TEST CASE OPTIMIZATION IN CONFIGURATION TESTING USING RIPPER ALGORITHM

Dheepigaa V S[1]

Master in Engineering, Software Engineering, University College of Engineering (BIT campus), Trichirappalli, India

Abstract

Software systems are highly configurable. Although there are lots of advantages in improving the configuration, it is difficult to test unique errors hiding in configurations. To overcome this problem, combinatorial interaction testing (CIT) is used to selects strength and computes a covering array which includes all configuration option combinations. It poorly identifies the effective configuration space. So the cost required for testing get increased. In this work, techniques includes hierarchical clustering algorithm and ripper algorithm. It gives high strength interaction which it can be missed by CIT approach and it identifies effective configuration space. We evaluated and comparecoverage achieves by CIT and RIPPER classification with hierarchical clustering. Using this approach we validate loop as well as statement based configurations. Our results strongly suggest that Proto-interaction formed by RIPPER classificationwith hierarchical clusteringcan effectively covers sets of configurations than traditional CIT.

Keywords: *Configuration options, Hierarchical Clustering, RIPPER Algorithm*

--***--

1. INTRODUCTION

A software system is a system of inter-communicating components based on software forming part of a computer system (a combination of hardware and software). It "consists of a number of separate programs, configuration files, which are used to set up these programs, system, which describes the structure of the system, and user documentation, which explains how to use the system". While a computer program is a set of instructions (source, or object code) a software system has many more components such as specification, test results, end-user documentation, maintenance records, etc.To alleviate this problem, researchers have proposed combinatorial interaction testing (CIT), whichidentifies a small but systematic set of configurations underwhich to test. For example, with one CIT approach, developers choose an interaction strength t and compute a covering array, which is a set of configurations such that all possiblet-way combinations of option settings appear at least once.The assumption underlying CIT is that configuration setsconstructed in this way are small in size while providing good coverage of the program's behavior. Thus the approach cost-effectively increases the likelihood of finding faults. However, our prior work challenges this assumption in several ways. The minimum set of configuration which is required for achieving particular goal is referred as Effective Configuration space. It typically comprises only a tiny subset of the full configuration space, and that subset of configurations is not well approximated by t-way covering arrays. To test this hypothesis, generally symbolic execution to discover a subject program's interactions, which are conjunctions of option settings needed to achieve specific testing goals, given a particular test suite is needed.But the cost complexity get increased. To achieve this concept hierarchical clustering and Ripper classification is used. From this technique,the proto-interaction get extracted. This proto-interaction is given by the conjunction of Configuration Options with their corresponding values.

2. RELATED WORK

In this section discuss various configuration testing approaches. Elnatan Reisner[1] implements symbolic evaluation for identifying how the settings of run-time configuration options will affect the line, basic block, edge, and condition coverage for the subjects under a given test suite.James C.King[13] introduced EFFIGY which provides symbolic execution for program testing .David Leon [2] evaluates test case filtering involves selecting a manageable number of tests to use from a large, existing test suite that contains redundant tests or is too large to use in its entirety. D. Richard Kuhn [11]developed tools for generating all pairs or higher degree combinations of input values. Patrick Francis [4] The tree-based techniques which are presented for refining an initial classification of failures takes place .These techniques is based on dendograms usage. They are all rooted trees used for representing the solutions for the hierarchical cluster analysis. The second technique which is used called a classification tree. This classification tree is used to constructed and recognize the failed executions. Using these two techniques, the tree representation is used for guiding the refinement process. This is experimentally evaluated on several subject programs.William Dickinson[3] proposed filtering procedures based on the concept that clustering are more effective than simple random sampling for identifying failures.Andy Podgurski[12] proposed automated support for classifying the reported software failures in order to facilitate prioritizing them and analysis their causes. Supervised and unsupervised pattern classification and multivariate visualization techniques are used. Similarity and dissimilarity are calculated using Euclidian distance and Manhattan distance. Christopher Henard [13] proposed the method of bypassing the combinatorial explosion using similarity for Generating and Prioritizing T-Wise Test Configurations for Software Product Lines. In this t-wise combination was used to the production of configurations for testing was proposed. Current t-wise approaches will applicable for small values of t in SPL. The fine control of the configuration process gets failed. So that the automatic generation and prioritization the configurations product for a large SPLs are required. A search-based approach is proposed for which is capable for generating product configurations in larger system. Sandro Fouche, Myra B. Cohen, Adam Porter[14] A new approach that incrementally builds covering the array schedules was proposed. In this approach at beginning starts at a low strength, and then as resource allows the strength get increased gradually. In every step previously tested configurations are reused, thus work duplication will be managed. The incremental approach developers need to cover the specific covering array strength progressively. Stronger covering array schedules were used, few of the configuration dependencies failures can be found and classified is cheaply and quick in time possible. It will remove the risks of performing to overly strong test schedules.

