorially increasing number of combinations to deal with. Therefore, for our experimental study we used percentage and 3-way levels of strengths to observe the effects of varying strength.

Testing Criterion:

In order to perform testing of the option interaction, and to guide the testing activity three different criterion were used. Each subsequent criterion possess a more detailed nature than the former one. Depend upon the time and resource constraints one can choose the criterion to define the scope of testing. For our 1-way virtual options testing phase we have used the following three criterion in order of increasing complexity Decision Coverage (DC), Condition Coverage (CC) and Modified Condition and Decision Coverage (MC/DC) and for the t-Way interaction testing of options we have used DC and CC.

Configuration Space Model (rModel):

The subject application has been statically analyzed to get configuration space model which comprised on configuration options, their settings, actual options and inter-option constraints as well as the interaction patterns.The rModel of the configuration of subjects was used in all the experiments and each sut possess a characteristic rModel.

Cyclometic Complexity (cyclo):

One of the important independent variable is cyclometic complexity of the if-then-else constructs. In order to study the effects of tangling on coverage provision and necessary amounts of test cases the cyclo variable is introduced. The chosen levels of cyclometic complexity are 2,3,4,5 and equal or greater than 6. The complexity of an if-then-else block is defined in terms of cyclo. The higher cyclo the higher complexity and the more testing effort needed.

Coverage Percentage:

The level of coverage provision is captured in terms of coverage percentage under a given criterion and given test suite. 100% coverage means full coverage under and vice versa.

Size of Test suite/No of Testcases:

The test suite size is the actual number of configurations comprising the test suite for a given testing activity. Full coverage test suites under a given criterion guarantees to provide 100% coverage. A test suite having minimum number of configurations(test cases) that can provide full coverage is better and desired, than one having more configurations for same coverage.

Total Construction Time:

The total construction time of a test suite is comprised on initialization of test suite generator and construction time. The initialization time is time taken for the test generator to set various data structures and perform memory allocations. The initialization cost is negligible to construction cost. The time is collectively reported in seconds. The smaller initialization time the better. The actual construction time of the test suite is the time to generate the test suite for a given set of unsatisfied constraints under a given criterion.

Low Complexity Region (LcR) and High Complexity Region (HcR): Low Complexity Region (LcR) is described as the region lying between cyclometic complexity 2 and 4, in terms of number of configuration options and the IfBlocks. Whereas, the region lying between cyclometic complexity 5 and onwards termed as HcR. If there is a high proportion of actual/virtual options in HcR the CA suffers more to provide structural and especially interaction coverage.

5.5.1. Data and Analysis:

The results of the performed experiments comprise mainly on set of coverage measurements under different criterion, set of full or additional test cases for full coverage under a given coverage strength and a set of test suite generation times. Moreover, the same stats were gathered under different cyclometic complexity levels of the subjects to study the effect of complexity on coverage and test cases required for each subject.

The experiments data can be found in appendices.

5.5.2. Study 1: An overview on the profile of subject applications

In this study the profile of the subjects were studied and presented in terms of percentage of actual number of configuration options and associated if-then-else blocks distributed across different levels of cyclometic complexity.

Note: In the box plots the delta shaped markers represent the mean values and the bars inside rectangles represent median values.

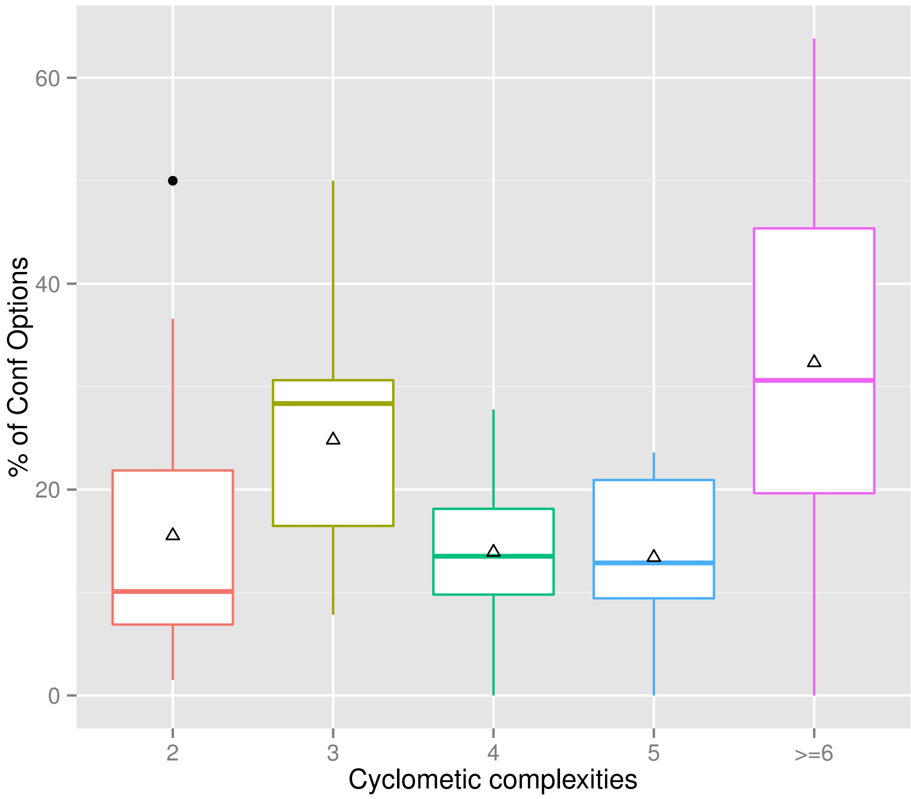

Figure 5.1: Percent distribution of configuration options of suts across different cyclo levels

	Cyclo				
	2	3	4	5	>=6
COpt %	15.5	24.8	13.9	13.4	32.3

Table 5.1: Mean % configuration options across cyclos

Figure 5.1, presents the comparison of the distribution of percentage of configuration options for the all the subjects across different cyclometic complexities levels. The mean percentage proportions are presented in Table 5.1. On average the mean proportion is almost equally divided into LcR (cyclo: 2,3,4) and HcR (cyclo:5,6+) i.e. 54.2% vs 46.8% with varying degree of variance across different cyclometic levels. Cyclo 2 and 3 represents the large variance in comparison to Cyclo 4 in LcR, while the HcR exhibits largest variance across Cyclo6 in a normal distribution. This implies a given t-way CA can provide better coverage in LcR in constrast to HcR. The sufferings of CA will be more prevalent in HcR.

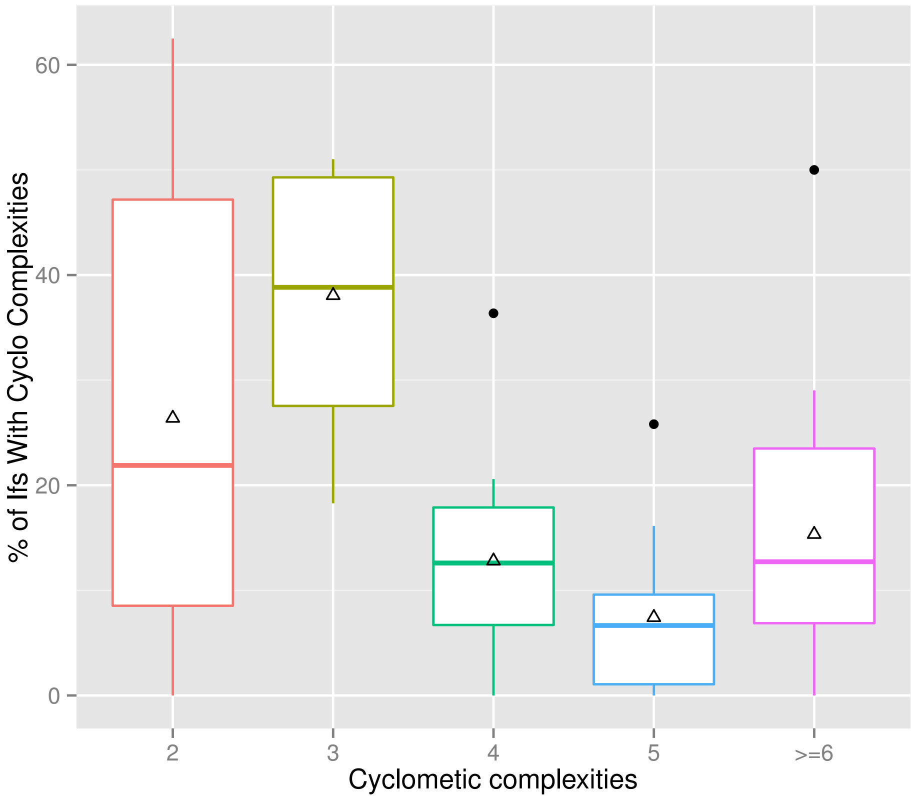

Figure 5.2: Percentage distribution of Ifs across different cyclo levels

	Cyclo				
	2	3	4	5	>=6
Ifs %	26.4	38.1	12.8	7.4	15.3

Table 5.2: Mean % Ifs across cyclos

Figure 5.2 illustrates the percent distribution of the Ifs (virtual options) across different cyclometic complexities. The means of the boxplots are shown in Table 5.2. The distribution shows that on average 73.6% of the distribution lies in the LcR with more comparative variance than HcR as whole. So in general more coverage can be achieved with fewer test cases.

Based on the distribution of Figure 5.1 and Figure 5.2, large proportion of configuration options and Ifs lie in LcR so its expected to get better coverage results in those regions and CAs are expected to suffer much lesser than regions of higher complexities.

5.5.3. Study 2: Traditional 1-Way Coverage of Virtual Option Testing

Coverage: In this study we've performed a set of experiments over set of subject applications in Table A.2 to determine the effectiveness of coverage provision of strength 2 and 3 way covering arrays for the three coverage criterion. We have generated additional test cases to complement those covering arrays for full 1-way VO testing. We measure to which extent covering arrays suffer.

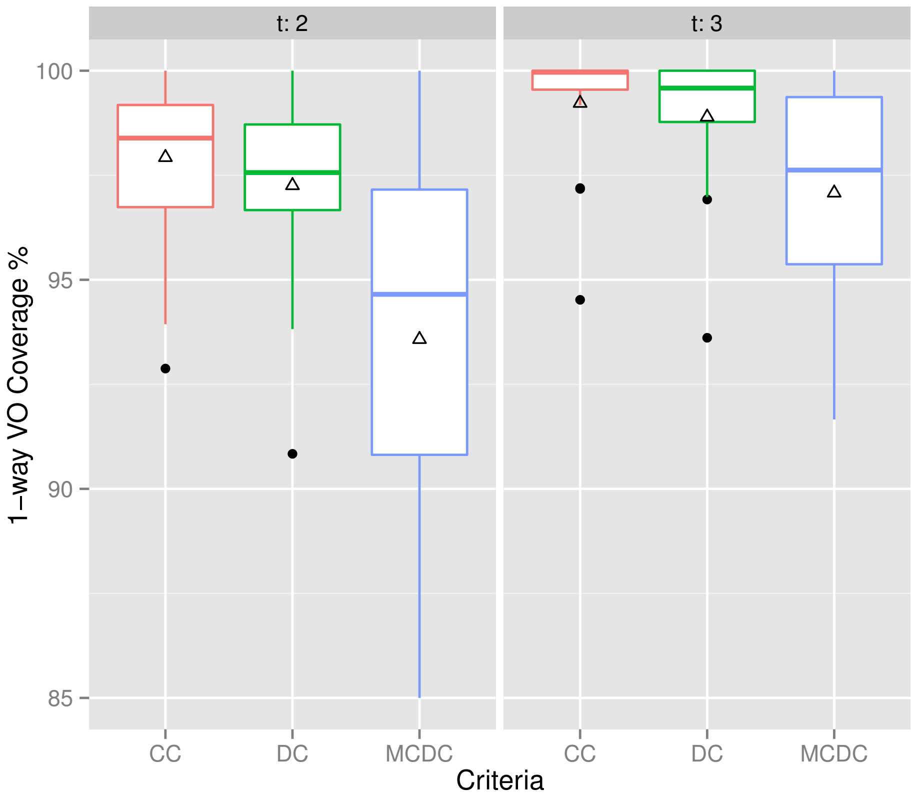

Figure 5.3: Comparison of coverage across different criterion for coverage strength 2,3

t	CC	DC	MCDC
2	97.9	97.3	93.6
3	99.2	98.9	97.1

Table 5.3: Mean % coverage in Fig.5.3

Figure 5.3 illustrates the coverage provided by t-way covering arrays for the three criterion for 1-way VO testing. The mean values of coverage are shown in Table 5.5.3. For 2-way, the CAs are suffering more in overall coverage provision than 3-way. Among the criterion CAs are suffering most for MC/DC with a high degree of coverage variance, which is due to the more complex nature of the criteria demanding more configurations, which CAs lack. The coverages of CC and DC are comparable where DC has received slightly lower coverage than CC, which is due to the fact that CC do not subsume DC and typically test suites are better at exercising individual conditions which result in better overall CC. For instance for the following expression *(a&&b)* and a hypothetical test suite *[(F,F),(F,T),(T,F)]* will exercise CC to 75% in comparison with DC to 50%. As t increases traditional CA suffers less but in practice t is typically small.

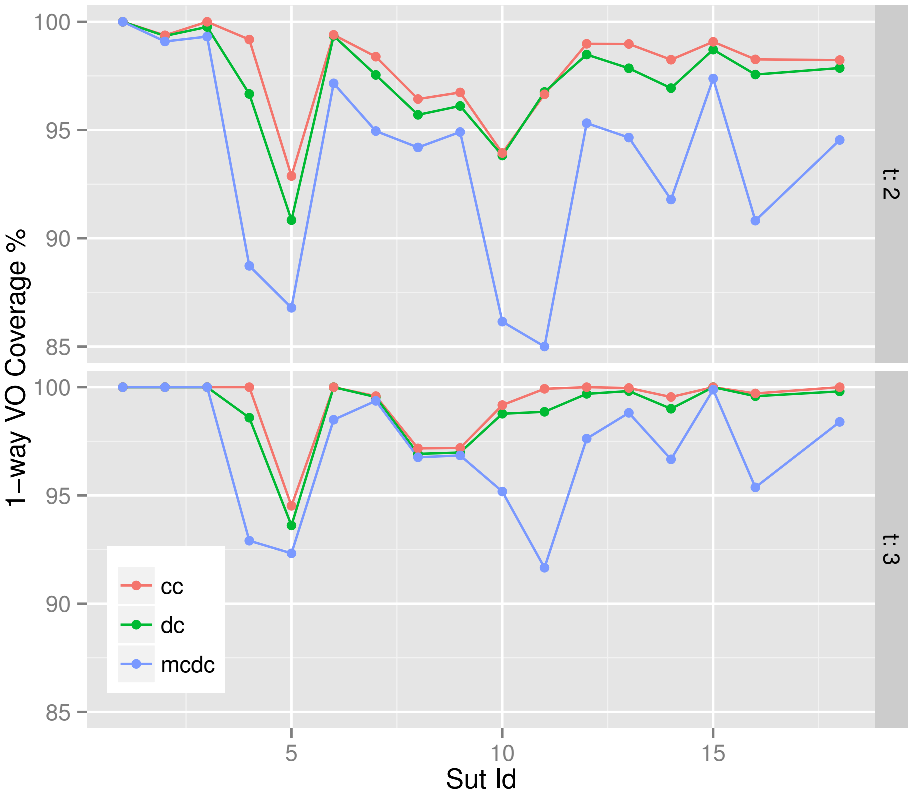

Figure 5.4: View of % coverage across all the suts for different criterion and different strengths

Figure 5.4 illustrates the different values of percent coverage for all criterion against 2-way and 3-way CAs for each of 17 subjects(suts). Overall, the coverage provided for 3-way is better than 2-way and generally, DC and CC are approximately equal for 3-way. Among criterion CC is performing slightly better than DC, evident by mean percentage coverage in Table 5.5.3.The MC/DC criterion in both halves in the figure is showing significant variations and received the least coverage. In both halves of the figure overall the structural coverage trend is similar, such that 3-way CAs are suffering less than 2-way. The sut with ids 5, 11 and 16 are showing significant drops in coverage in comparison to their neighbours, which is attributed to higher proportion of configuration options and Ifs in the HcR.

Although, as number of configuration options increase, different suts receive different degree of coverage rather a proportional decrease in coverage which lead us to investigate the suts in terms of cyclometic complexities.

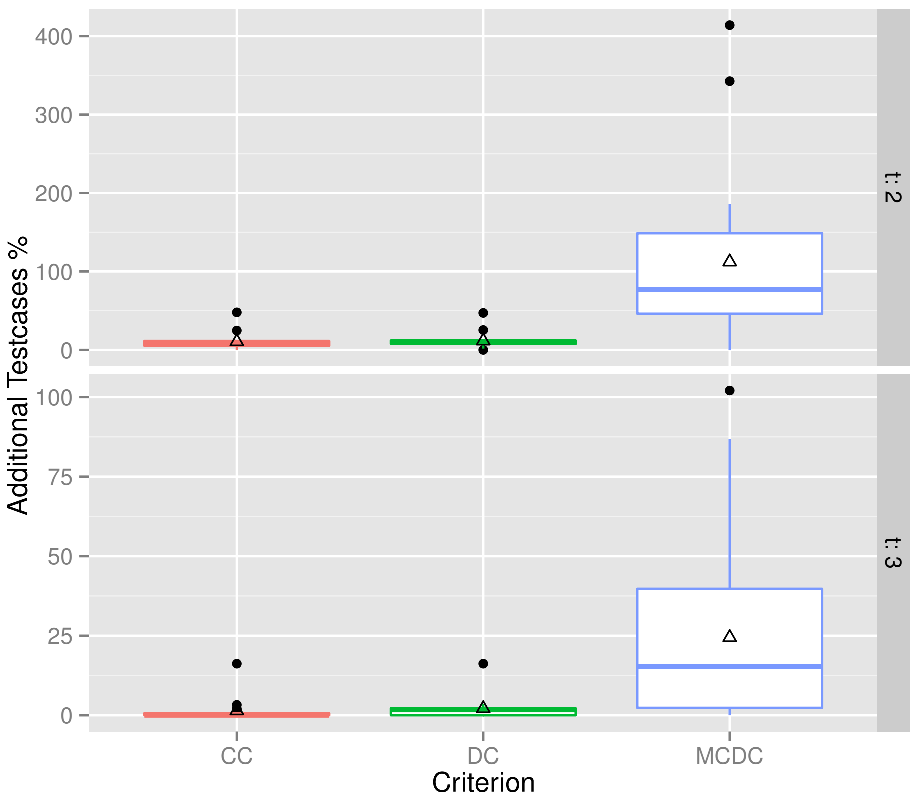

Figure 5.5: Comparison of percentage of additional test cases generated for different coverage strengths and criterion

t	CC	DC	MCDC
2	10.5	11.5	112.2
3	1.4	2.1	24.5

Table 5.4: Mean % additional test cases

t	2	3
size	15	41

Table 5.5: Mean CA size

Figure 5.5 illustrates a percentage comparison of additional test cases needed to complement a given t-way covering array for complete structural coverage of virtual options. The mean percentage of test cases for DC, CC and MC/DC for 2-way and 3-way are described in Table 5.4. It is apparent that CC and DC require fewer test cases than MC/DC. The abnormaly huge mean percentage of test cases for MC/DC is due to the high number of unsatisfied constraints subject applications, due to suffering of CAs. Thus, during additional test cases generation runs for MC/DC, a larger percentage of additional test cases is required, due to the nature of the unsatisfied constraints. However, for 3-way the proportion of such unsatisfied constraints is much lower and requires compartively fewer percentage of additional test cases. From this data it is quite evident that 3-way covering arrays provide significantly better structural coverage than 2-way requiring small percentage of additional test cases.

