This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

Constrained Interaction Testing: A Systematic Literature Study

Bestoun S. Ahmed*, Kamal Z. Zamli, Wasif Afzal, and Miroslav Bures

***Abstract*—Interaction testing can be used to effectively detect faults that are otherwise difficult to find by other testing techniques. However, in practice, the input configurations of software systems are subjected to constraints, especially in the case of highly configurable systems. Handling constraints effectively and efficiently in combinatorial interaction testing is a challenging problem. Nevertheless, researchers have attacked this challenge through different techniques, and much progress has been achieved in the past decade. Thus, it is useful to reflect on the current achievements and shortcomings and to identify potential areas of improvements. This paper presents the first comprehensive and systematic literature study to structure and categorize the research contributions for constrained interaction testing. Following the guidelines of conducting a literature study, the relevant data is extracted from a set of 103 research papers belonging to constrained interaction testing. The topics addressed in constrained interaction testing research are classified into four categories of constraint test generation, application, generation & application and model validation studies. The papers within each of these categories are extensively reviewed. Apart from answering several other research questions, this study also discusses the applications of constrained interaction testing in several domains such as software product lines, fault detection & characterization, test selection, security and GUI testing. The study ends with a discussion of limitations, challenges and future work in the area.**

***Index Terms*—Constrained interaction testing; constrained combinatorial testing; Software testing; Test generation tools; Test case design techniques.**

I. Introduction

SOFTWARE has become an innovative key for many applications and methods in science and engineering. Ensuring the quality and correctness of software is challenging because of the different configurations and input domains of each program. Ensuring the quality of software demands the exhaustive evaluation of all possible configurations and input interactions against their expected outputs. However, such an exhaustive testing effort is impractical because of time and resource limitations. Thus, different sampling techniques have been used to sample these input domains and configurations. The use of these sampling techniques for black box system testing is usually called as interaction testing, which can be used to detect faults that are otherwise undetectable effectively. Interaction testing has been given other alternative names such as combinatorial testing (CT), combinatorial interaction testing (CIT), and $t-way$, $t-wise$, or $n-wise$ testing (where t or n indicate the interaction strength). However, throughout this paper, the term "interaction testing" is used as a representative term as it is a popular name in existing software testing literature.

Interaction testing has been used successfully for testing different configurable software systems. Several review and survey papers exist on the topic that covers developed strategies and their applications (see e.g. [1]–[3]). Although useful, interaction testing has suffered from a limitation of efficiently handling constraints. The ability to handle constraints is a crucial aspect for the real-world applicability of interaction testing techniques since most of the real-world systems are subjected to constraints among input parameters or among particular system configurations. Hence, recent times have seen a shift in interaction testing research that concerns the handling of constraints and is typically called as constrained interaction testing [4]. Adding this feature opens up several new directions for research that promises to guide further development of interaction testing [5]. However, there exists no comprehensive and dedicated review paper in this direction.

Although many interaction testing strategies have been developed in the past, only a few of them can satisfy the constraints in the final generated test suite. Handling constraints add extra complexity in designing efficient interaction testing strategies. Hence, in the last decade, researchers have looked into different ways of supporting the generation of constrained interaction test suites. Besides, application of constrained interaction testing on various software systems and the associated empirical evaluations have started to surface. To this end, this paper provides a comprehensive systematic literature study to structure and categorize the available evidence for constrained interaction testing research during the last decade. The goal of the study is to identify the relevant papers, their results and the type of research such that one can discuss future research opportunities in the area. The study uses a systematic method to collect and analyze the related research published during the last decade. In doing so, methods and approaches for the generation as well as their applications are addressed.

The remainder of this paper is organized as follows: Section II presents the motivation and the overview of related work for this study. Section III describes the methodology of the literature study. Section IV presents the results and outcomes of the study. Section V discusses the threats to validity. Finally, Section VI concludes the work.