3. PROPOSED WORK

In this work the configuration files of the software system is taken as the input, then the configuration options which is present in statement as well as loop get collected. The collected configuration options are get combined using combinatorial method and test suite will cover all the configuration options obtained. Then clustering and classification algorithm are used for extracting proto-interaction.
Clustering of the test suite will be done by Hierarchical clustering algorithm for grouping the similar configuration.Then finally RIPPER classification algorithm will be used to determine which class the configuration belongs .This algorithm will be used for discovering high coverage configuration.

4. PROPOSED FRAMEWORK

Fig1, represent the proposed framework will consists of

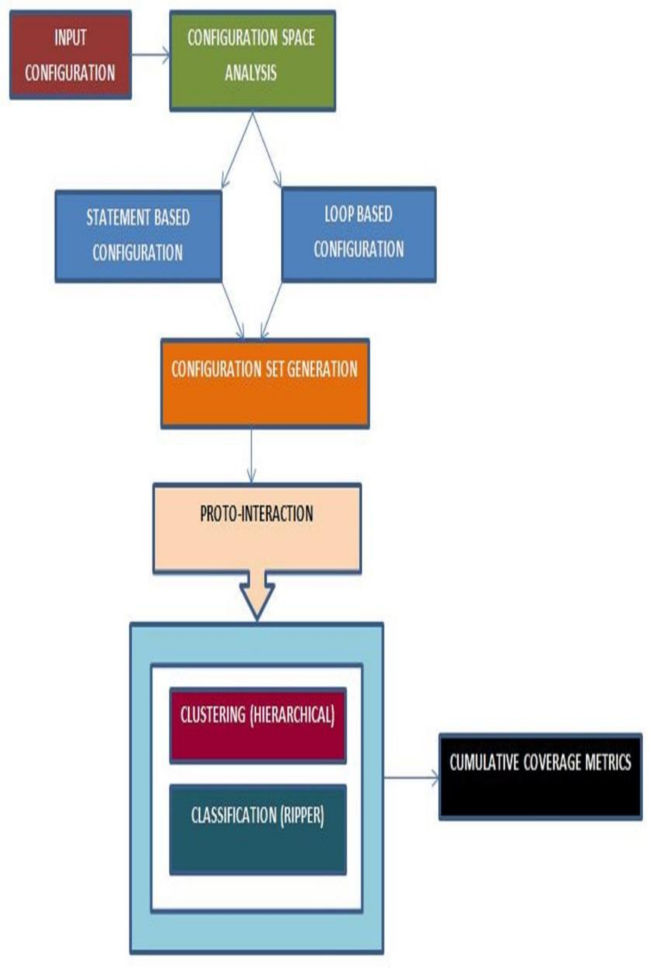

INPUT CONFIGURATION
CONFIGURATION SPACE ANALYSIS
STATEMENT BASED CONFIGURATION
LOOP BASED CONFIGURATION
CONFIGURATION SET GENERATION
PROTO-INTERACTION
CLUSTERING (HIERARCHICAL)
CLASSIFICATION (RIPPER)
CUMULATIVE COVERAGE METRICS

Fig 1: Proposed Framework

4.1 Configuration Acquisition

In this system software configuration file will be taken as the input file. The configuration options from the configuration file will get gathered.

4.2 Configuration Set Generation

The configuration options which are collected get combined using combinatorial method. Through this proto-interaction will be obtained.Proto-interaction is the combination of two or more configuration options. Let us consider the example of ngircd config.ngircd is Internet Relay chat daemon. ngircd is a free portable and light weight Internet relay Chat server for smaller private networks, developed under General public license. There are so many configuration options will be presented in this config file but for example we consider four configuration options which we call it as attributes

AllowRemoteOper,CloakUserToNick,ConnectIpv4,DNS.
AllowRemoteOper^ CloakUserToNick be the one of the interaction. ConnectIpv4^DNS be another interaction. The combination of two or more interaction is referred as proto-interaction that is AllowRemoteOper^CloakUserToNick^ConnectIpv4^DNS.