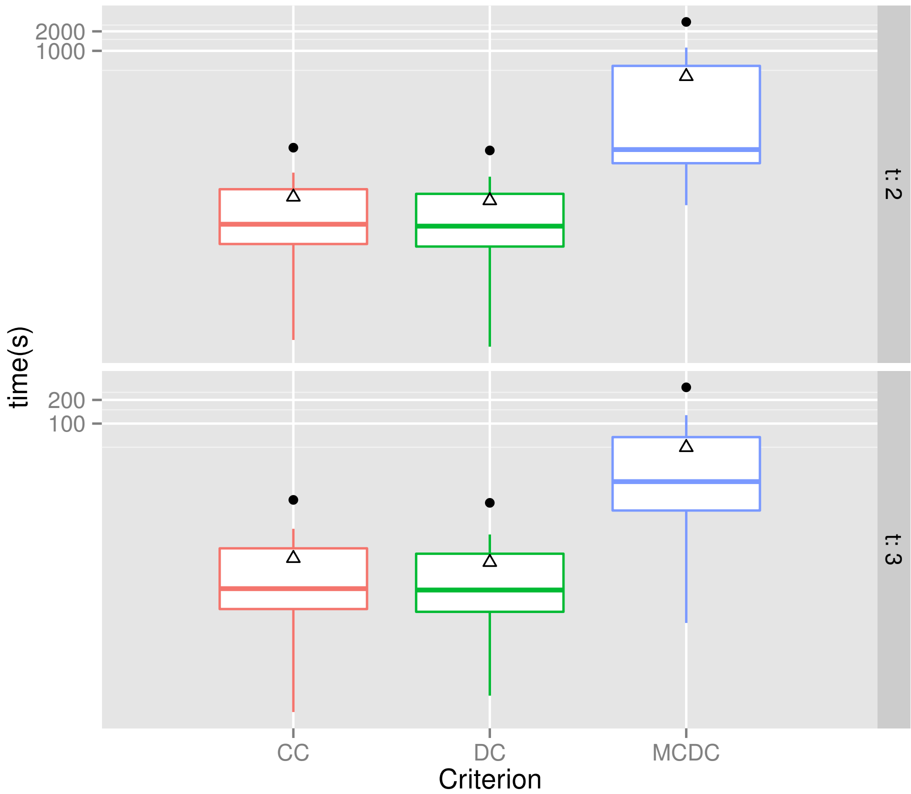

Figure 5.6: Comparison to timings for additional test case generation across different strengths and criterion

t	CC	DC	MCDC
2	5.6	4.9	403.6
3	1.9	1.7	24.5

Table 5.6: Mean test cases construction time(s)

Figure 5.6 illustrates a comparison of timings for additional testcase generation for percentage and 3-way. The mean time for the criterion for percentage and 3-way is shown in Table5.6. Both DC and CC additional test generation process have comparative mean timings while MC/DC requires a longer time, attributed to high suffering of CAs when the test criterion is complex.

5.5.4. Study 3: Effect of Cyclometic Complexity on Traditional Coverage (1Way VO Coverage)

In this study we've performed a set of experiments over the of subject applications in Table A.2 to determine the effectiveness of coverage provision of CAs for 1-way VO testing, under the effect of cyclometic complexity and reveal their coverage sufferings.

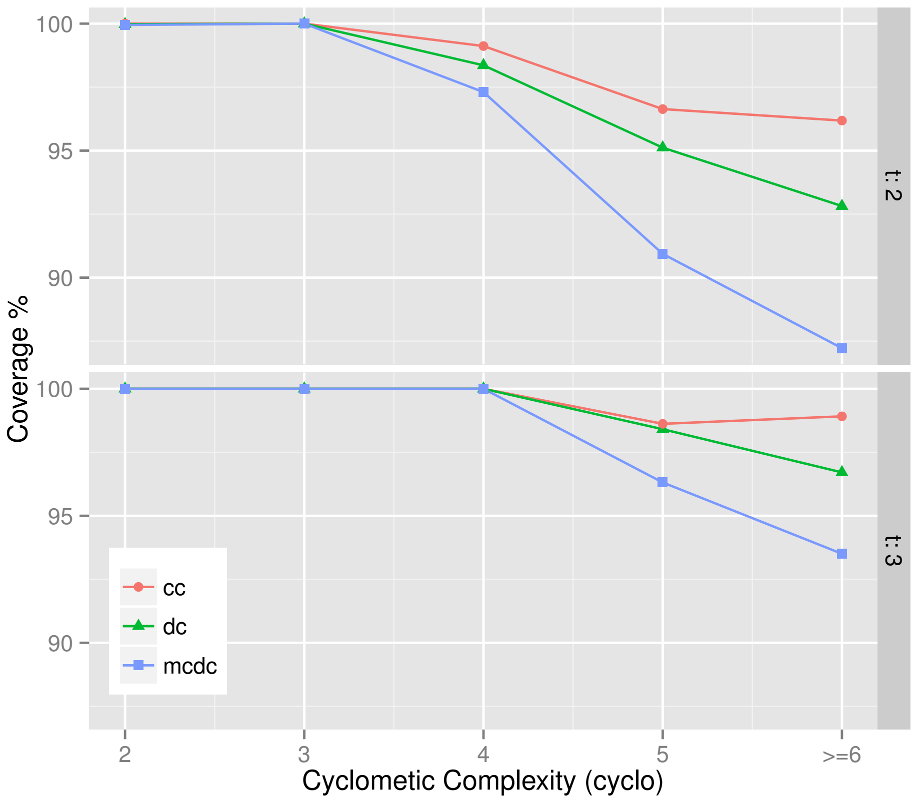

Figure 5.7: Effect of cyclometic complexities on mean coverage across criterion and coverage strengths

Figure 5.7 illustrates the effect of cyclometic complexity on the mean coverage of all the suts for used criterion. From both halves of the figure it is evident that that coverage is getting lesser and lesser as the cyclometic complexity increases. The effect is more pronounced for t=2 than t=3. Upto cyclo=3 and t=2 the subjects observe full coverage but suffers from cyclo=3 onwards, similar is the case for t=3 where the subjects observe full coverage upto cyclo=4. In comparison coverage strength t=3 provides much stronger coverage than t=2, which means its suffers less. The DC and CC follows approximately same coverage till cyclo=4 in t=3 and then start to diverage. Clearly, CAs suffer to provide coverage even under low cyclometic complexities. This plot summarises the effect on coverage w.r.t cyclometic complexity and coverage strengths.

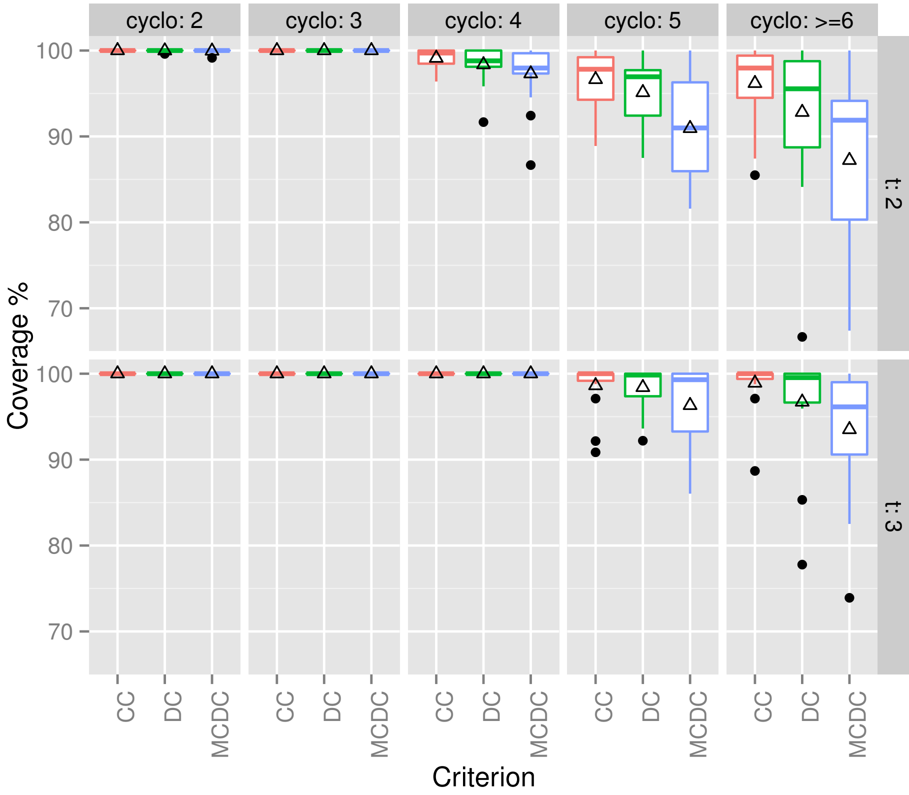

Figure 5.8: Comparison of coverage for different strengths and cyclo levels on the all subjects under all criterion

t	CC	DC	MCDC
2	96.2	92.8	87.2
3	98.3	96.7	93.2

Table 5.7: Cyclo >=6 mean coverage %

Figure5.8 illustrates the effect on coverage across all criterion and different strengths. This plot shows same trend as observed in previous figure in more detail 5.7. The most pronounced effect on affected coverage is visible in cyclo=6 region where the criterion face large variances in coverage especially for percentage.The mean coverages in cyclo=6 for DC, CC and MC/DC for percentage and 3-way are shown in Table 5.7. Among the criterion MC/DC is the most affected one showing the max degree of variance. The significant variance for MC/DC is attributed to the the complexity of criterion and the structure of the virtual options, where CA can't effectively provide complete coverage. The structure of VO is more diverse in terms of logical operators and the number of actual conditions. For instance MC/DC will require more complex test cases to test *(!o1 && o2 || !o3 || o4)* than testing *{(!o1),(o2),(!o3), (o4), (!o1 && o2) and (!o3 || o4)}*.

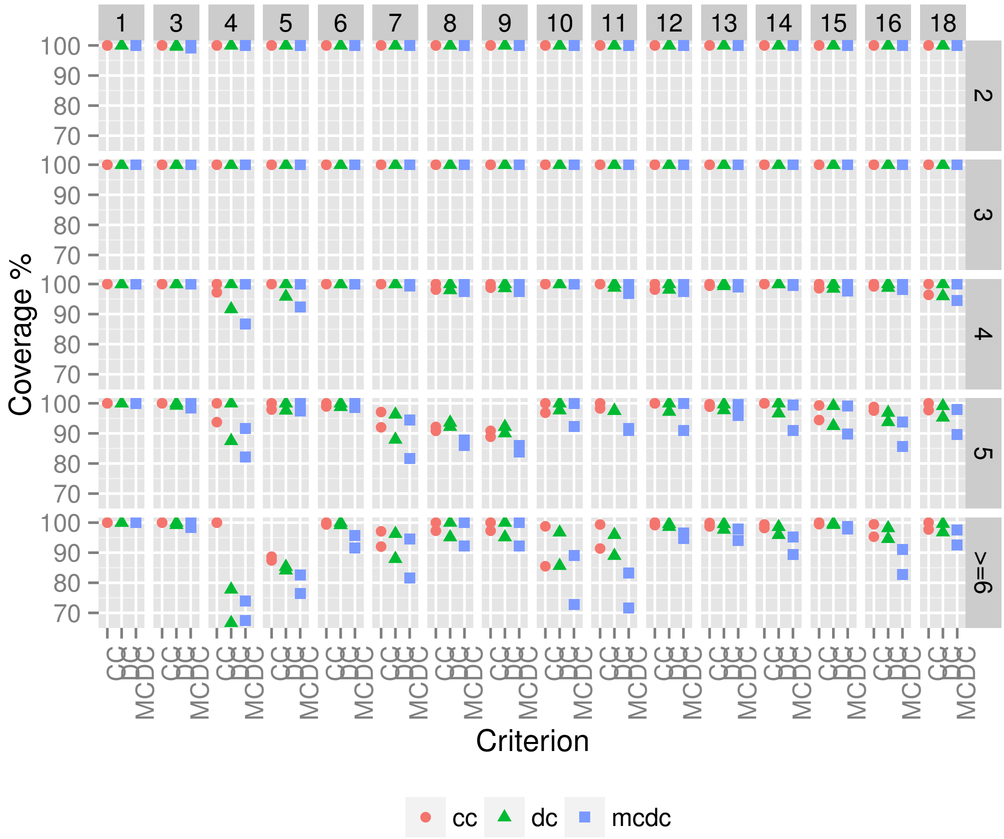

Figure 5.9: Coverage received by each subject application under different criterion

Figure 5.9 illustrates the coverage received by each subject for 2-way and 3-way for all the test criterion across different cyclo levels and elaboration of Figure 5.8. All the subjects receive full coverage for all criterion for cyclo levels 2 and 3. The CAs start to suffer and provide lesser coverage as cyclo levels increase reaching to lowest coverage in cyclo=6 region with high degree of variance.

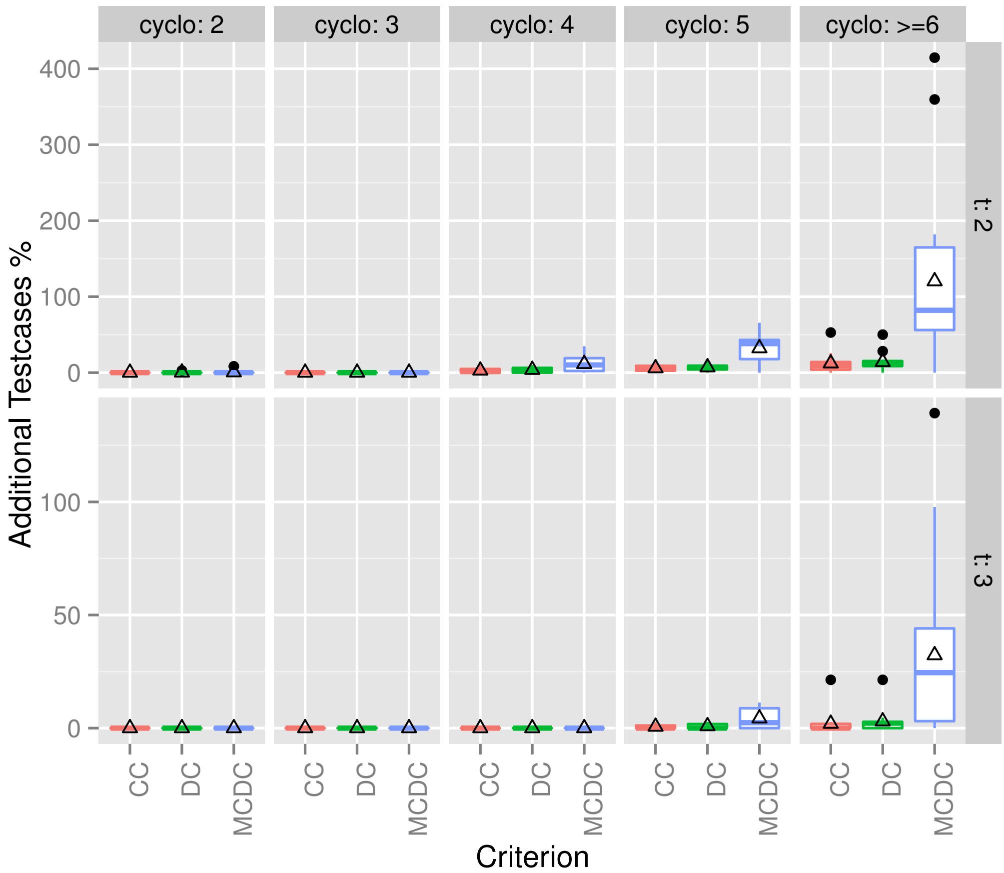

Figure 5.10: Comparison of percentage of addition test cases for full coverage across different cyclo levels

Criteria	**t**	**Cyclo**				
		2	3	4	5	>=6
CC	2	0	0	2.7	5.9	12.2
DC		0.2	0	3.7	7.1	14
MCDC		0.5	0	11.4	32.2	120.3
CC	3	0	0	0	0.6	1.9
DC		0	0	0	0.9	3
MCDC		0	0	0	4.3	32.3

Table 5.8: Mean % additional test cases in Fig 5.10

Figure 5.10 illustrates the comparison of percentage additional test cases in comparison to corresponding size of covering array for each subject. The comparison is shown across different cyclo levels and t strengths. Its is apparent the proportion of additional test cases is increasing across cyclo levels and for 2-way the proportion is more than 3-way. Among the criterion MC/DC is the one requiring the largest degree of additional test cases especially for 2-way. For MC/DC coverage its evident that t-way covering arrays are not suitable especially under high cyclo levels, they suffer the most, which can be observed for cyclo=6 where the proportion of test cases for MC/DC in 2-way on average is 4 times more than 3-way.