B. Ahmed, Software Testing Intelligent Lab (STILL), Department of Computer Science, Faculty of Electrical Engineering Czech Technical University, Karlovo nam. 13, 121 35 Praha 2, Czech Republic, (email: albeybes@fel.cvut.cz)

K. Zamli, Faculty of Computer Systems and Software Engineering, University Malaysia Pahang, Gambang, Malaysia, (email: kamalz@ump.edu.my)

W. Afzal, School of Innovation, Design and Engineering, Mälardalen University, Sweden, (email: wasif.afzal@mdh.se)

M. Bures, Software Testing Intelligent Lab (STILL), Department of Computer Science, Faculty of Electrical Engineering Czech Technical University, Karlovo nam. 13, 121 35 Praha 2, Czech Republic, (email: buresm3@fel.cvut.cz)

II. Motivation and related work

Multiple factors motivated the decision to carry out a literature study in this paper. First, no existing paper aggregates available research in *constrained* interaction testing (although several review papers exist on interaction testing in general). Second, constrained interaction testing is an upcoming research direction in interaction testing [6]; so it is interesting to investigate this topic. Third, a literature study paper is a community service that potentially saves significant time for interested researchers in getting to know about a research topic such as constrained interaction testing.

Nie and Lueng [2] conducted one of the first comprehensive reviews covering combinatorial testing and its applications. The study focuses on the basic concepts of combinatorial testing, the detailed methods of combinatorial test suite construction and the associated applications. The article also reviews constrained interaction testing methods. Kuliamin and Petukhov [7] also presented various methods of constructing combinatorial interaction test suites. On similar lines, Bestoun and Zamli [8] conducted a review study on the application of interaction testing. This paper is a useful complement to these existing review papers as it presents a more complete and recent review of the field. The previous review papers are not exhaustive in its coverage of constrained interaction testing. They are also not conducted as systematic literature studies.

Focusing on search-based test generation, Afzal et al. [1] and Ali et al. [9] have dedicated parts in their systematic literature reviews for interaction testing. More recently, Lopez-Herrejon *et al.* [3] published the first literature study on interaction testing for software product lines (SPL). Here, due to the presence of different constraints in SPL testing, the study has included and reviewed the constrained interaction testing. However, the study has only discussed constraints from the SPL application point-of-view.

This study considers the above-mentioned review papers as an excellent source of information since some of them include papers on constrained interaction testing. Our study further takes inspiration from other recently published systematic literature studies, e.g., [10], [11].

III. Method

This section illustrates the method that this literature study follows. This study follows the methodology recommendations given by [12]. The methodology has six stages. First is the definition of the research questions. Second is to undertake the search process, in which the search strategy is established, and the primary research papers are selected. The third stage is the selection and the quality assessment; this stage acts as a screening stage in which irrelevant papers are excluded based on the title, abstract, full-text reading and quality assessments. Data extraction is the fourth stage to extract data from the remaining papers. The fifth stage is the analysis and the data classification to classify the extracted information from the papers by tabulating and analyzing them. The last stage is the validity evaluation in which the threats to validity are evaluated and presented. For better illustration and organization of these steps, they were further classified into three main phases, as shown in Figure 1.

- **Phase 1. Searching:** The research questions that determine the focus of the study are defined in this phase. Based on these research questions, the search string is designed. The search string has undergone an experimental refinement process to identify and return only closely relevant papers.
- **Phase 2. Filtering:** Here, the relevant papers are selected, and their quality is assessed. Irrelevant papers are excluded based on the title, abstract, full-text reading and quality assessments.
- **Phase 3. Analysis:** The relevant data answering the research questions are extracted from the primary set of papers in this phase (in the case of this study 103 papers). The extracted data from the primary papers are classified and analyzed to visualize and understand the outcome. Here, Tables and Figures are used. Threats to validity are also analyzed and presented in this phase with the aim to disclose possible limitations of this study.

The following sections illustrate different stages covering the phases mentioned above that have been undertaken in detail. Input and output of each step are depicted in Figure 1. Some of the following subsections describe more steps joined. For example, defining research questions needs the research scope to be defined first. Hence, they are included in one subsection. In addition, the design, selection, and optimization of the search string are merged into one section.