4.3 Hierarchical Clustering

In data mining, hierarchical clustering (also called hierarchical cluster analysis or HCA) is a method of cluster analysis which seeks to build a hierarchy of clusters.
In general, there are two types of hierarchical clustering methods

- Agglomerative hierarchical clustering: This bottom-up strategy starts by localizing each object in its own cluster and then coalesce there atomic cluster into larger cluster, until all of the object are in a single cluster or certain term are fulfilled.
- Divisive hierarchical clustering: This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. Algorithms that are illustration of the agglomerative hierarchical clustering admit:
  (1) highly in similarity
  (2) modular in similarity
  (3) less in similarity

Algorithm:

1. Collect the configuration options from the configuration file
2. Place all the configuration as a single Cluster
3. Based on the Dissimilarity between configurations split the cluster in to sub clusters.

$$\text{Dissim}(C_i.C_j)=\frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

4. Repeat the step2 until the condition satisfied

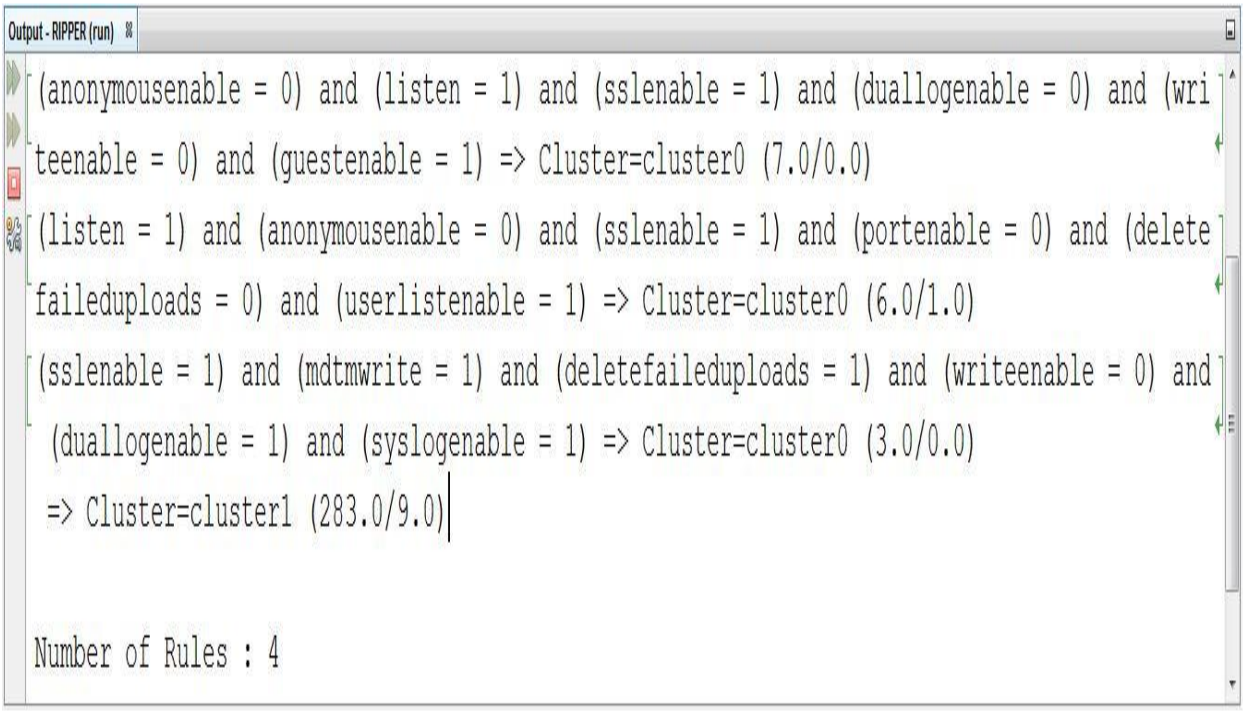
Thus this Hierarchical Divisive approach provide accurate clustering of the configuration. It provide classes to the configuration present in the system software.