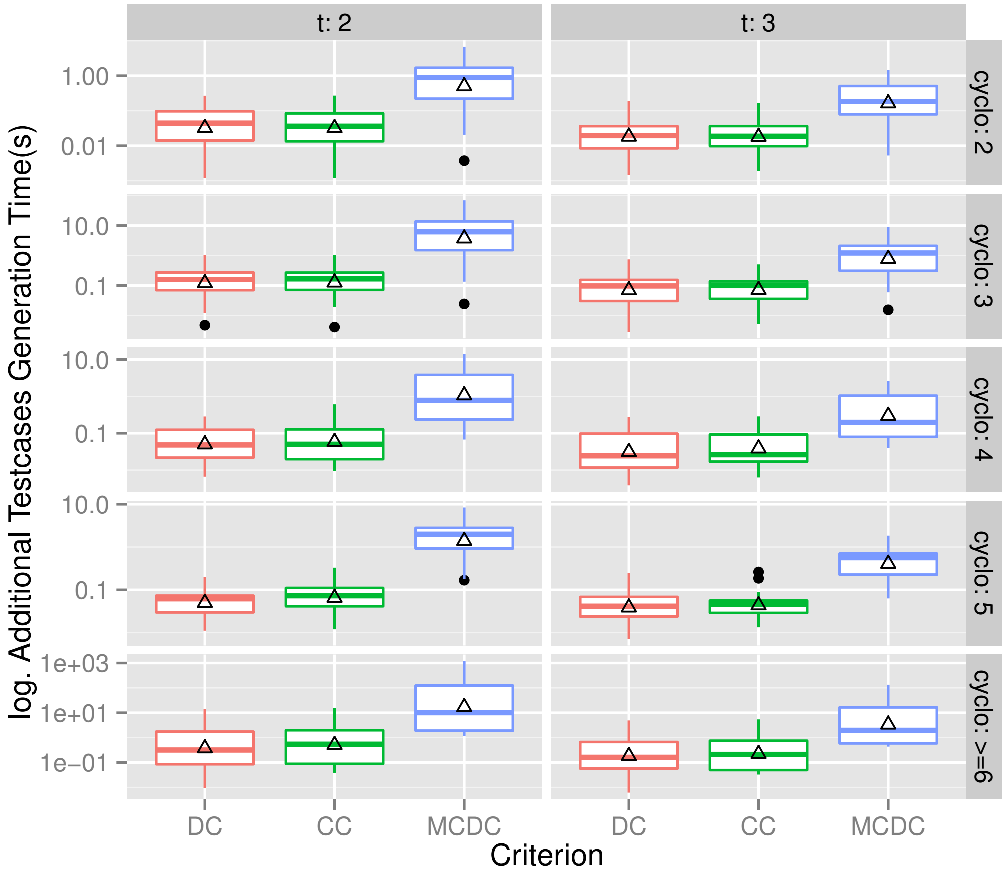

Figure 5.11: Comparison of additional testcase generation time across different cyclo levels, t and criterion

t	CC	DC	MCDC
2	1.7	2.1	139.1
3	0.7	0.8	16.8

Table 5.9: Cyclo>=6 mean additional test cases gen. time(s)

Figure 5.11 illustrates the comparison of additional testcase generation time for different t strengths and cyclo levels. Overall the testcase generation time are incresing as the cyclo level increase and decrease as t increase. The max variation in timings exist in cyclo=6 region due to lower coverage by CAs. Thus, more additional test cases are required to generate especially for t=2. The mean timings across the criterion and t for cyclo=6 are shown in Table 5.9. The overall mean timing for t=2 cyclo6 is 7.8 times more than t=3. The mean percentage of test cases required for cyclo=6 t=2 are approx. 4 times higher than cyclo=6 t=3.

5.5.5. Study 4: t-way Interaction Coverage of Virtual Options Without Cyclo

In this study we've performed a set of experiments over the of subject applications of Table A.2. In these set of experiments the configuration space model of subject application was kept intact, without breaking down into constituent cyclometic complexity segments. This portion of experimentation mainly investigates the sufferings of CAs for t-way interaction testing of virtual options for each of the subject application used. The test criterion in this portion of experimentation are CC and DC without MC/DC for the reasons explained in Chapter 2.

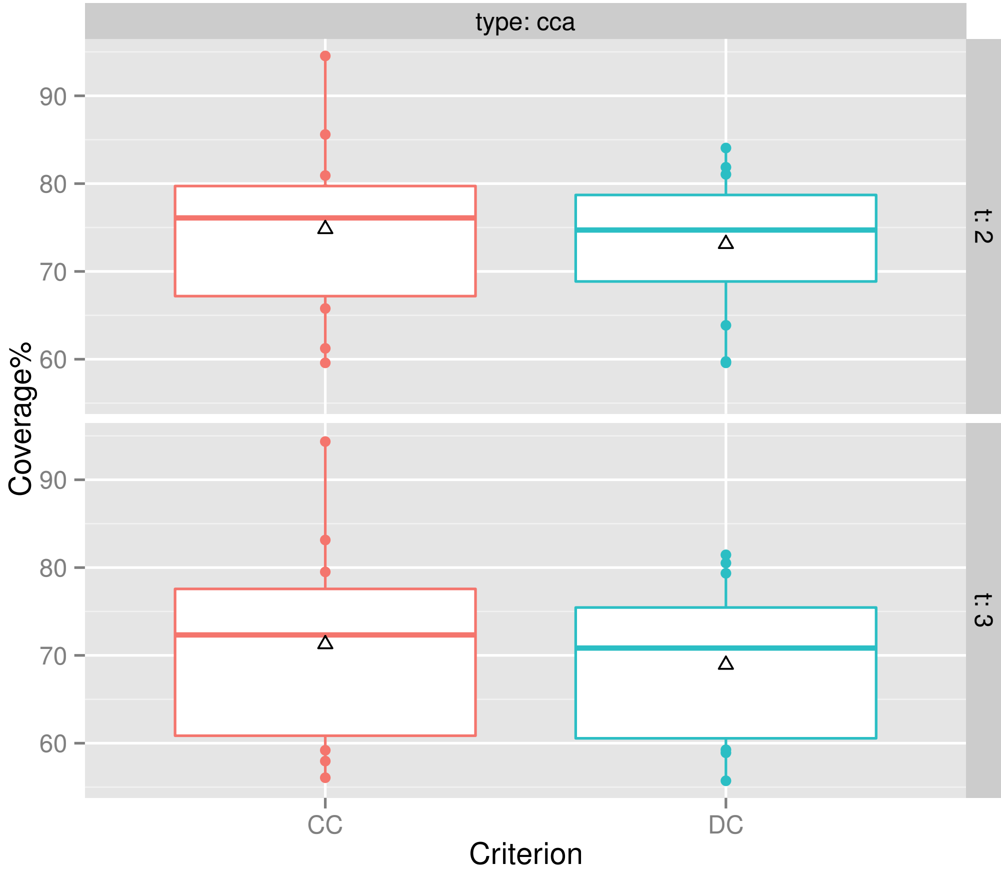

Figure 5.12: Interaction coverage by CCA test suites

t	CC	DC
2	74.8	73.1
3	71.3	68.9

Table 5.10: Mean CCA coverage %

Figure 5.12 presents an overall summary of the coverage provided by CCA for 2-way and 3-way for the subjects. From the figure the CCA test suites suffer in coverage provision which get worse as t is increased. The CCA coverage for both CC and DC is comparable however, DC receives slightly less coverage than CC. The lowest coverage levels under 2-way are approx 60% unlike 3-way where the coverage further drops to 55%. The coverage is further expected to drop significantly as the complexity increased with a higher t-level. That trend is more apparent under the CCA coverages for higher complexity levels in upcoming figures.

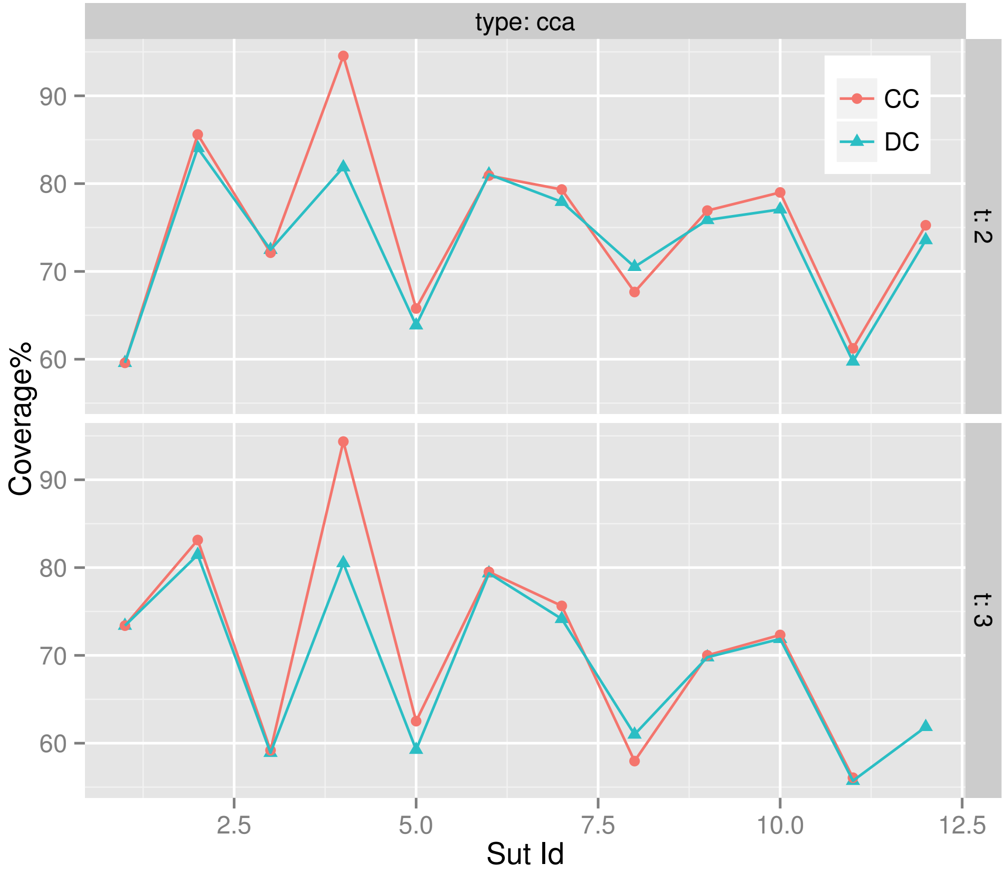

Figure 5.13: Interaction coverage by CCA for individual suts

Figure 5.13 illustrates interaction coverage provided by CCA test suites for all subjects under both coverage criterion for 2-way and 3-way strengths. Generally, the 2-way coverage is better than 3-way coverage for both criterion where CC is closely following DC and performing slightly better than its counterpart. For subject application id=4 CC is significantly better than the corresponding DC for both strengths, that gap is attributed to the high proportion of conditions of this particular subject in the low compelxity region (LcR) vs high complexity region (HcR). This is evident that CCA can provide a better coverage for lower t-strengths but suffer under higher strengths due to an exponential increase in new constraints to satisfy and their associated structural complexity. In comparison 2-way and 3-way interaction testing of virtual options receive much lower coverage than 1-way.

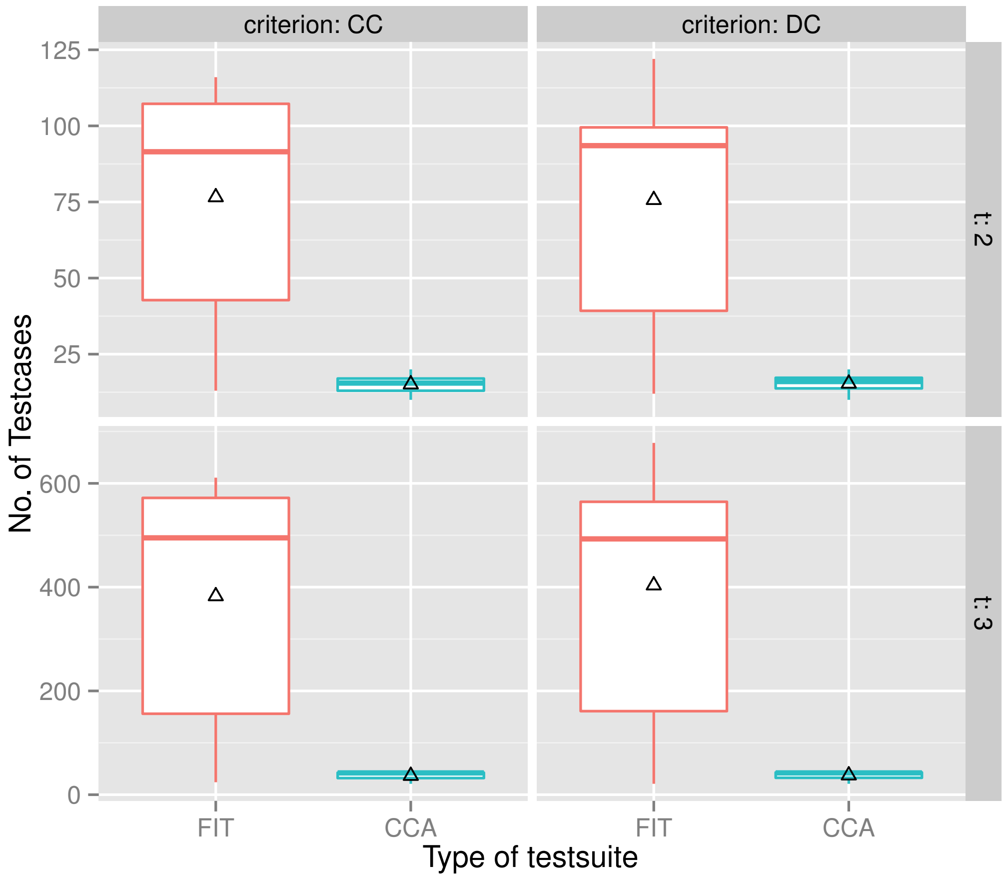

Figure 5.14: Comparison of size of different test suites

t	FIT		CCA	
	CC	*DC*	*CC*	*DC*
2	76.6	75.7	15.1	15.3
3	382.6	403.5	36.4	37.2

Table 5.11: Mean test cases of FIT and CCA test suite

Figure 5.14 presents a comparison of the number of test cases comprising complete coverage FIT test suite vs the corresponding low coverage CCA. Overall, the FIT test suites for 2-way coverages are about 5 times smaller than corresponding counterparts for 3-way way suites. This infact acceptable increase, since for 3-way case to cover the exponential increase in new satisfiable constraints(t-tuples) under both criterion 5x large full coverage test suite is only a fraction to the proportional increase of the number of constraints. The mean sizes of 2-way and 3-way FIT test suites correspondingly are presented in Table 5.11. Depending on the coverage requirements and size of test suite, practitioner has to make a tradeoff in choosing FIT vs CCA based on the coverage requirements of application. If interaction coverage requirements are strict FIT has to be chosen over CCA and vice-versa. Moreover, if the cost of running testcases is low FIT can be choosen over CCA for even applications with flexible coverage requirements.

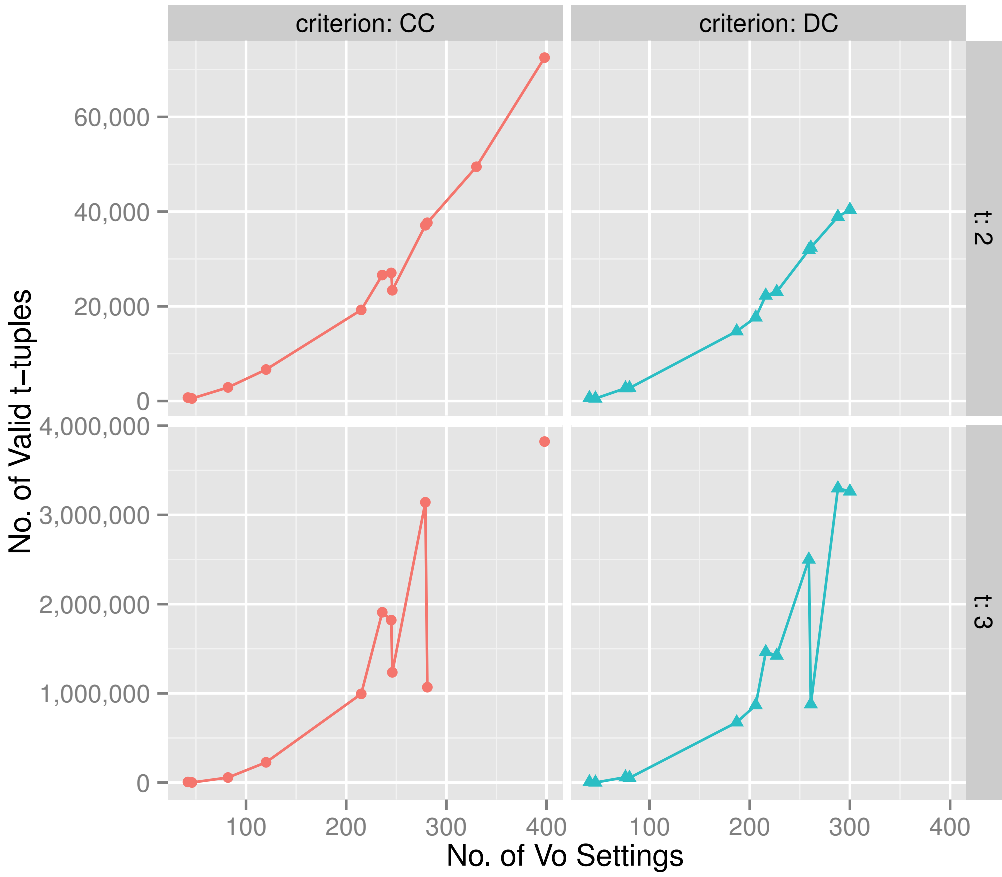

Figure 5.15: Comparison of the count of valid t-tuples w.r.t to VO settings

t-tuples	**CC**	**DC**
2	25320	19004
3	1298559	1207606

Table 5.12: Mean t-tuples count across t and criterion

Figure 5.15 presents a comparison of valid t-way tuples of virtual option settings under the used criterion vs the t-way interactions. In 3-way interaction testing t-tuples of vo-settings have observed an approximate 5 fold increase in comparison to 2-way, for both criterion. The count of t-tuples is comparable in both criterion for fixed t where the count of t-tuples for CC is more than DC. The mean t-tuples for both criterion under 2,3-way interactions is presented in Table 5.12. The number of t-tuples increase exponentially with t. For full coverage under a given t, all of the valid t-tuples should be covered by the test suite. The actual reason behind high sufferings of CAs with increase in t is their significant lack in such configurations that can satisfy this huge population of t-tuples of option settings.