A. Research Questions

As mentioned before, this study is a systematic literature study, and the goal is to structure and categorize the available evidence for constrained interaction testing research during the last decade. A number of research questions (RQs) were formulated to help achieve our goal:

- **RQ1:** What is the evolution in the number of published studies over the last decade in constrained interaction testing?
- **RQ2:** Which individuals, organizations, and countries are active in conducting constrained interaction testing research?
- **RQ3:** What topics/subjects have been addressed in the constrained interaction testing research and what is their distribution?
- **RQ4:** What are the existing strategies, tools, and techniques that support the generation of constrained interaction test suites?
- **RQ5:** What kinds of benchmarks (industrial or otherwise) are used to evaluate constrained interaction testing techniques and what is their provenance?
- **RQ6:** What are the applications of constrained interaction in software testing?
- **RQ7:** What are the current limitations and challenges in constrained interaction test generation?
- **RQ8:** What are the possible directions for future research?

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

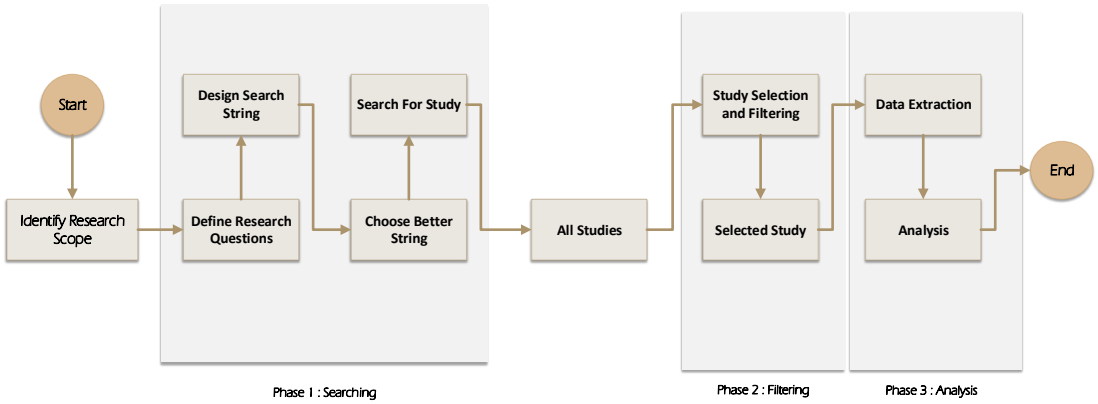

Fig. 1. The systematic literature detail steps

B. Search Strategy

Identifying the keywords for textual search is a challenging task. Kitchenham and Charters [12] established the PICO (Population, Intervention, Comparison, and Outcomes) criteria to identify the keywords formally. 'Population' refers to a role in software engineering, an application area or a discipline in the field, while the 'intervention' refers to software engineering tools, methodologies, procedures or strategies to address a specific issue. 'Comparison' identifies the different procedures or methods that have been used for comparison. 'Outcomes' deal with those keywords that are outcomes of the research and development, which are essential for practitioners such as improving performance or reliability.

Based on the research questions and the PICO guidelines, the keywords were categorized into three sets. The first set is related to the scoping the search for constrained interaction testing, i.e., "constrained interaction testing" or "constrained combinatorial interaction." To make the search broader, the second set of keywords is constructed to form terms and strings related to the generation of constrained test suites such as "strategy." The third set is related to the application of constrained interaction testing. These strings were combined to form one string but with different trials. To combine the search terms, Boolean AND is used, whereas Boolean OR is used to join alternate terms.

A preliminary string was constructed and then it took several trials to form the final search string (Set # 5 in Table I). These trials were needed mainly due to the close relationship between combinatorial interaction testing and constrained interaction testing. Since constrained interaction testing is the scope of this paper, a search string is required that returns only those combinatorial interaction strategies that support constraints. In doing so, these search strings were evaluated based on how much the returned results were related to the scope. In order to make sure that the search string is not missing out relevant papers, 20 "pilot" papers were selected, to make sure that different changes to the search (when experimented on IEEExplore and ScienceDirect indexing databases) are always able to find these 20 core/pilot set of studies. Finally, the string that is more related to the study area and can return these studies was selected.

Table I shows the result of five trials of different search strings. The first three sets of the search strings were excluded in the selection process since there were many irrelevant results returned by them, even though they returned the pilot papers also. The reason behind these results is the general terms in the strings that led to return many irrelevant papers. In addition, during the trials, it has been found that the term "constraint" is used in different ways depending on its situation in the sentences. To this end, three different terms were used, (constraint OR constrained OR constraints), which led to covering more papers. Additionally, it was also found that these synonyms were used with different terms of interaction testing. As a result, those terms were observed in the literature and used in different ways ("interaction testing" OR "combinatorial testing" OR "combinatorial interaction" OR "combinatorial test design" OR "covering array" OR "t-way testing"). To make sure that the scope was fully covered in the research questions, additional terms were added such as ("strategy," "technique," "method," "approach" and "tool").