4.4 Ripper Classification Algorithm:

Repeated Incremental Pruning to Produce Error Reduction (RIPPER) is rule learner, which was proposed by William W. Cohen.It is an optimized version of IREP. It is based in association rules with reduced error pruning (REP), a very common and effective technique found in decision tree algorithms. In REP for rules algorithms, the training data is split into a growing set and a pruning set. First, an initial rule set is formed that over the growing set, using some heuristic method. This overlarge rule set is then repeatedly simplified by applying one of a set of pruning operators typical pruning operators would be to delete any single condition or any single rule. At each stage of simplification, the pruning operator chosen is the one that yields the greatest reduction of error on the pruning set. Simplification ends when applying any pruning operator would increase error on the pruning set.

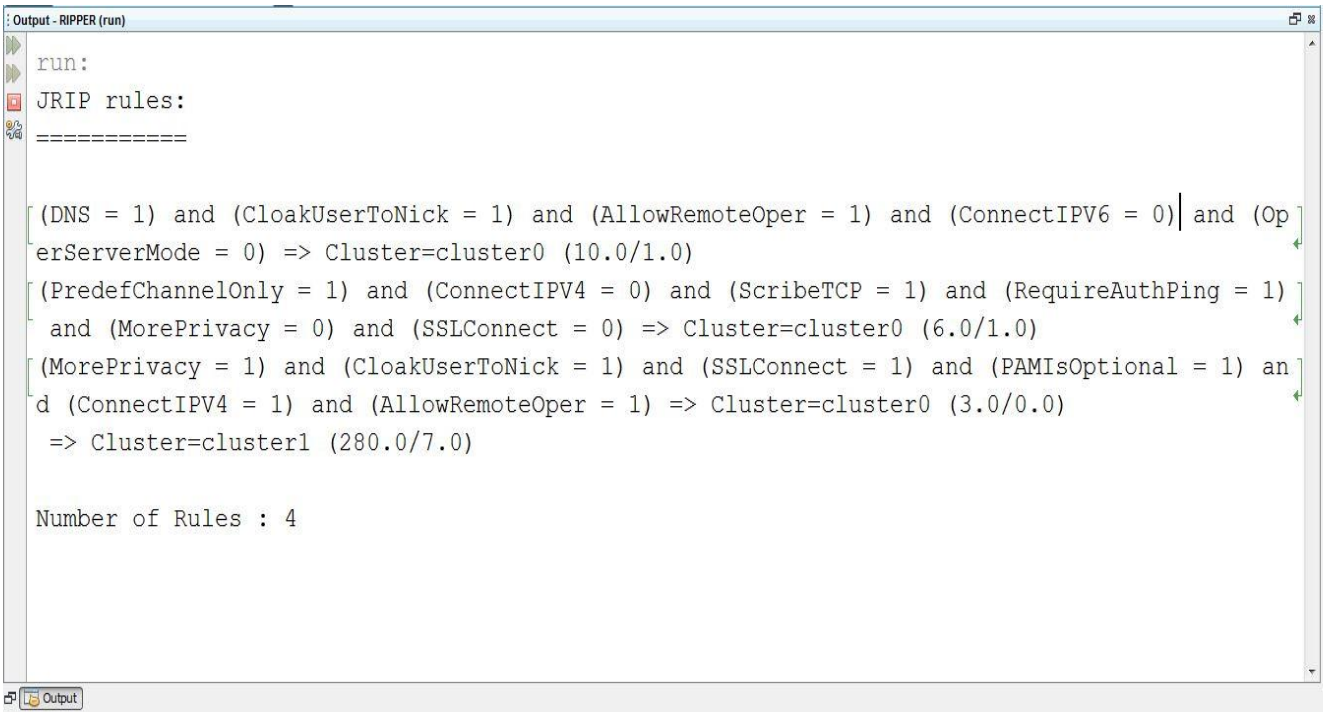
Algorithm as follows as:

- Start from empty test sets
- Add the test suits to check the configuration.
- Stop when rule no longer covers negative examples
- Prune the rule immediately using incremental reduced error pruning .Measure for pruning: v = (posconfig - negconfig) / (posconfig +negconfig).posconfig: number of positive examples covered by the rule in the validation set, negconfig: number of negative examples covered by the rule in the validation set Pruning method.



```
(anonymousenable = 0) and (listen = 1) and (sslenable = 1) and (duallogenable = 0) and (wri
teenable = 0) and (guestenable = 1) => Cluster=cluster0 (7.0/0.0)
(listen = 1) and (anonymousenable = 0) and (sslenable = 1) and (portenable = 0) and (delete
faileduploads = 0) and (userlistenable = 1) => Cluster=cluster0 (6.0/1.0)
(sslenable = 1) and (mdtmwrite = 1) and (deletefaileduploads = 1) and (writeenable = 0) and
 (duallogenable = 1) and (syslogenable = 1) => Cluster=cluster0 (3.0/0.0)
 => Cluster=cluster1 (283.0/9.0)


Number of Rules : 4
```

Fig.2. Proto-interaction generation from vsftpd, a widely used secure FTP daemon

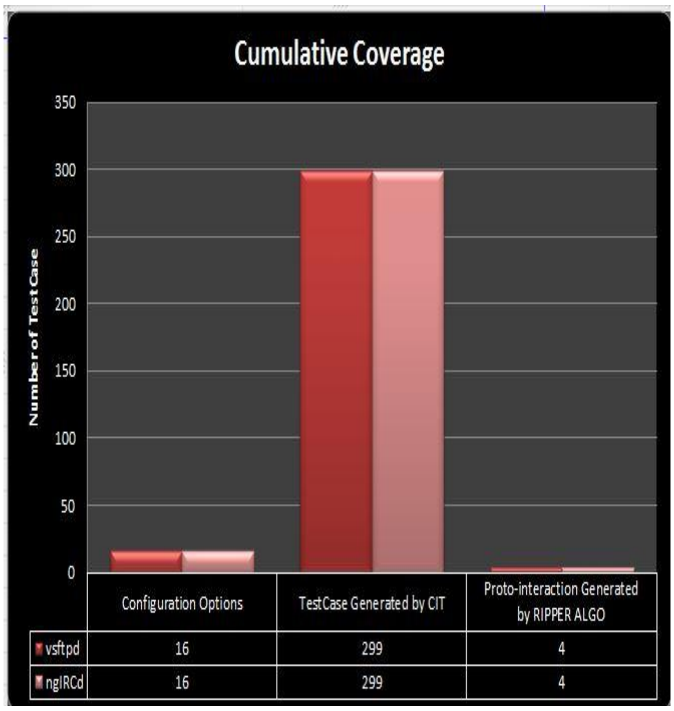


```
run:
JRIP rules:
===========

(DNS = 1) and (CloakUserToNick = 1) and (AllowRemoteOper = 1) and (ConnectIPV6 = 0) and (Op
erServerMode = 0) => Cluster=cluster0 (10.0/1.0)
(PredefChannelOnly = 1) and (ConnectIPV4 = 0) and (ScribeTCP = 1) and (RequireAuthPing = 1)
 and (MorePrivacy = 0) and (SSLConnect = 0) => Cluster=cluster0 (6.0/1.0)
(MorePrivacy = 1) and (CloakUserToNick = 1) and (SSLConnect = 1) and (PAMIsOptional = 1) an
d (ConnectIPV4 = 1) and (AllowRemoteOper = 1) => Cluster=cluster0 (3.0/0.0)
 => Cluster=cluster1 (280.0/7.0)

Number of Rules : 4
```

Fig.3. Proto-interaction generation from ngIRCd, next generation IRC daemon

5. DISCUSSION

We can evaluate the proposed approach using cumulative coverage metric. Our proposed approach provides high coverage in both statement and loop coverage. When number of configurations are high, cumulative coverage also high in proposed approach other than existing sampling and clustering approaches.In this 16 Configuration options are taken.Then test case generated using CIT and RIPPER algorithm are compared.



	Configuration Options	TestCase Generated by CIT	Proto-interaction Generated by RIPPER ALGO
🟥 vsftpd	16	299	4
🟥 ngIRCd	16	299	4

Fig.4. Cumulative Coverage

6. CONCLUSION

We conclude that proposed a new novel technique is to support configurable systems. It can be select small set configurations with test suite and achieve high coverage. This technique is based upon insights gained from our previous empirical studies in which we precisely quantified the relationships between software configuration and program execution behaviors. These insights led us to create a heuristic process that effectively searches out configurations in which high coverage is likely. To evaluate performance metrics using cumulative coverage metric. The first set of studies evaluated the basic approach and its parameters. The second set of studies compared this technique with t-way covering arrays existing techniques. The studies suggested that RIPPER Algorithm produced higher coverage than the other techniques while testing fewer configurations. The results strongly suggested that this new technique achieved higher coverage at lower cost than existing techniques.