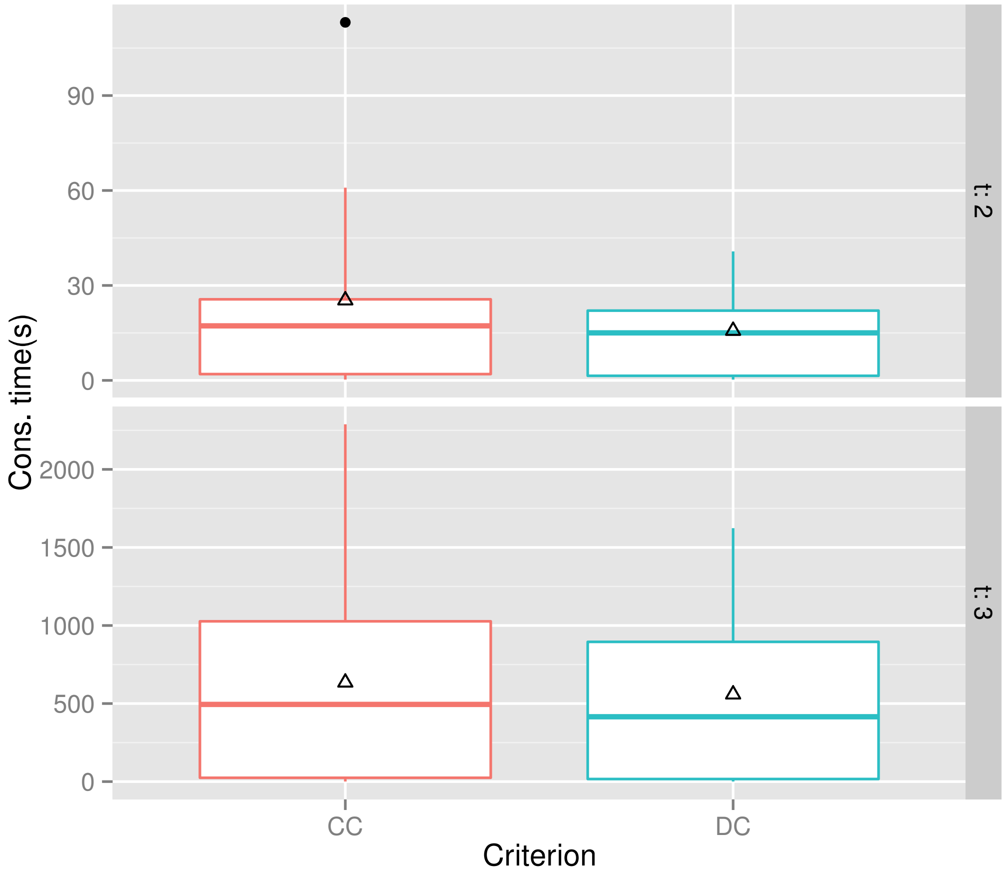

Figure 5.16: Comparison of the construction time of t-way FIT test suite in both criterion for all Suts

t	CC	DC
2	25.3	15.7
3	634.8	558.7

Table 5.13: Mean FIT suite construction time(s)

Figure 5.16 illustrates the comparison of construction time for FIT test suite for 2,3 way interaction testing of subjects. The mean construction time of DC and CC are quite similar but the time linearly scales for 3-way testing. The mean construction time for 2-way and 3-way test suites are presented in Table 5.13. The construction time of 3-way suite is about 25 times larger than 2-way but still under 10 minutes which acceptable. Comparing to the exponential amounts of t-tuples for which the full interaction coverage test suite has to be generated thus, mean time of 10 minutes is reasonable.

5.5.6. Study 5: Effects of Cyclometic Complexity on Interaction Testing

This section of experiments presents the result of experiments for t-way interaction testing of virtual options. This section of experiments investigate the sufferings of CCAs under the effect of different levels of cyclometic complexities in configuration space models of subject applications.

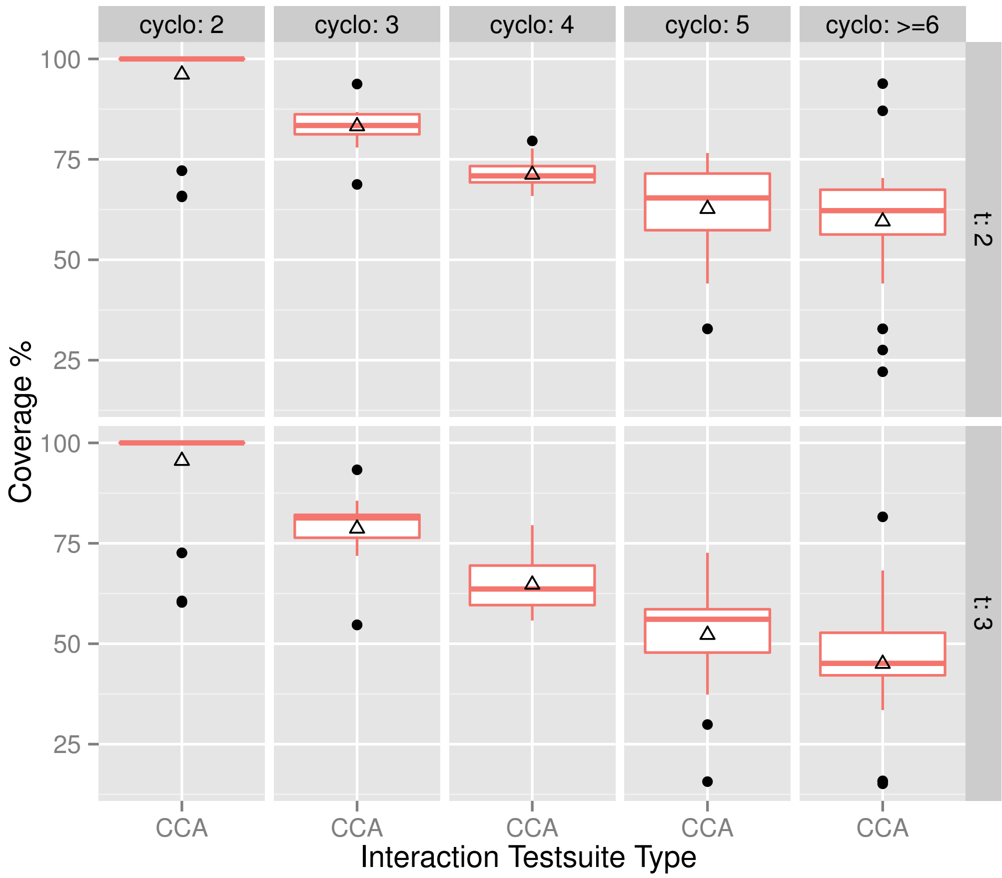

Figure 5.17: Overview of the coverage suffering of CCA across different t and cyclo levels

t	**Cyclo**				
	2	3	4	5	>=6
2	96.1	83.2	71.2	62.7	59.6
3	95.5	78.7	64.7	52.2	45

Table 5.14: Mean % Coverage CCA test suites across cyclos

Figure 5.17 illustrates the percentage coverage provided by CCA across 2-way and 3-way interactions and cyclo levels for all the subjects. The overall coverage of CCA is decreasing as cyclo levels increase and t increase the suffering begins from cyclo=3 unlike cyclo=4 which is earlier than 1-way VO testing. The minimum CCA coverage is revealed in cyclo 6 where the mean coverage of CCA in cyclo=6 2-way vs 3-way respectively is 59.6% and 45% with maximum variance in overall coverage across the subjects, the means are shown in Table 5.17. The reason being due to exponential new t-way interaction tuples to cover which t-way CCA can't provide, due to high tangling. Although CCA can provide significant raw coverage under low cyclometic complexities but it suffers significantly for small increase both t and cyclometic complexity.

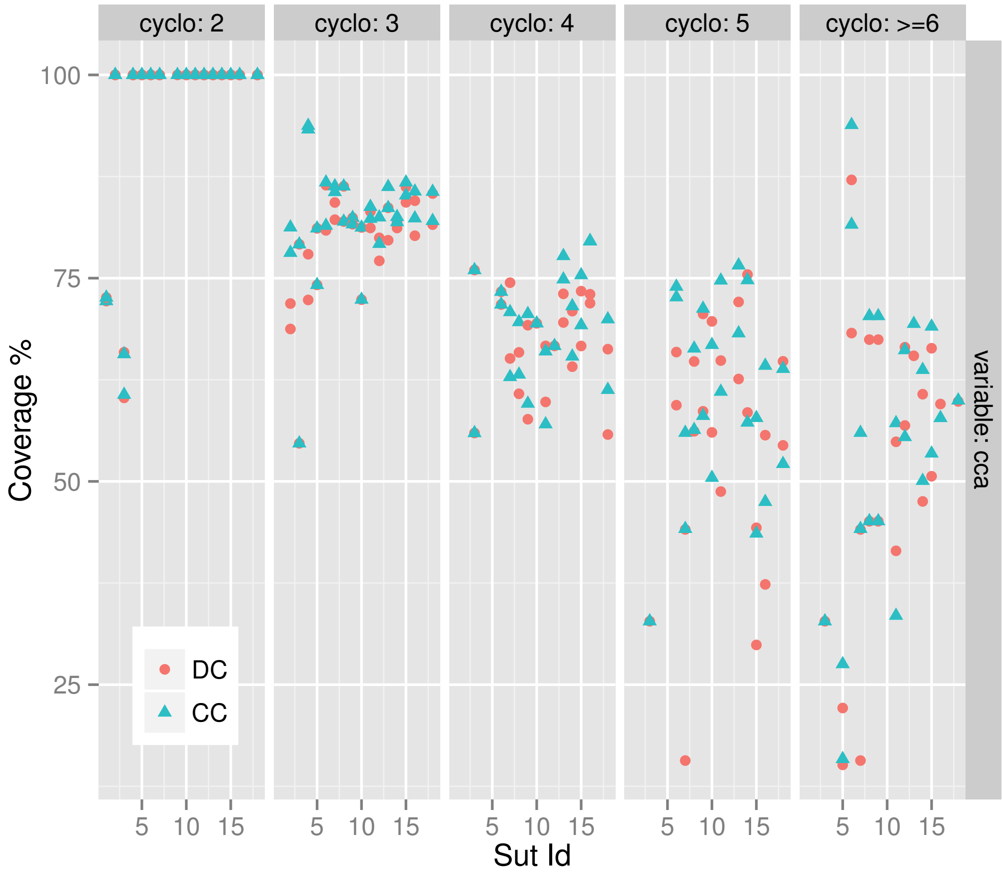

Figure 5.18: CCA coverage for both criterion for each subject against cyclo levels

Figure 5.18 illustrates the degree of coverage provided to each subject across different cyclo levels for both DC and CC criterion. The figure demonstrates partial coverage provided by CCA coverage which can provide effective coverage in cyclo=2 but started to suffer from cyclo=3 onwards. From the figure the cluster of CCA coverage getting more and more scattered and shifting towards lower percent coverage as cyclo increases and observed least coverage for cyclo=6 due to the prominent tangling effect, where the range of coverage is between 15% and 88% with mean coverage of approx 52%. The coverage of CCA has faced significant suffering for cyclo>=6 which indicate the extent to which option tangling can affect coverage.

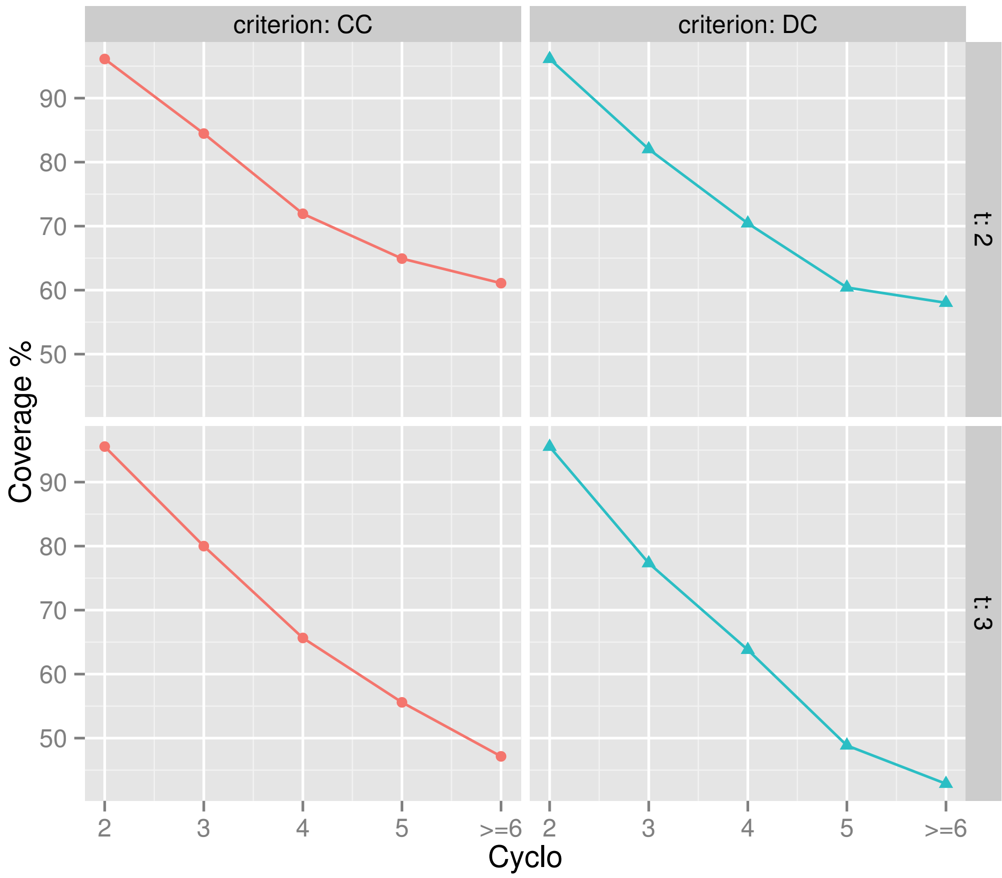

Figure 5.19: Summary of coverage across different cyclo,t levels for both criterion

Figure 5.19 capture the overall summary of coverages provided by CCA against different cyclo levels for the subjects. Criterion wise both DC and CC show very close behaviour for both t levels, but both are significantly affected as t is increased. The range of mean coverage for DC for both 2-way and 3-way is 96%-58% and 95%-42% while the CC receives 96%-61% and 95%-47%. The minimum coverage for 2-way is approx. 60% while 3-way receives approx. 45% coverage. Generally, 3-way observed 15% less coverage than 2-way, which is a quite significant percentage in t-way interaction testing.

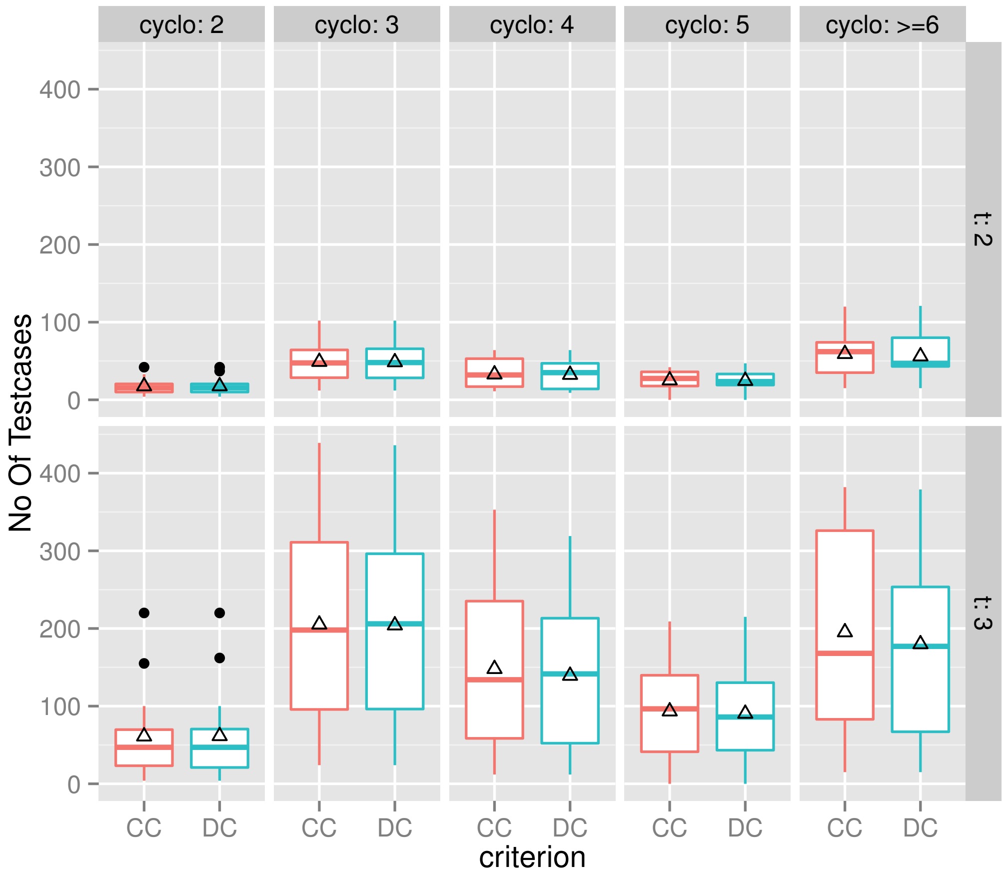

Figure 5.20: Comparison of FIT test suite size for both criterion and different cyclo levels

Crit.	t	Cyclo				
		2	3	4	5	>=6
CC	2	17.5	48.9	32.8	25.1	59.1
DC	2	17.6	48.5	32.2	24.4	56
CC	3	61.1	205.1	147.9	93.3	195.2
DC	3	61.5	204.1	139.2	90.3	179.8

Table 5.15: Mean test cases across cyclos and criterion

Figure 5.20 illustrates the number of test cases composing FIT test suites across different cyclo levels and interaction strength 2 and 3. Its is apparent that 2-way interaction coverage requires much less number of test cases in comparison to 3-way. Criterion wise DC and CC require almost comparable number of test cases. The overall proportion of 3-way test cases is much higher than 2-way which is expected. The range of mean test cases for DC and CC for 2-way across all cyclo level ranges between 18-60 test cases vs 61-200 test cases. In comparison with the number of t-tuples to cover, the FIT test suite size are fraction, but they can provide 100% interaction coverage. The mean test cases are presented in Table 5.15.

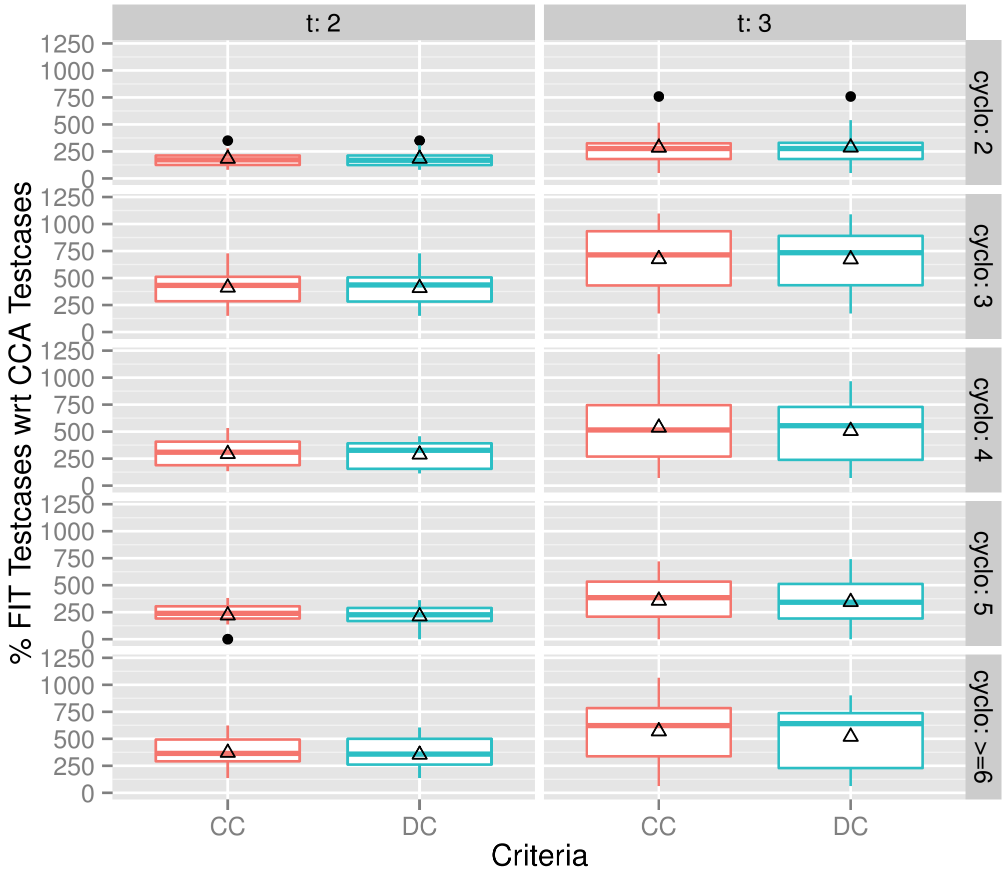

Figure 5.21: Comparison of proportion of FIT test suite vs CCA for all subjects across different cyclo and criterion

Figure 5.21 illustrates the relative size of FIT test suites compared to CCA size, for both criterion and all cyclo levels. For 2-way, the size is expectedly lower than 3-way. The mean percent test cases for 2-way across all cyclo levels range between approx 182-353% in comparison to 3-way test cases which range between 285-520%. The mean test suite size for 3-way is about 1.3 times more than corresponding 2-way test suite which can provide full coverage to orders of valid t-tuples of settings.

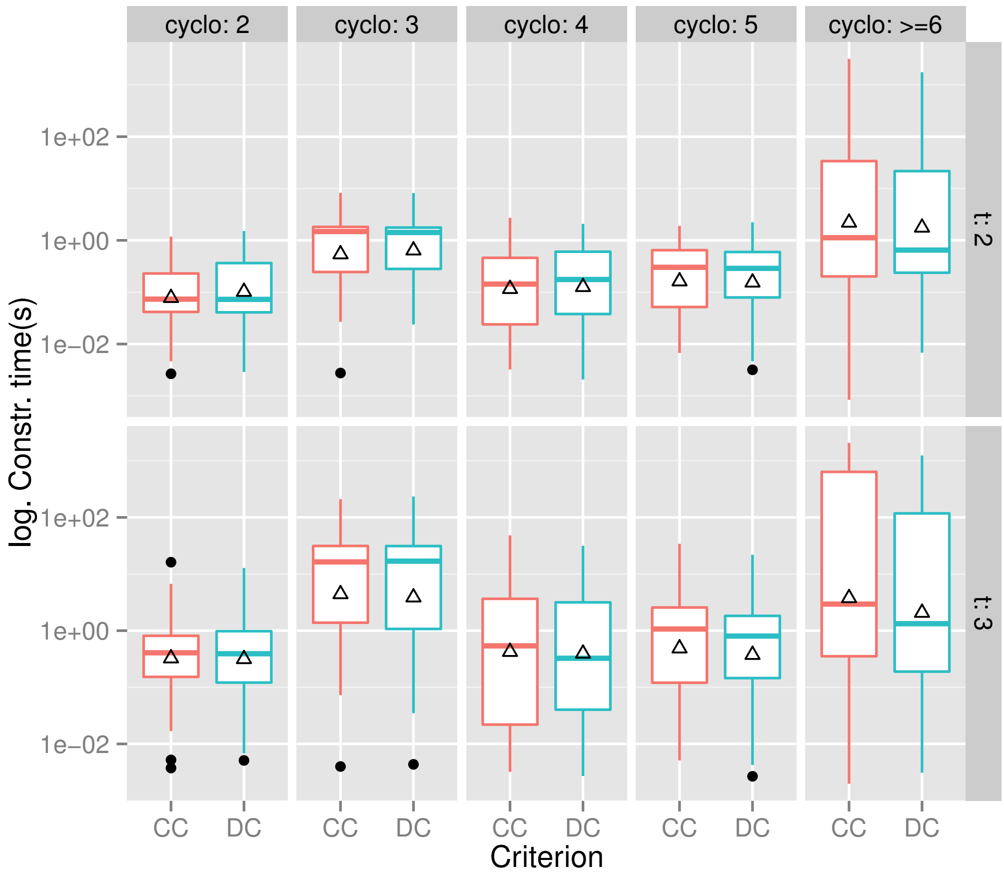

Figure 5.22: Comparison of FIT testsutie generation time

Crit.	**t**	**Cyclo**				
		2	3	4	5	>=6
CC	2	0.2	1.6	0.6	0.5	235.2
DC	2	0.3	1.6	0.6	0.5	128.4
CC	3	1.9	32.3	9	4.1	345.2
DC	3	1.7	31.6	6.3	2.8	206.5

Table 5.16: Mean FIT suite generation times(s)

The Figure 5.22 illustrates a comparison of FIT test suite generation timings for both criterion against 2-way and 3-way interactions across the range of cyclos. The figure reveals overall the test suite generation timing for 2-way is significantly less than 3-way but grows linearly as cyclo levels increase. For both t-levels the max timings are observed in cyclo=6 region where 2-way mean timings across criterion are 128s and 235s as compared to 3-way timings of 345s and 206s. Generally, the test suite generation time for CC is higher than DC which is due to the more virtual option settings to cover for CC. The mean test suite generation timings is under 6 minutes.

5.6. Discussion

In these studies the goal was to investigate the possible use of Covering Arrays (CAs) for the t-way option and interaction testing under a test criteria, for real world highly configurable softwares. Therefore, we had devised two categories of experiments i.e. virtual option testing and t-way virtual option interaction testing.

The t-way virtual option testing revealed on average the covering arrays suffer but 3-way covering arrays can provide sufficient coverage for 1-way virtual option testing for DC and CC. However, there MC/DC coverage was limited. We've found out that complementing both t-2 and 3-way covering arrays with a fraction of test cases can turn them into full coverage 1-way interaction test suites while a quarter-of additional test cases w.r.t to the the size of 3-way CA can turn in to full coverage test suite. However, for 2-way case the CAs suffer greatly providing full coverage and on average require the equivalent number of additional test cases for MC/DC complete coverage.

Covering arrays (CAs) suffer effectively, by the effects of complexity of the options and their tangling. For 1-way testing the suffering effects start from cyclometic complexity level 3 for all criterion given a 2-way CA and same degree of suffering begin for cyclo level 4 in 3-way CA case. From, that segment of experiments we've observed that 3-way CA with additional test cases can be adopted for 1-way virtual option testing.

The t-way virtual option interaction testing with CAs revealed their inadequacy for interaction testing. The 2-way interaction testing performed better than its stronger equivalent 3-way. However, CA faced significant coverage drops for 3-way. The effect was more pronounced with the increasing levels of cyclometic complexity. As the complexity increased from cyclo=2 and onwards the coverage dropped to less than 50%. For software application where testing requirements are flexible and low coverage levels are acceptable, CAs can be adopted. Despite the fact CAs suffer greatly with the only benefit of the shorter generation time or off-the-shelf use. While, the software specific test suites obtained by our approach guarantee to provide complete t-way testing and t-way interaction coverage for any level of option tangling.

6

THREATS TO VALIDITY

In the thesis our primary validity threats are external validity threats that can limit our ability to generalize our results to industrial-grade higly large scale software systems.

The foremost validity threat is the optimal full coverage interaction test suite generation by our algorithm, since our designed algorithm is greedy and generates the test suite specific to a given testsubject for a given criterion, the generated test suite might not be an optimal one, despite our best efforts to come up with an optimal test suite. The reason being the size of full coverage test suite is largely attributed to the subject's complexity and the actual structure of virtual options. In the case of large degree of collisions of constraints under a given criterion, the algorithm will result in a larger test suite. We've made an effort to minimize this effect by optimally placing the new incoming constraints in the pool of contraints in the recent past, that way the probability of constraint collision in minimized and the placement time is tried to minimize while avoiding creation of new contraint pools to the best. We didn't randomize the order of incoming contraints during test generation runs, although the different order of arrival in constraints can result in different overall test suite size. It is believed that an investigation in this regard might reveal optimized test suites.

For the experimentation and analysis we have used 18 different subject applications. Although, all these subject applications are real-world widely used applications however, they still represent limited number of data points. All of the subjects were C based and the configurability mechanisms were C preprocessor(cpp) based which may not comprehensively generalize the results. However, expanding the spectrum of subject applications from other languages and very high configuration complexity might reveal additional limitations.

The timings reported from our interaction test suite generation phases cannot characterize the efficiency of our testgeneration phase and it is believed that more efficient timing statistics might have been obtained if the implementation of our algorithm was based on a non-symbolic computation language. Our implementation was based in Mathematica, despite the flexibility and power of the language its is believed not to be an optimal choice for implementing an industrial grade tool, despite Mathematica excels in certain areas of optimality yet its timing performance as a symbolic computation can not match the traditional languages.

Another potential threat to our approach is the static configurability of the software systems, since in our test subjects we have tested only the virtual options that were statically present in the codebase which can be configured during compilation. However, in the broad spectrum of recent real world softwares the configurablity is also linked with runtime code generation and runtime configurability placement in the code. Its believed that certain application possess limited static configrability and larger runtime configurability which can remain hidden during the static analysis of a code base. It is believed for a complete capture of configuration space, runtime configurability should be taken into account.

Finally, we have not evaluated the fault revealing ability of our approach. However, we believe from [21] work that t-way interaction faults caused by the interaction of virtual options can be effectively revealed by our approach. Mutation testing of faulty variants with seeded interaction faults can be quantatively measured. Investigation in this regard, would have provided empirical evidence of fault revealing ability of our proposed approach.

7

CONCLUSION AND FUTURE WORK

In this thesis we have addressed the problem of t-way and t-wise interaction testing of configuration options (virtual options) in highly configurable software systems and investigated the suitability of covering arrays in this application domain.

Covering Arrays CAs are employed for testing the t-way configurations of applications in a very small number of test configuration during CIT. But they are not appropriate to test configuration options. They suffer under the effect of structural complexity and tangling of options. Failure to provide complete coverage cannot establish test confidence and critical faults might remain still hidden. We've developed an approach to address this issue by proposing a graybox based static analysis of source code of actual real world software applications. We've also proposed novel CIT criterion for t-wise interaction testing of virtual options. The configuration information obtained from the static analysis of source code, guided by the coverage criterion has been utilized to generate testcase that can provide missing or complete coverage in this circumstances. Although, CAs can be used to test the structure of virtual options and high strength covering arrays can provide more than 85% of test coverage. However, the results indicate they were not suitable for interaction coverage and they are significantly affected under the effects of structural complexity and option tangling. CAs cannot provide appropriate t-wise interaction where the coverage drops to less than 50% which get more worse for high values of t. Our greedy approach based tool can generate full coverage t-way and t-wise interaction coverage testsuite specific to a given highly configurable software based on its static analysis of the configurability code.

For 1-way testing of a configuration option under a given testing criterion we had complemented a CA with a fraction of additional testcases for complete coverage. However, for t-way interaction coverage we've generated the subject specific interaction testsuites, whose size is although larger than CA aimed to address, the missing coverages, which CA fail to provide due to their sufferings.

For a future work we intend to investigate on techniques for generation of full coverage optimal testsuites. Moreover, a quantitative investigation of faults caused by the interactions of virtual options will determine the fault revealing ability of the obtained testsuites. Another direction of future work we intend to extend the static analysis of the subject application to dynamic analysis, through which we will be able to capture the runtime configurability and development of techniques that can perform runtime testsuite generation and perform testing activity on the fly.

A

EMPIRICAL RESULTS

This appendix contains the row data from the experiment we have conducted.

Column Name	Description
Sut Id/sutid	Subject Application Id
Sut Name/sutname	Subject Application Name
Version	Version of Sut
Conf.Opts/nco	No of Configuration Options
PncX	Percent Configuration Options in Cyclo:X
PHIcX	Percent High Level Ifs in Cyclo:X
CoI	Complexity Index
NtXca	Size of t:X way CA
tXcc	% CC Coverage for t:X way
tXdc	% DC Coverage for t:X way
tXmcdc	% MCDC Coverage for t:X way
Atc	No. of Additional testcases
AtcT	No. of Additional testcases Construction time(s)
dc	Decision Coverage
cc	Condition Coverage
mcdc	MCDC
Tec	Total constraints to cover
Eco	Constraints covered
nvo	No of Virtual Options
Nvosett	No of Virtual Option Settings
Nunqsett	No of Unique Settings
seedsize	Size of Seed
initT	Initialization Time(s)
consT	Construction Time(s)
Nttup	Total t-tuples
Nvttup	No of valid t-tuples
Nvalcovsd	No of valid t-tuples covered by seed
Ncitts	No of testcases in FIT testsuite
coverage	% Coverage
ccats	No of Testcases in CCA
cacov	% Coverage by CCA

Table A.1: Description of column names

Sut Id	Sut Name	Version	Application	Conf. Opts	Batch
0	berkeleydb	4.7.25	database system	2	1
1	mpsolve	2.2	mathematical solver	14	1
2	dia	0.96.1	diagramming application	15	1
3	irissi	0.8.13	IRC client	30	1
4	xterm	2.4.3	terminal emulator	38	1
5	parrot	0.9.1	virtual machine	51	1
6	pidgin	2.4.0	IM	53	1
7	python	2.6.4	programming language	68	1
8	gimp	2.6.8	graphics manipulator	79	1
9	vim	7.3	text editor	79	1
10	xfig	3.2.5	vector graphics editor	79	1
11	sylpheed	2.6.0	email client	84	1
12	cherokee	1.0.2	web server	97	1
13	privoxy	3.0.12	proxy server	130	2
14	lighthttpd	1.4.22	web server	133	2
15	clamav	0.94.2	antivirus	161	2
16	gnumeric	1.9.5	spreadsheet application	169	2
18	openvpn	2.0.9	security	211	2

Table A.2: Subject applications (SUTs)

sutid	sutname	nco	PNc2	PNc3	PNc4	PNc5	PNc6	PHIc2	PHIc3	PHIc4	PHIc5	PHIc6	PcLcr	PcHcr	PifLcr	PifHcr	CoI
0	berkeleydb	2	50	50	0	0	0	50	50	0	0	0	100	0	100	0	0.0094
1	mpsolve	14	27.45	7.84	21.57	21.57	21.57	0	50	0	0	50	56.86	43.14	50	50	0.05464
2	dia	15	33.33	38.89	27.78	0	0	62.5	25	12.5	0	0	100	0	100	0	0.07109
3	irissi	30	36.59	9.76	9.76	21.95	21.95	0	36.36	36.36	0	27.27	56.1	43.9	72.73	27.27	0.10057
4	xterm	38	22.81	28.07	10.53	14.04	24.56	54.84	25.81	6.45	6.45	6.45	61.4	38.6	87.1	12.9	0.11874
5	parrot	51	14.93	29.85	7.46	7.46	40.3	48.72	30.77	5.13	5.13	10.26	52.24	47.76	84.62	15.38	0.15124
6	pidgin	53	10.29	30.88	16.18	16.18	26.47	16.33	51.02	16.33	8.16	8.16	57.35	42.65	83.67	16.33	0.16427
7	python	68	19	29	14	19	19	49.25	37.31	7.46	0	5.97	62	38	94.03	5.97	0.20616
8	gimp	79	2	36	21	23	18	4.08	51.02	18.37	14.29	12.24	59	41	73.47	26.53	0.26411
9	vim	79	2.97	36.63	20.79	21.78	17.82	6	50	18	14	12	60.4	39.6	74	26	0.26372
10	xfig	79	6.74	16.85	8.99	23.6	43.82	19.35	41.94	9.68	25.81	3.23	32.58	67.42	70.97	29.03	0.26978
11	sylpheed	84	9.91	28.83	9.91	11.71	39.64	20.75	47.17	9.43	9.43	13.21	48.65	51.35	77.36	22.64	0.26554
12	server	97	7.62	13.33	8.57	6.67	63.81	42.55	21.28	6.38	4.26	25.53	29.52	70.48	70.21	29.79	0.33657
13	privoxy	130	5.95	10.81	15.68	14.59	52.97	16.13	18.28	20.43	16.13	29.03	32.43	67.57	54.84	45.16	0.57959
14	lighttpd	133	7.32	16.46	12.2	10.98	53.05	24.64	36.23	13.04	8.7	17.39	35.98	64.02	73.91	26.09	0.4308
15	clamav	161	7.98	28.64	18.78	9.86	34.74	23.02	46.03	12.7	4.76	13.49	55.4	44.6	81.75	18.25	0.50086
16	gnumeric	169	13.04	18.36	13.04	9.66	45.89	32.35	26.47	20.59	6.86	13.73	44.44	55.56	79.41	20.59	0.51172
18	openvpn	211	1.5	16.48	14.23	9.36	58.43	4.39	40.35	17.54	9.65	28.07	32.21	67.79	62.28	37.72	0.83985