ACKNOWLEDGMENTS

We would like to express our gratitude to Anna University Tiruchirappalli, India for supporting this research.

REFERENCES

[1]. E. Reisner, C. Song, K.-K. Ma, J.S. Foster, and A. Porter, “UsingSymbolic Evaluation to Understand Behavior in ConfigurableSoftware Systems,” Proc. ACM/IEEE 32nd Int’l Conf. Software Eng.(ICSE), pp. 445-454, 2010.
[2]. D. Leon and A. Podgurski, “A Comparison of Coverage-Basedand Distribution-Based Techniques for Filtering and PrioritizingTest Cases,” Proc. 14th Int’l Symp. Software Reliability Eng. (ISSRE),pp. 442-453, 2003.

[3]. W. Dickinson, D. Leon, and A. Podgurski, “Finding Failures byCluster Analysis of Execution Profiles,” Proc. 23rd Int’l Conf. SoftwareEng. (ICSE), pp. 339-348, 2001.
[4]. P. Francis, D. Leon, M. Minch, and A. Podgurski, “Tree-BasedMethods for Classifying Software Failures,” Proc. 15th Int’l Symp.Software Reliability Eng. (ISSRE), pp. 451-462, 2004.
[5]. K. Burr and W. Young, “Combinatorial Test Techniques: Table-Based Automation, Test Generation and Code Coverage,” Proc.Int’l Conf. Software (ICSE) Analysis & Rev., 1998.
[6]. C. Cadar, D. Dunbar, and D.R. Engler, “KLEE: Unassisted andAutomatic Generation of High-Coverage Tests for Complex SystemsPrograms,” Proc. Eighth USENIX Conf. Operating SystemsDesign and Implementation (OSDI), pp. 209-224, 2008.
[7]. D.M. Cohen, S.R. Dalal, M.L. Fredman, and G.C. Patton, “TheAETG System: An Approach to Testing Based on CombinatorialDesign,” IEEE Trans. Software Eng., vol. 23, no. 7, pp. 437-444, July1997.
[8]. M.B. Cohen, Combinatorial Interaction Testing Portal: Casa, 2009.
[9]. S.R. Dalal, A. Jain, N. Karunanithi, J.M. Leaton, C.M. Lott, G.C.Patton, and B.M. Horowitz, “Model-Based Testing in Practice,”Proc. 21st Int’l Conf. Software Eng. (ICSE), pp. 285-294, 1999.
[10]. W. Dickinson, D. Leon, and A. Podgurski, “Finding Failures byCluster Analysis of Execution Profiles,” Proc. 23rd Int’l Conf. SoftwareEng. (ICSE), pp. 339-348, 2001.
[11]. D. Kuhn and M. Reilly, “An Investigation of the Applicability of Design of Experiments to Software Testing,” Proc. NASA Goddard/IEEE Software Eng. Workshop, pp. 91-95, 2002.
[12]. A. Podgurski, D. Leon, P. Francis, W. Masri, M. Minch, J. Sun,and B. Wang, “Automated Support for Classifying Software Failure Reports,” Proc. 25th Int’l Conf. Software Eng. (ISSRE), pp. 465-475, 2003.
[13]. Christopher Henard, Mike Papadakis,Gilles Perrouin, Jacques Klein, Patrick Heymans,And Yves Le Traon,” Bypassing The Combinatorial Explosion: Using Similarity To Generate And Prioritize T-Wise Test Configurations For Software Product Lines,” IEEE Transactions On Software Engineering, Vol. 40, No. 7, July 2014.
[14]. S. Fouch_e, M.B. Cohen, and A. Porter, “Incremental Covering Array Failure Characterization in Large Configuration Spaces,”Proc. Int’l Symp. Software Testing and Analysis (ISSTA), pp. 177-188,2009.
[15]. C. Song, A. Porter, and J.S. Foster, “Itree: Efficiently Discovering High-Coverage Configurations Using Interaction Trees,” Proc.Int’l Conf. Software Eng. (ICSE), pp. 903-913, 2012.