Table A.3: Distribution of conf. options and IfBlocks

sutid	nco	Nt2ca	Nt3ca	t2dc	t2cc	t2mcdc	t3dc	t3cc	t3mcdc	t2dcAtc	t3dcAtc	t2ccAtc	t3ccAtc	t2mcdcAtc	t3mcdcAtc
1	14	9.9	21.6	100	100	100	100	100	100	0	0	0	0	0	0
2	15	10	22.3	99.348	99.375	99.091	100	100	100	0.3	0	0.3	0	0.6	0
3	30	12.1	30.3	99.762	100	99.316	100	100	100	0.2	0	0	0	0.8	0
4	38	12.9	33.4	96.667	99.18	88.727	98.59	100	92.909	1.1	0.9	0.6	0	14.3	9.8
5	51	14	37	90.838	92.877	86.79	93.613	94.521	92.325	6.6	6	6.7	6	10.8	8.2
6	53	14	37.6	99.351	99.398	97.157	100	100	98.495	0.8	0	0.8	0	7.7	1
7	68	14.1	41.1	97.545	98.388	94.953	99.545	99.587	99.369	1.7	0.7	1.5	0.7	7.8	1.4
8	79	15.4	42.7	95.703	96.431	94.197	96.92	97.173	96.762	1.4	0.1	1.1	0	7.8	1.5
9	79	15.4	42.7	96.113	96.737	94.91	96.981	97.193	96.848	1.2	0	0.8	0	7	1
10	79	15.4	42.7	93.821	93.937	86.15	98.774	99.173	95.18	3.9	1.5	3.8	1.4	22.9	17.5
11	84	15.7	43.7	96.759	96.65	85	98.862	99.925	91.66	2.1	1	1.8	0.2	65	44.6
12	97	16	45.5	98.488	98.983	95.318	99.691	100	97.624	1.2	1	1.1	0	13.4	10.2
13	130	16.33	50	97.852	98.976	94.653	99.821	99.962	98.815	2	0.666	2	0.333	19	7.667
14	133	17	49.67	96.935	98.249	91.791	99.004	99.546	96.667	2	1	2	1	31.667	21.667
15	161	17.33	52.34	98.714	99.075	97.38	100	100	99.884	1.333	0	1.333	0	8	0.667
16	168	18	53	97.564	98.263	90.814	99.584	99.711	95.371	2	1	1.667	0	61.667	46
18	211	18	55.34	97.865	98.232	94.542	99.806	100	98.397	2	1	2	0	29.333	22

Table A.4: 1-way VO testing

sutid	nco	t2dcAtcT	t3dcAtcT	t2ccAtcT	t3ccAtcT	t2mcdcAtcT	t3mcdcAtcT
1	14	0.076	0.04	0.073	0.042	4.218	0.329
2	15	0.077	0.035	0.075	0.04	4.705	0.382
3	30	0.027	0.038	0.035	0.021	4.157	0.291
4	38	0.274	0.122	0.397	0.166	17.689	3.859
5	51	0.957	0.403	1.047	0.437	21.61	7.84
6	53	1.141	0.454	1.234	0.482	18.481	9.576
7	68	1.428	0.507	1.539	0.581	22.088	11.833
8	79	1.975	0.785	2.118	0.858	29.994	18.265
9	79	2.004	0.763	2.113	0.795	29.931	18.579
10	79	1.677	0.687	2.006	0.788	39.409	16.105
11	84	2.421	0.937	3.374	1.275	78.501	28.739
12	97	3.146	1.178	3.437	1.244	52.12	29.611
13	130	10.61	3.638	12.448	4.191	983.439	102.973
14	133	6.221	2.211	7.335	2.589	586.624	67.272
15	161	11.45	3.886	11.99	4.032	1052.057	112.653
16	168	10.695	3.741	13.229	4.587	1122.96	128.022
18	211	29.075	9.812	32.082	10.693	2793.912	289.159

Table A.5: Additional testcase generation time(s) for 1-way VO testing

sutid	t2dcTec	t2dcEco	t3dcTec	t3dcEco	t2ccTec	t2ccEco	t3ccTec	t3ccEco	t2mcdcTec	t2mcdcEco	t3mcdcTec	t3mcdcEco
1	50	50	50	50	50	50	50	50	55	55	55	55
2	46	45.7	46	46	48	47.7	48	48	66	65.4	66	66
3	84	83.8	84	84	86	86	86	86	117	116.2	117	117
4	78	75.4	78	76.9	122	121	122	122	165	146.4	165	153.3
5	191	173.5	191	178.8	219	203.4	219	207	271	235.2	271	250.2
6	231	229.5	231	231	249	247.5	249	249	299	290.5	299	294.5
7	220	214.6	220	219	242	238.1	242	241	317	301	317	315
8	263	251.7	263	254.9	283	272.9	283	275	386	363.6	386	373.5
9	265	254.7	265	257	285	275.7	285	277	387	367.3	387	374.8
10	212	198.9	212	209.4	254	238.6	254	251.9	361	311	361	343.6
11	290	280.6	290	286.7	400	386.6	400	399.7	530	450.5	530	485.8
12	324	319.1	324	323	354	350.4	354	354	425	405.1	425	414.9
13	745	729	745	743.667	879	870	879	878.667	1041	985.333	1041	1028.667
14	435	421.667	435	430.667	514	505	514	511.667	670	615	670	647.667
15	648	639.667	648	648	685	678.667	685	685	865	842.333	865	864
16	561	547.333	561	558.667	691	679	691	689	929	843.667	929	886
18	1202	1176.333	1202	1199.667	1320	1296.667	1320	1320	1539	1455	1539	1514.333

Table A.6: Constraints to cover for 1-way VO testing

sutid	t	criterion	nco	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttup	Nttup	Nvalcovsd	Ncitts	coverage	ccats	cacov
1	2	DC	14	3	46	26	0	0.021132	0.232846	560	532	0	12	100	10	59.586
1	2	CC	14	3	46	26	0	0.032696	0.229455	560	532	0	13	100	10	59.586
2	2	DC	15	10	40	32	0	0.048759	0.371819	712	646	0	24	100	10	84.056
2	2	CC	15	10	42	34	0	0.043229	0.332827	788	722	0	24	100	10	85.596
3	2	DC	30	10	80	66	0	0.12703	1.279258	2772	2738	0	37	100	13	72.462
3	2	CC	30	10	82	68	0	0.136258	1.338096	2908	2874	0	33	100	13	72.129
4	2	DC	38	30	76	66	0	0.14193	1.31777	2772	2707	0	40	100	14	81.862
4	2	CC	38	30	120	82	0	0.166388	1.995473	6792	6650	0	46	100	13	94.541
5	2	DC	51	28	187	134	0	0.545496	11.390728	15902	14735	0	97	100	20	63.862
5	2	CC	51	28	215	147	0	0.639084	14.788133	20754	19242	0	97	100	20	65.778
6	2	DC	53	42	227	155	0	0.691276	17.398808	25000	23068	0	80	100	14	81.056
6	2	CC	53	42	245	167	0	0.724391	21.406398	29062	27059	0	86	100	14	80.919
7	2	DC	68	48	216	152	0	0.594998	10.851623	22768	22282	0	98	100	15	77.924
7	2	CC	68	48	236	172	0	0.769003	13.303258	27136	26600	0	109	100	15	79.327
8	2	DC	79	27	206	170	0	0.982423	35.15333	18248	17666	0	104	100	19	70.52
8	2	CC	79	27	246	204	0	1.555198	59.342436	24020	23392	0	98	100	19	67.656
9	2	DC	79	48	261	215	0	1.065085	17.41591	33178	32423	0	106	100	17	75.857
9	2	CC	79	48	281	227	0	1.001	18.033108	38438	37653	0	116	100	16	76.926
10	2	DC	79	47	259	215	0	1.691474	18.4625	32660	31918	0	122	100	17	77.066
10	2	CC	79	47	279	227	0	1.255302	22.576201	37880	37110	0	108	100	17	78.995
11	2	DC	84	47	288	228	111	1.251938	39.506549	39825	38925	0	98	100	18	59.741
11	2	CC	84	47	398	300	110	1.994473	111.130635	74187	72526	0	107	100	17	61.236
12	2	DC	97	27	300	212	0	1.329274	26.395684	41820	40410	0	90	100	17	73.568
12	2	CC	97	27	330	238	0	1.31702	29.669726	50916	49478	0	82	100	17	75.258

Table A.7: FIT generation, coverage and CCA coverage for t=2

sutid	t	criterion	nco	nvo	Nvosett	Nunqsett	sdsize	initT	consT	Nttup	Nttup	Nvalcovsd	Ncitts	coverage	ccats	cacov
1	3	DC	14	3	46	26	12	0.027307	0.553085	1568	1176	627	21	100	21	73.384
1	3	CC	14	3	46	26	13	0.035799	0.63778	1568	1176	608	24	100	21	73.384
2	3	DC	15	10	40	32	24	0.047813	1.839529	7424	5710	4068	84	100	22	81.454
2	3	CC	15	10	42	34	24	0.053741	1.894323	8696	6850	4829	87	100	22	83.139
3	3	DC	30	10	80	66	37	0.128374	16.476464	54880	52820	34598	162	100	31	58.908
3	3	CC	30	10	82	68	33	0.134066	16.201585	58792	56664	39442	155	100	31	59.195
4	3	DC	38	30	76	66	40	0.156781	15.834845	64640	60218	44623	158	100	33	80.514
4	3	CC	38	30	120	82	46	0.177578	32.484021	242016	226980	183187	157	100	33	94.353
5	3	DC	51	28	187	134	97	0.523584	281.672824	826464	675394	505500	563	100	43	59.254
5	3	CC	51	28	215	147	97	0.686926	432.816826	1213480	994485	728426	573	100	43	62.51
6	3	DC	53	42	227	155	80	0.658997	453.508493	1780584	1423958	1129546	433	100	37	79.352
6	3	CC	53	42	245	167	86	0.911831	546.229558	2225024	1821003	1465600	424	100	37	79.508
7	3	DC	68	48	216	152	98	0.605337	376.204007	1560944	1463426	1095780	501	100	42	74.153
7	3	CC	68	48	236	172	109	0.935221	493.475025	2026256	1908784	1435421	495	100	42	75.637
8	3	DC	79	27	206	170	104	1.050267	653.949907	971120	867894	672448	569	100	43	60.991
8	3	CC	79	27	246	204	98	1.720423	968.362671	1368264	1235382	967317	571	100	43	57.964
9	3	DC	79	48	261	215	106	1.073902	885.745721	2738116	878193	1859923	630	100	42	69.788
9	3	CC	79	48	281	227	116	1.045958	1082.439822	3411652	1069000	2342562	611	100	42	70.003
10	3	DC	79	47	259	215	122	1.136124	919.171668	2672796	2500473	1801310	678	100	42	71.878
10	3	CC	79	47	279	227	107	2.321028	1111.588173	3335892	3141900	2363934	578	100	42	72.33
11	3	DC	84	47	288	228	582	1.193048	1469.085732	3527878	3298507	2618698	558	100	45	55.719
11	3	CC	84	47	398	300	611	2.808477	2286.080856	4589805	3821922	2749322	534	100	44	56.07
12	3	DC	97	27	300	212	90	1.222645	1622.267003	3605968	3263504	2242449	485	100	46	61.859
12	3	CC	97	27	330	238	82	NA	NA	NA	NA	NA	NA	NA	NA	NA

Table A.8: FIT generation, coverage and CCA coverage for t=3

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	2	DC	3	46	26	0	0.040608	0.468214	560	532	0	12	100	10	72.18
1	2	CC	3	46	26	0	0.040786	0.421898	560	532	0	13	100	10	72.18
2	2	DC	9	18	12	0	0.012082	0.059354	144	128	0	10	100	8	100
2	2	CC	9	18	12	0	0.016244	0.056482	144	128	0	10	100	8	100
3	2	DC	10	80	66	0	0.221882	1.295538	2772	2738	0	37	100	12	65.887
3	2	CC	10	82	68	0	0.084632	1.0964	2908	2874	0	33	100	12	65.657
4	2	DC	16	32	24	0	0.039411	0.120209	480	460	0	19	100	10	100
4	2	CC	16	32	24	0	0.031903	0.198462	480	460	0	19	100	10	100
5	2	DC	18	36	18	0	0.022972	0.181812	612	528	0	15	100	9	100
5	2	CC	18	36	18	0	0.022023	0.159172	612	528	0	15	100	9	100
6	2	DC	7	14	12	0	0.00778	0.02811	84	80	0	10	100	8	100
6	2	CC	7	14	12	0	0.014678	0.058739	84	80	0	10	100	8	100
7	2	DC	32	64	36	0	0.074503	0.600109	1984	1876	0	27	100	10	100
7	2	CC	32	64	36	0	0.024241	0.205273	1984	1876	0	28	100	10	100
8	2	DC	1	2	2	0	0.004416	0.151466	NA	NA	NA	NA	NA	4	NA
8	2	CC	1	2	2	0	0.001458	0.001209	NA	NA	NA	NA	NA	4	NA
9	2	DC	2	4	4	0	0.001034	0.001849	4	4	0	4	100	5	100
9	2	CC	2	4	4	0	0.001705	0.002941	4	4	0	4	100	5	100
10	2	DC	5	10	10	0	0.002938	0.01052	40	40	0	9	100	8	100
10	2	CC	5	10	10	0	0.002775	0.009659	40	40	0	9	100	8	100
11	2	DC	10	20	20	0	0.007906	0.03293	180	180	0	16	100	9	100
11	2	CC	10	20	20	0	0.007965	0.033672	180	180	0	16	100	9	100
12	2	DC	19	38	16	0	0.007266	0.065412	684	620	0	13	100	8	100
12	2	CC	19	38	16	0	0.007438	0.07024	684	620	0	13	100	8	100
13	2	DC	14	28	22	0	0.010008	0.052086	364	340	0	18	100	9	100
13	2	CC	14	28	22	0	0.011545	0.059093	364	340	0	18	100	9	100
14	2	DC	16	32	22	0	0.010501	0.058368	480	444	0	17	100	10	100
14	2	CC	16	32	22	0	0.010249	0.062608	480	444	0	18	100	10	100
15	2	DC	28	56	32	0	0.067562	0.44261	1512	1416	0	25	100	10	100
15	2	CC	28	56	32	0	0.023886	0.20109	1512	1416	0	25	100	10	100
16	2	DC	32	64	54	0	0.052299	0.312172	1984	1960	0	42	100	12	100
16	2	CC	32	64	54	0	0.218659	0.679556	1984	1960	0	42	100	12	100
18	2	DC	4	8	8	0	0.01216	0.009605	24	24	0	7	100	6	100
18	2	CC	4	8	8	0	0.00223	0.006842	24	24	0	7	100	6	100

Table A.9: FIT generation, coverage and CCA coverage for t=2 and cyclo=2

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	3	DC	3	46	26	12	0.0397	0.926303	1568	1176	627	21	100	23	72.619
1	3	CC	3	46	26	13	0.042528	0.768779	1568	1176	608	24	100	23	72.619
2	3	DC	9	18	12	10	0.018811	0.165925	672	464	341	20	100	13	100
2	3	CC	9	18	12	10	0.015189	0.21156	672	464	341	20	100	13	100
3	3	DC	10	80	66	37	0.07893	12.729335	54880	52820	34598	162	100	30	60.276
3	3	CC	10	82	68	33	0.130359	16.065535	58792	56664	39442	155	100	30	60.666
4	3	DC	16	32	24	19	0.031775	0.947609	4480	3936	2746	66	100	21	100
4	3	CC	16	32	24	19	0.012669	0.485121	4480	3936	2746	66	100	21	100
5	3	DC	18	36	18	15	0.018619	0.772329	6528	4160	3228	39	100	19	100
5	3	CC	18	36	18	15	0.00781	0.661911	6528	4160	3206	39	100	19	100
6	3	DC	7	14	12	10	0.011026	0.110107	280	240	136	26	100	14	100
6	3	CC	7	14	12	10	0.010529	0.142158	280	240	136	26	100	14	100
7	3	DC	32	64	36	27	0.025764	2.788713	39680	33536	24642	84	100	25	100
7	3	CC	32	64	36	28	0.076409	3.128014	39680	33536	25184	81	100	25	100
8	3	DC	1	2	2	0	0.002555	0.004244	NA	NA	NA	NA	NA	1	NA
8	3	CC	1	2	2	0	0.001455	0.002293	NA	NA	NA	NA	NA	1	NA
9	3	DC	2	4	4	4	0.001612	0.003495	NA	NA	NA	4	NA	8	NA
9	3	CC	2	4	4	4	0.001624	0.003591	NA	NA	NA	4	NA	8	NA
10	3	DC	5	10	10	9	0.002751	0.025657	80	80	44	21	100	13	100
10	3	CC	5	10	10	9	0.013459	0.114181	80	80	44	21	100	13	100
11	3	DC	10	20	20	16	0.008029	0.183087	960	960	538	53	100	18	100
11	3	CC	10	20	20	16	0.010821	0.185921	960	960	538	53	100	18	100
12	3	DC	19	38	16	13	0.01829	0.820039	7752	5656	4190	42	100	14	100
12	3	CC	19	38	16	13	0.00835	0.477751	7752	5656	4190	42	100	14	100
13	3	DC	14	28	22	18	0.009586	0.315835	2912	2400	1552	59	100	18	100
13	3	CC	14	28	22	18	0.017773	0.389015	2912	2400	1556	59	100	18	100
14	3	DC	16	32	22	17	0.010153	0.381152	4480	3552	2475	52	100	20	100
14	3	CC	16	32	22	18	0.010508	0.383487	4480	3552	2522	52	100	20	100
15	3	DC	28	56	32	25	0.022309	1.880015	26208	21504	15427	100	100	24	100
15	3	CC	28	56	32	25	0.08709	2.28494	26208	21504	15427	100	100	24	100
16	3	DC	32	64	54	42	0.051822	6.003636	39680	38256	21569	220	100	29	100
16	3	CC	32	64	54	42	0.221479	6.488815	39680	38256	21569	220	100	29	100
18	3	DC	4	8	8	7	0.002909	0.01878	32	32	17	15	100	8	100
18	3	CC	4	8	8	7	0.002217	0.014624	32	32	17	15	100	8	100

Table A.10: FIT generation, coverage and CCA coverage for t=3 and cyclo=2

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	2	DC	1	4	4	0	0.001948	0.124876	NA	NA	NA	NA	NA	6	NA
1	2	CC	1	4	4	0	0.001889	0.000869	NA	NA	NA	NA	NA	6	NA
2	2	DC	3	10	10	0	0.005638	0.018196	32	32	0	12	100	8	68.75
2	2	CC	3	12	12	0	0.008125	0.018639	48	48	0	12	100	8	81.25
3	2	DC	3	12	12	0	0.005213	0.027196	48	48	0	15	100	8	79.167
3	2	CC	3	12	12	0	0.006572	0.037078	48	48	0	15	100	8	79.167
4	2	DC	7	18	18	0	0.012445	0.04223	136	136	0	15	100	10	77.941
4	2	CC	7	28	28	0	0.027363	0.115038	336	336	0	22	100	10	93.75
5	2	DC	11	44	44	0	0.072646	0.345469	880	874	0	43	100	11	81.121
5	2	CC	11	44	44	0	0.063871	0.413086	880	874	0	40	100	11	81.121
6	2	DC	24	93	65	0	0.09745	1.458183	4142	3620	0	43	100	10	86.436
6	2	CC	24	95	65	0	0.130052	1.467931	4324	3762	0	43	100	10	86.789
7	2	DC	24	88	66	0	0.13413	1.283315	3704	3568	0	54	100	12	84.305
7	2	CC	24	96	70	0	0.144229	1.450997	4416	4266	0	52	100	12	86.31
8	2	DC	24	96	92	0	0.191371	1.572174	4416	4348	0	65	100	13	86.293
8	2	CC	24	96	92	0	0.171581	1.715615	4416	4348	0	62	100	13	86.293
9	2	DC	24	96	92	0	0.222171	2.190943	4416	4358	0	68	100	13	82.423
9	2	CC	24	96	92	0	0.185509	1.853135	4416	4358	0	72	100	13	82.423
10	2	DC	12	48	30	0	0.065498	0.423383	1056	890	0	29	100	10	81.236
10	2	CC	12	48	30	0	0.028993	0.286651	1056	890	0	30	100	10	81.236
11	2	DC	24	94	84	0	0.138946	1.532968	4232	4140	0	57	100	12	83.188
11	2	CC	24	96	86	0	0.150368	1.614606	4416	4322	0	60	100	12	83.781
12	2	DC	9	34	26	0	0.061582	0.219482	512	484	0	26	100	10	79.959
12	2	CC	9	36	28	0	0.022494	0.222394	576	548	0	24	100	10	82.482
13	2	DC	16	58	46	0	0.112497	0.848603	1572	1432	0	42	100	11	83.659
13	2	CC	16	64	46	0	0.060592	0.882099	1920	1728	0	41	100	11	86.227
14	2	DC	24	94	62	0	0.200636	1.337673	4232	4044	0	53	100	12	81.85
14	2	CC	24	96	64	0	0.083131	1.393922	4416	4226	0	56	100	12	82.537
15	2	DC	57	224	148	0	0.507527	7.673572	24644	23899	0	102	100	14	86.213
15	2	CC	57	228	150	0	0.439112	7.779601	25536	24775	0	102	100	14	86.785
16	2	DC	26	98	82	0	0.415729	1.467079	4612	4549	0	75	100	13	84.524
16	2	CC	26	104	84	0	0.166911	1.648394	5200	5122	0	71	100	13	85.67
18	2	DC	45	178	121	0	0.588073	4.059063	15488	14844	0	77	100	13	85.408
18	2	CC	45	180	122	0	0.250393	4.253563	15840	15164	0	81	100	13	85.637

Table A.11: FIT generation, coverage and CCA coverage for t=2 and cyclo=3

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	3	DC	1	4	4	0	0.002189	0.002162	NA	NA	NA	NA	NA	8	NA
1	3	CC	1	4	4	0	0.002016	0.001973	NA	NA	NA	NA	NA	8	NA
2	3	DC	3	10	10	12	0.00566	0.029151	32	32	13	24	100	14	71.875
2	3	CC	3	12	12	12	0.009416	0.063439	64	64	28	24	100	14	78.125
3	3	DC	3	12	12	15	0.005389	0.056427	64	64	28	33	100	14	54.688
3	3	CC	3	12	12	15	0.007737	0.074978	64	64	28	33	100	14	54.688
4	3	DC	7	18	18	15	0.016087	0.248949	560	560	323	45	100	23	72.321
4	3	CC	7	28	28	22	0.019901	0.451911	2240	2240	1246	50	100	23	93.304
5	3	DC	11	44	44	43	0.047672	3.258727	10560	10344	5355	167	100	25	74.169
5	3	CC	11	44	44	40	0.052348	3.403837	10560	10344	5576	168	100	25	74.169
6	3	DC	24	93	65	43	0.092812	16.781499	117568	82203	61824	186	100	26	80.874
6	3	CC	24	95	65	43	0.102101	16.281819	125488	86721	66587	169	100	26	81.458
7	3	DC	24	88	66	54	0.100759	17.890694	99232	88728	63927	226	100	30	82.193
7	3	CC	24	96	70	52	0.140765	19.174532	129536	116752	89273	227	100	30	85.597
8	3	DC	24	96	92	65	0.180833	31.979467	129536	123734	78234	286	100	33	81.965
8	3	CC	24	96	92	62	0.208931	31.135783	129536	123734	80883	309	100	33	81.965
9	3	DC	24	96	92	68	0.200362	31.013763	129536	124554	80045	327	100	34	81.636
9	3	CC	24	96	92	72	0.200646	33.228803	129536	124554	74823	317	100	34	81.636
10	3	DC	12	48	30	29	0.03424	2.214955	14080	8806	6539	100	100	22	72.36
10	3	CC	12	48	30	30	0.048393	1.730702	14080	8806	6834	101	100	22	72.36
11	3	DC	24	94	84	57	0.165777	25.655742	121440	113721	77477	273	100	32	81.177
11	3	CC	24	96	86	60	0.203099	29.09474	129536	121464	80119	298	100	32	82.311
12	3	DC	9	34	26	26	0.020949	1.053085	4480	3752	2159	85	100	23	77.132
12	3	CC	9	36	28	24	0.020461	1.361829	5376	4592	2783	80	100	23	79.225
13	3	DC	16	58	46	42	0.116496	3.836383	26424	20300	15332	141	100	25	79.665
13	3	CC	16	64	46	41	0.123371	4.273838	35840	26584	21267	132	100	25	83.637
14	3	DC	24	94	62	53	0.06888	17.548559	121440	105995	71792	242	100	29	81.171
14	3	CC	24	96	64	56	0.111914	21.396007	129536	113546	77409	249	100	29	81.9
15	3	DC	57	224	148	102	0.714391	233.972036	1774960	1620855	1324736	436	100	40	84.323
15	3	CC	57	228	150	102	1.142168	209.631418	1872640	1712160	1390542	439	100	40	85.187
16	3	DC	26	98	82	75	0.130751	33.411874	138744	133078	79949	333	100	33	80.222
16	3	CC	26	104	84	71	0.187705	37.278937	166400	158958	101139	317	100	33	82.371
18	3	DC	45	178	121	77	0.260206	115.771249	877888	775052	595987	361	100	35	81.582
18	3	CC	45	180	122	81	0.425881	137.06943	908160	799324	611741	369	100	35	82.053

Table A.12: FIT generation, coverage and CCA coverage for t=3 and cyclo=3

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	2	DC	1	14	14	0	0.014776	0.160793	NA	NA	NA	NA	NA	9	NA
1	2	CC	1	14	14	0	0.007528	0.000874	NA	NA	NA	NA	NA	9	NA
2	2	DC	1	6	4	0	0.001464	0.00061	NA	NA	NA	NA	NA	7	NA
2	2	CC	1	6	4	0	0.002236	0.000994	NA	NA	NA	NA	NA	7	NA
3	2	DC	3	18	12	0	0.009528	0.031237	108	100	0	9	100	8	76
3	2	CC	3	18	12	0	0.008107	0.047266	108	100	0	11	100	8	76
4	2	DC	1	6	6	0	0.003182	0.001337	NA	NA	NA	NA	NA	8	NA
4	2	CC	1	6	6	0	0.002553	0.00088	NA	NA	NA	NA	NA	8	NA
5	2	DC	1	4	4	0	0.002103	0.000852	NA	NA	NA	NA	NA	7	NA
5	2	CC	1	6	6	0	0.004103	0.000764	NA	NA	NA	NA	NA	7	NA
6	2	DC	7	42	28	0	0.024624	0.231665	756	648	0	24	100	9	71.759
6	2	CC	7	42	28	0	0.029092	0.326158	756	648	0	22	100	9	71.759
7	2	DC	4	22	20	0	0.01826	0.104629	180	180	0	18	100	10	74.444
7	2	CC	4	24	22	0	0.016774	0.126541	216	216	0	20	100	10	70.833
8	2	DC	8	44	40	0	0.041892	0.351091	844	838	0	35	100	10	65.871
8	2	CC	8	48	44	0	0.049212	0.404962	1008	1000	0	32	100	10	69.6
9	2	DC	8	44	40	0	0.047777	0.347717	844	838	0	35	100	10	69.212
9	2	CC	8	48	44	0	0.051343	0.408062	1008	1000	0	32	100	10	70.6
10	2	DC	2	12	12	0	0.011876	0.026073	36	36	0	12	100	8	69.444
10	2	CC	2	12	12	0	0.007456	0.017267	36	36	0	12	100	8	69.444
11	2	DC	4	22	18	0	0.010483	0.063395	180	168	0	14	100	9	66.667
11	2	CC	4	24	20	0	0.01165	0.0765	216	200	0	17	100	9	66
12	2	DC	2	12	12	0	0.006471	0.020854	36	36	0	12	100	9	66.667
12	2	CC	2	12	12	0	0.006641	0.017308	36	36	0	12	100	9	66.667
13	2	DC	18	98	69	0	0.248791	1.459935	4528	4274	0	54	100	12	73.07
13	2	CC	18	108	74	0	0.093658	1.854209	5508	5222	0	53	100	12	77.729
14	2	DC	8	46	42	0	0.054582	0.54905	924	912	0	36	100	11	70.943
14	2	CC	8	47	43	0	0.036847	0.375928	966	953	0	34	100	11	71.563
15	2	DC	15	82	78	0	0.369769	1.256935	3128	3109	0	59	100	14	73.4
15	2	CC	15	88	84	0	0.137236	1.394054	3612	3586	0	57	100	14	75.376
16	2	DC	20	104	68	0	0.296338	1.711527	5128	4516	0	47	100	12	71.922
16	2	CC	20	120	76	0	0.106105	2.409781	6840	6124	0	64	100	12	79.589
18	2	DC	19	102	89	0	0.151121	1.927132	4920	4696	0	64	100	14	66.269
18	2	CC	19	114	94	0	0.331417	2.38739	6156	5824	0	60	100	14	69.952

Table A.13: FIT generation, coverage and CCA coverage for t=2 and cyclo=4

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	3	DC	1	14	14	0	0.009887	0.001969	NA	NA	NA	NA	NA	18	NA
1	3	CC	1	14	14	0	0.012051	0.003995	NA	NA	NA	NA	NA	18	NA
2	3	DC	1	6	4	0	0.003031	0.002286	NA	NA	NA	NA	NA	12	NA
2	3	CC	1	6	4	0	0.002076	0.001924	NA	NA	NA	NA	NA	12	NA
3	3	DC	3	18	12	9	0.005502	0.075646	216	168	83	23	100	14	55.952
3	3	CC	3	18	12	11	0.005182	0.066439	216	168	84	23	100	14	55.952
4	3	DC	1	6	6	0	0.002268	0.001739	NA	NA	NA	NA	NA	13	NA
4	3	CC	1	6	6	0	0.001894	0.001311	NA	NA	NA	NA	NA	13	NA
5	3	DC	1	4	4	0	0.001489	0.001223	NA	NA	NA	NA	NA	12	NA
5	3	CC	1	6	6	0	0.002663	0.001754	NA	NA	NA	NA	NA	12	NA
6	3	DC	7	42	28	24	0.027233	1.332933	7560	4800	3085	84	100	18	73.333
6	3	CC	7	42	28	22	0.02048	1.242475	7560	4800	3043	74	100	18	73.333
7	3	DC	4	22	20	18	0.012316	0.31317	648	648	290	57	100	23	65.123
7	3	CC	4	24	22	20	0.021447	0.517218	864	864	396	62	100	23	62.847
8	3	DC	8	44	40	35	0.046655	3.106205	9216	9012	5063	146	100	26	60.775
8	3	CC	8	48	44	32	0.052359	3.59998	12096	11808	6707	134	100	26	63.152
9	3	DC	8	44	40	35	0.04017	2.911803	9216	9012	5063	146	100	26	57.645
9	3	CC	8	48	44	32	0.059546	3.592934	12096	11808	6707	134	100	26	59.57
10	3	DC	2	12	12	12	0.01045	0.057287	NA	NA	NA	NA	NA	14	NA
10	3	CC	2	12	12	12	0.004523	0.017396	NA	NA	NA	NA	NA	14	NA
11	3	DC	4	22	18	14	0.013852	0.20629	648	532	277	38	100	18	59.774
11	3	CC	4	24	20	17	0.01328	0.283908	864	696	369	48	100	18	57.04
12	3	DC	2	12	12	12	0.008265	0.031957	NA	NA	NA	12	NA	17	NA
12	3	CC	2	12	12	12	0.006432	0.021903	NA	NA	NA	12	NA	17	NA
13	3	DC	18	98	69	54	0.089018	21.322972	131056	110996	76949	235	100	30	69.545
13	3	CC	18	108	74	53	0.275574	33.102238	176256	150872	102474	230	100	30	74.856
14	3	DC	8	46	42	36	0.034864	3.128174	10584	10170	6288	137	100	25	64.11
14	3	CC	8	47	43	34	0.050262	3.1643	11340	10884	6650	134	100	25	65.371
15	3	DC	15	82	78	59	0.121148	19.851003	73616	72315	43693	267	100	34	66.658
15	3	CC	15	88	84	57	0.390257	19.624541	91728	89812	56116	251	100	34	69.221
16	3	DC	20	104	68	47	0.21181	22.447785	159392	114340	82203	206	100	29	73.023
16	3	CC	20	120	76	64	0.39766	38.546762	246240	181896	120494	353	100	29	79.528
18	3	DC	19	102	89	64	0.142029	31.440447	149168	130424	90387	319	100	33	55.765
18	3	CC	19	114	94	60	0.209556	48.022784	209304	178840	125400	320	100	33	61.27

Table A.14: FIT generation, coverage and CCA coverage for t=3 and cyclo=4

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	2	DC	1	14	14	0	0.01696	0.290191	NA	NA	NA	NA	NA	9	NA
1	2	CC	1	14	14	0	0.013466	0.001437	NA	NA	NA	NA	NA	9	NA
2	2	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	2	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	2	DC	2	32	30	0	0.018648	0.111308	252	250	0	15	100	11	32.8
3	2	CC	2	32	30	0	0.018643	0.096913	252	250	0	15	100	11	32.8
4	2	DC	1	6	6	0	0.002614	0.000567	NA	NA	NA	0	NA	8	NA
4	2	CC	1	8	8	0	0.005385	0.001363	NA	NA	NA	0	NA	8	NA
5	2	DC	1	6	6	0	0.003517	0.001128	NA	NA	NA	0	NA	7	NA
5	2	CC	1	8	8	0	0.005938	0.001288	NA	NA	NA	0	NA	7	NA
6	2	DC	3	20	18	0	0.012927	0.066273	132	132	0	19	100	9	65.909
6	2	CC	3	24	22	0	0.012641	0.065658	192	192	0	17	100	9	73.958
7	2	DC	3	26	18	0	0.008544	0.070347	220	220	0	19	100	11	44.091
7	2	CC	3	34	26	0	0.040815	0.388173	384	384	0	27	100	11	55.99
8	2	DC	6	39	35	0	0.034149	0.237007	620	610	0	31	100	11	64.754
8	2	CC	6	43	39	0	0.083034	0.558054	760	749	0	30	100	11	66.355
9	2	DC	6	39	35	0	0.049318	0.256821	620	602	0	29	100	12	70.598
9	2	CC	6	43	39	0	0.05616	0.536283	760	741	0	28	100	12	71.255
10	2	DC	7	52	40	0	0.065312	0.498622	1156	1072	0	34	100	11	69.683
10	2	CC	7	56	44	0	0.075511	0.703559	1344	1256	0	42	100	11	66.799
11	2	DC	4	25	21	0	0.015894	0.103029	222	222	0	20	100	10	64.865
11	2	CC	4	31	27	0	0.032467	0.151822	360	360	0	20	100	10	74.722
12	2	DC	1	4	4	0	0.003671	0.001042	NA	NA	NA	NA	NA	8	NA
12	2	CC	1	8	8	0	0.005893	0.001489	NA	NA	NA	NA	NA	8	NA
13	2	DC	14	96	76	0	0.305051	1.9294	4260	4011	0	47	100	13	72.077
13	2	CC	14	110	80	0	0.108173	1.792526	5616	5324	0	42	100	13	76.578
14	2	DC	5	38	34	0	0.103302	0.597667	576	566	0	20	100	12	75.442
14	2	CC	5	40	36	0	0.024695	0.229365	640	630	0	23	100	11	74.762
15	2	DC	5	34	32	0	0.063062	0.494213	456	456	0	27	100	11	44.298
15	2	CC	5	40	38	0	0.055236	0.606007	640	640	0	29	100	11	57.813
16	2	DC	6	41	35	0	0.06768	0.705429	686	686	0	35	100	12	55.685
16	2	CC	6	47	41	0	0.032766	0.325694	920	920	0	38	100	12	64.239
18	2	DC	10	76	56	0	0.172901	1.329428	2596	2390	0	46	100	13	64.77
18	2	CC	10	80	58	0	0.17308	1.597836	2880	2653	0	41	100	13	63.852

Table A.15: FIT generation, coverage and CCA coverage for t=2 and cyclo=5

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	3	DC	1	14	14	0	0.013557	0.002789	NA	NA	NA	NA	NA	18	NA
1	3	CC	1	14	14	0	0.004831	0.001008	NA	NA	NA	NA	NA	18	NA
2	3	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	3	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	3	DC	2	32	30	15	0.050898	0.299552	NA	NA	NA	15	NA	24	NA
3	3	CC	2	32	30	15	0.05278	0.293732	NA	NA	NA	15	NA	24	NA
4	3	DC	1	6	6	0	0.001705	0.000975	NA	NA	NA	0	NA	14	NA
4	3	CC	1	8	8	0	0.005775	0.002756	NA	NA	NA	0	NA	14	NA
5	3	DC	1	6	6	0	0.00272	0.001518	NA	NA	NA	0	NA	12	NA
5	3	CC	1	8	8	0	0.006016	0.002826	NA	NA	NA	0	NA	12	NA
6	3	DC	3	20	18	19	0.022038	0.395192	288	288	125	43	100	18	59.375
6	3	CC	3	24	22	17	0.018138	0.269479	512	512	244	35	100	18	72.656
7	3	DC	3	26	18	19	0.017122	0.284252	600	600	330	44	100	25	15.667
7	3	CC	3	34	26	27	0.014777	0.571463	1440	1440	516	63	100	25	44.167
8	3	DC	6	39	35	31	0.045658	1.723089	5120	4841	2771	122	100	28	56.166
8	3	CC	6	43	39	30	0.035179	2.133726	7040	6698	4008	123	100	28	56.345
9	3	DC	6	39	35	29	0.025826	1.630583	5120	4657	2630	122	100	27	58.621
9	3	CC	6	43	39	28	0.056543	2.272913	7040	6482	3820	124	100	27	58.068
10	3	DC	7	52	40	34	0.056849	3.510213	14240	11516	6949	141	100	26	56.026
10	3	CC	7	56	44	42	0.080126	4.925551	17920	14724	8394	145	100	26	50.455
11	3	DC	4	25	21	20	0.012738	0.363059	800	800	471	58	100	21	48.75
11	3	CC	4	31	27	20	0.018954	0.658186	1856	1856	1068	60	100	21	61.045
12	3	DC	1	4	4	0	0.002378	0.001964	NA	NA	NA	NA	NA	14	NA
12	3	CC	1	8	8	0	0.003468	0.001635	NA	NA	NA	NA	NA	14	NA
13	3	DC	14	96	76	47	0.107379	21.791886	115792	97652	69354	215	100	29	62.597
13	3	CC	14	110	80	42	0.150901	34.052197	176384	151204	110291	209	100	29	68.224
14	3	DC	5	38	34	20	0.044904	1.573022	4352	4130	2609	71	100	24	58.475
14	3	CC	5	40	36	23	0.049653	1.64051	5120	4886	2804	85	100	24	57.245
15	3	DC	5	34	32	27	0.043146	1.50185	3008	3008	1692	101	100	26	29.887
15	3	CC	5	40	38	29	0.029269	1.680582	5120	5120	3011	108	100	26	43.594
16	3	DC	6	41	35	35	0.025469	1.977546	5952	5952	3198	133	100	25	37.332
16	3	CC	6	47	41	38	0.062331	3.368847	9600	9600	4836	147	100	25	47.49
18	3	DC	10	76	56	46	0.06437	9.64671	52480	41290	29687	199	100	29	54.434
18	3	CC	10	80	58	41	0.173048	12.901529	61440	48434	34420	192	100	29	52.156

Table A.16: FIT generation, coverage and CCA coverage for t=3 and cyclo=5

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	2	DC	1	14	14	0	0.013373	0.286537	NA	NA	NA	NA	NA	9	NA
1	2	CC	1	14	14	0	0.010454	0.001118	NA	NA	NA	NA	NA	9	NA
2	2	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	2	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	2	DC	2	32	30	0	0.046829	0.253695	252	250	NA	15	100	11	32.8
3	2	CC	2	32	30	0	0.054296	0.274149	252	250	NA	15	100	11	32.8
4	2	DC	1	4	4	0	0.005514	0.001468	NA	NA	NA	NA	NA	11	NA
4	2	CC	1	18	18	0	0.024296	0.001396	NA	NA	NA	NA	NA	10	NA
5	2	DC	3	42	42	0	0.104404	0.455234	476	470	0	47	100	18	22.128
5	2	CC	3	52	50	0	0.058819	0.687529	780	770	0	67	100	18	27.532
6	2	DC	3	32	18	0	0.012025	0.083204	276	271	0	17	100	11	87.085
6	2	CC	3	44	30	0	0.022831	0.198898	624	618	0	16	100	11	93.851
7	2	DC	3	26	18	0	0.014905	0.103013	220	220	0	19	100	11	44.091
7	2	CC	3	34	26	0	0.014431	0.136199	384	384	0	27	100	11	55.99
8	2	DC	5	56	42	0	0.043389	0.600108	1224	1164	0	43	100	12	67.44
8	2	CC	5	64	50	0	0.082243	0.984571	1620	1578	0	35	100	12	70.342
9	2	DC	5	56	42	0	0.045472	0.609053	1224	1164	0	43	100	12	67.44
9	2	CC	5	64	50	0	0.054418	1.131343	1620	1578	0	35	100	12	70.342
10	2	DC	NA	NA	NA	NA	0.006237	0.000552	NA	NA	NA	NA	NA	15	NA
10	2	CC	NA	NA	NA	NA	0.000323	0.000516	NA	NA	NA	NA	NA	15	NA
11	2	DC	6	111	87	0	0.206889	5.046938	4936	4864	0	46	100	15	22.128
11	2	CC	6	199	149	0	0.969959	27.739003	15426	15135	0	74	100	15	27.532
12	2	DC	11	204	138	0	0.624382	15.047936	18236	17715	0	58	100	16	87.085
12	2	CC	11	228	160	0	0.586403	19.300373	23100	22560	0	56	100	16	93.851
13	2	DC	26	443	288	0	2.608911	172.220343	89690	87533	0	93	100	18	44.091
13	2	CC	26	531	330	0	3.081292	322.589759	128038	125285	0	90	100	18	55.99
14	2	DC	11	203	161	0	0.745895	23.427355	17628	17396	0	60	100	17	67.44
14	2	CC	11	269	203	0	1.137193	53.909003	30726	30486	0	62	100	17	70.342
15	2	DC	16	232	192	0	0.994105	19.868393	24952	24722	0	80	100	16	67.44
15	2	CC	16	244	200	0	1.056655	23.427723	27660	27372	0	65	100	16	70.342
16	2	DC	13	237	215	0	1.555992	58.81097	25046	24720	0	86	100	17	22.128
16	2	CC	13	353	289	0	2.565916	142.572514	55240	54499	0	106	100	17	27.532
18	2	DC	31	805	513	0	8.466515	1742.437742	300080	288450	0	121	100	20	87.085
18	2	CC	31	899	585	0	11.34612	3148.398846	370330	356176	0	120	100	20	93.851

Table A.17: FIT generation, coverage and CCA coverage for t=2 and cycl=6

sutid	t	criterion	nvo	Nvosett	Nunqsett	seedsize	initT	consT	Nttuples	Nttuple	Nvalidcovseed	Ncitts	coverage	ccats	cacov
1	3	DC	1	14	14	0	0.013405	0.002749	NA	NA	NA	NA	NA	18	NA
1	3	CC	1	14	14	0	0.01309	0.002721	NA	NA	NA	NA	NA	18	NA
2	3	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	3	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	3	DC	2	32	30	15	0.055621	0.316195	NA	NA	NA	15	NA	24	NA
3	3	CC	2	32	30	15	0.057247	0.296003	NA	NA	NA	15	NA	24	NA
4	3	DC	1	4	4	0	0.004379	0.002743	NA	NA	NA	NA	NA	24	NA
4	3	CC	1	18	18	0	0.019962	0.002877	NA	NA	NA	NA	NA	23	NA
5	3	DC	3	42	42	47	0.062278	1.265885	1560	1500	627	136	100	35	15.133
5	3	CC	3	52	50	67	0.077002	2.870443	3600	3480	1552	193	100	29	15.862
6	3	DC	3	32	18	17	0.009217	0.180081	432	422	213	32	100	24	68.246
6	3	CC	3	44	30	16	0.018443	0.679589	2880	2808	1326	28	100	24	81.588
7	3	DC	3	26	18	19	0.008601	0.211879	600	600	330	44	100	25	15.667
7	3	CC	3	34	26	27	0.023939	0.635379	1440	1440	516	63	100	25	44.167
8	3	DC	5	56	42	43	0.069088	5.47025	13120	11216	5209	177	100	24	45.096
8	3	CC	5	64	50	35	0.052726	8.633563	20304	18730	7865	143	100	24	45.115
9	3	DC	5	56	42	43	0.040802	5.29554	13120	11216	5209	177	100	24	45.096
9	3	CC	5	64	50	35	0.089344	10.375931	20304	18730	7865	143	100	24	45.115
10	3	DC	NA	NA	NA	NA	0.000737	0.002355	NA	NA	NA	NA	NA	35	NA
10	3	CC	NA	NA	NA	NA	0.000477	0.001498	NA	NA	NA	NA	NA	34	NA
11	3	DC	6	111	87	46	0.252412	117.50345	111444	106557	47663	240	100	36	41.463
11	3	CC	6	199	149	74	0.965347	752.154951	596408	567011	315325	373	100	35	33.487
12	3	DC	11	204	138	58	0.765163	525.041729	941536	859876	568524	258	100	42	56.875
12	3	CC	11	228	160	56	0.725548	637.088071	1372512	1276160	884223	266	100	41	55.465
13	3	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
13	3	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
14	3	DC	11	203	161	60	1.21393	1241.066761	870708	836764	583177	340	100	44	41.463
14	3	CC	11	269	203	62	1.352085	2070.673066	1965408	1916754	1329830	382	100	44	33.487
15	3	DC	16	232	192	80	1.006311	785.033231	1651424	1605498	1152435	379	100	42	56.875
15	3	CC	16	244	200	65	1.450688	999.242845	1934272	1873884	1290342	346	100	42	55.465
16	3	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
16	3	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
18	3	DC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
18	3	CC	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Table A.18: FIT generation, coverage and CCA coverage for t=3 and cyclo=6

BIBLIOGRAPHY

[1] B. S. Ahmed, K. Z. Zamli, and C. P. Lim. Application of particle swarm optimization to uniform and variable strength covering array construction. *Applied soft computing*, 12(4):1330–1347, 2012.

[2] S. Arlt, A. Podelski, and M. Wehrle. Reducing gui test suites via program slicing. 2014.

[3] A. Bansal. A comparative study of software testing techniques. *vol*, 3:579–584, 2014.

[4] A. Barrett and D. Dvorak. A combinatorial test suite generator for gray-box testing. In *Space Mission Challenges for Information Technology, 2009. SMC-IT 2009. Third IEEE International Conference on*, pages 387–393. IEEE, 2009.

[5] A. C. Barrett. A parametric testing environment for finding the operational envelopes of simulated guidance algorithms. In *26th Aerospace Testing Seminar, Manhattan Beach, California, March 28, 2011.* Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2011., 2011.

[6] A. Calvagna and A. Gargantini. Combining satisfiability solving and heuristics to constrained combinatorial interaction testing. In *Tests and Proofs*, pages 27–42. Springer, 2009.

[7] J. J. Chilenski and S. P. Miller. Applicability of modified condition/decision coverage to software testing. *Software Engineering Journal*, 9(5):193–200, 1994.

[8] M. B. Cohen, C. J. Colbourn, and A. C. Ling. Augmenting simulated annealing to build interaction test suites. In *Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on*, pages 394–405. IEEE, 2003.

[9] M. B. Cohen, M. B. Dwyer, and J. Shi. Constructing interaction test suites for highly-configurable systems in the presence of constraints: A greedy approach. *Software Engineering, IEEE Transactions on*, 34(5):633–650, 2008.

[10] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn, and J. S. Collofello. A variable strength interaction testing of components. In *Computer Software and Applications Conference, 2003. COMPSAC 2003. Proceedings. 27th Annual International*, pages 413–418. IEEE, 2003.

[11] M. Davies and G. Limes. Finding system-level failures in flight-critical software systems. In *Digital Avionics Systems Conference (DASC), 2011 IEEE/AIAA 30th*, pages 7C5–1. IEEE, 2011.

[12] M. D. Ernst, G. J. Badros, and D. Notkin. An empirical analysis of c preprocessor use. *Software Engineering, IEEE Transactions on*, 28(12):1146–1170, 2002.

[13] C. Fang, Z. Chen, and B. Xu. Comparing logic coverage criteria on test case prioritization. *Science China Information Sciences*, 55(12):2826–2840, 2012.

[14] J. D. Frank, B. J. Clement, J. M. Chachere, T. B. Smith, and K. J. Swanson. The challenge of configuring model-based space mission planners. In *7th International Workshop on Planning and Scheduling for Space (IWPSS 2011), Darmstaft, Germany, June 8-10, 2011.* Pasadena, CA: Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2011., 2011.

[15] T. Hadzic, R. M. Jensen, and H. R. Andersen. Calculating valid domains for bdd-based interactive configuration. *arXiv preprint arXiv:0704.1394*, 2007.

[16] A. Hartman. Software and hardware testing using combinatorial covering suites. In *Graph Theory, Combinatorics and Algorithms*, pages 237–266. Springer, 2005.

[17] K. J. Hayhurst, D. S. Veerhusen, J. J. Chilenski, and L. K. Rierson. *A practical tutorial on modified condition/decision coverage*. National Aeronautics and Space Administration, Langley Research Center, 2001.

[18] http://pdepend.org/documentation/software-metrics/cyclomatic complexity.html. Cyclometic complexity, June 2013.

[19] D. R. Kuhn and M. J. Reilly. An investigation of the applicability of design of experiments to software testing. In *Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE*, pages 91–95. IEEE, 2002.

[20] D. R. Kuhn, D. R. Wallace, and J. AM Gallo. Software fault interactions and implications for software testing. *Software Engineering, IEEE Transactions on*, 30(6):418–421, 2004.

[21] R. Kuhn, Y. Lei, and R. Kacker. Practical combinatorial testing: Beyond pairwise. *IT Professional*, 10(3):19–23, 2008.

[22] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence. Ipog: A general strategy for t-way software testing. In *Engineering of Computer-Based Systems, 2007. ECBS'07. 14th Annual IEEE International Conference and Workshops on the*, pages 549–556. IEEE, 2007.

[23] J. Liebig, S. Apel, C. Lengauer, C. KaÌĹstner, and M. Schulze. An analysis of the variability in forty preprocessor-based software product lines. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, volume 1, pages 105–114. IEEE, 2010.

[24] W. Linzhang, Y. Jiesong, Y. Xiaofeng, H. Jun, L. Xuandong, and Z. Guoliang. Generating test cases from uml activity diagram based on gray-box method. In *Software Engineering Conference, 2004. 11th Asia-Pacific*, pages 284–291. IEEE, 2004.

[25] T. J. McCabe. A complexity measure. *Software Engineering, IEEE Transactions on*, (4):308–320, 1976.

[26] F. Medeiros, M. Ribeiro, and R. Gheyi. Investigating preprocessor-based syntax errors. In *Proceedings of the 12th international conference on Generative programming: concepts & experiences*, pages 75–84. ACM, 2013.

[27] M. Mendonca, A. Wąsowski, and K. Czarnecki. Sat-based analysis of feature models is easy. In *Proceedings of the 13th International Software Product Line Conference*, pages 231–240. Carnegie Mellon University, 2009.

[28] C. Nie and H. Leung. A survey of combinatorial testing. *ACM Comput. Surv.*, 43(2):11:1–11:29, Feb. 2011.

[29] C. Nie and H. Leung. A survey of combinatorial testing. *ACM Computing Surveys (CSUR)*, 43(2):11, 2011.

[30] Y. Padioleau. Parsing c/c++ code without pre-processing. In *Compiler Construction*, pages 109–125. Springer, 2009.

[31] G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In *Software Maintenance, 1998. Proceedings., International Conference on*, pages 34–43. IEEE, 1998.

[32] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. *Software Engineering, IEEE Transactions on*, 27(10):929–948, 2001.

[33] P. J. Schroeder, P. Bolaki, and V. Gopu. Comparing the fault detection effectiveness of n-way and random test suites. In *Empirical Software Engineering, 2004. ISESE'04. Proceedings. 2004 International Symposium on*, pages 49–59. IEEE, 2004.

[34] D. Schuler and A. Zeller. Assessing oracle quality with checked coverage. In *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, pages 90–99. IEEE, 2011.

[35] T. Shiba, T. Tsuchiya, and T. Kikuno. Using artificial life techniques to generate test cases for combinatorial testing. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 72–77. IEEE, 2004.

[36] K.-C. Tai and Y. Lie. A test generation strategy for pairwise testing. *IEEE Transactions on Software Engineering*, 28(1):109–111, 2002.

[37] Z. Wang, B. Xu, and C. Nie. Greedy heuristic algorithms to generate variable strength combinatorial test suite. In *Quality Software, 2008. QSIC'08. The Eighth International Conference on*, pages 155–160. IEEE, 2008.

[38] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur. Effect of test set minimization on fault detection effectiveness. In *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pages 41–41. IEEE, 1995.

[39] C. Yilmaz. Test case-aware combinatorial interaction testing. *Software Engineering, IEEE Transactions on*, 39(5):684–706, 2013.

[40] C. Yilmaz, S. Fouche, M. Cohen, A. Porter, G. Demiroz, and U. Koc. Moving forward with combinatorial interaction testing. 2013.

[41] L. Yu, F. Duan, Y. Lei, R. N. Kacker, and D. R. Kuhn. Combinatorial test generation for software product lines using minimum invalid tuples. In *High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on*, pages 65–72. IEEE, 2014.

[42] L. Yu, Y. Lei, M. Nourozborazjany, R. N. Kacker, and D. R. Kuhn. An efficient algorithm for constraint handling in combinatorial test generation. In *Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference on*, pages 242–251. IEEE, 2013.

[43] Y. T. Yu and M. F. Lau. A comparison of mc/dc, {MUMCUT} and several other coverage criteria for logical decisions. *Journal of Systems and Software*, 79(5):577 – 590, 2006. Quality Software.

[44] X. Yuan, M. B. Cohen, and A. M. Memon. Gui interaction testing: Incorporating event context. *Software Engineering, IEEE Transactions on*, 37(4):559–574, 2011.