The databases were selected based on the guidelines and suggestions provided by [13], [14]. Based on these guidelines, the following databases were selected:

- IEEE Xplore
- ScienceDirect
- ACM Digital Library
- Scopus
- SpringerLink

During searching, indexing, and sorting of a large number of references, different duplicate references appeared due to the slight differences in the reference indexing in the databases. To manage the references and to remove duplicates, the well-known reference management software EndNote X7 was used. For more accuracy, Mendeley v1.16 reference manager software was also used. As mentioned earlier, this is a literature study covering the last decade starting from 2005 as there has been increasing research trends from that time. It should be mentioned here that this study started in early 2016 and finished early 2017. The papers from 2017 are not included in this study. To figure out the number of published research

TABLE I
SEARCH STRINGS TRIES ON THE INDEXING DATA BASES

Keyword	Searching String	Returned Results	Missing Studies
Set # 1	(constrained) AND ((interaction testing) OR (combinatorial testing) OR (covering array) OR (t-way testing)) AND (testing) AND (strategy OR technique OR method OR approach)	242,439	0
Set # 2	(constrained OR constraint OR constraints) AND ((interaction testing) OR (combinatorial testing) OR (covering array) OR (t-way testing)) AND (testing) AND (strategy OR technique OR method OR approach)	511,049	0
Set # 3	(((constrained OR constraint OR constraints) AND (”interaction testing” or ”combinatorial testing” or ”combinatorial interaction testing” or ”covering array” or t-way testing) AND (strategy OR technique OR method OR approach)))	226,894	0
Set # 4	(constrained OR constraint OR constraints) AND (”interaction testing” OR ”combinatorial testing” OR ”covering array” OR ”t-way testing”) AND (testing) AND (strategy OR technique OR method OR approach)	1,102	3
Set # 5	(constrained OR constraint OR constraints) AND (”interaction testing” OR ”combinatorial testing” OR ”combinatorial interaction” OR ”combinatorial test design” OR ”covering array” OR ”t-way testing”) AND (testing) AND (strategy OR technique OR method OR approach OR tool OR application)	1,172	0

TABLE II
NUMBER OF PUBLISHED RESEARCH

Database	Search Results
IEEE Xplore	456
ScienceDirect	716
ACM Digital Library	43
Scopus	659
SpringerLink	794
Total	2,668

in this direction, Table II summarizes the number of research papers published in the mentioned period for each considered database.

C. Paper Selection Criteria and Quality Assurance

The papers were excluded or selected based on the title, abstract and full-text reading. The quality of the papers was also considered for the selection. To increase the reliability of selection, this process was conducted by the first author and reviewed again by other authors of the study. It should be mentioned that some papers can be selected or excluded based on the title and abstract. However, rest of them required full-text reading for deciding on their selection. For better understanding, the studies with the following criteria were selected:

- Studies for interaction testing with the support of constraints.
- Studies dealing with the applications of constrained interaction.
- Studies that are in the field of Software Engineering/Computer Science.
- Studies that are published online in the last decade (i.e., from 2005 when the first constraint-related paper got published).

It is also worth mentioning here the excluding criteria for the studies. Following the guidelines provided by [14], the published studies were excluded based on the following criteria:

- Studies dealing with interaction testing but without the support of constraints.
- Application of constrained interaction in fields other than Software Engineering/Computer Science.
- Studies not published in the English language.
- Studies without full text.
- Books and gray literature.
- Studies from non-peer reviewed sources.

Applying these criteria helped to capture better the number of papers that should be included in the study scope. As mentioned previously, many of these papers were duplicated, and they were removed finally. For example, some of those published papers in ScienceDirect were also indexed in Scopus database. In fact, Scopus acted as a valuable resource for double checking the results from other databases. Figure 2 shows these studies and the selection stages clearly.

As can be seen from Figure 2, to choose papers from a large set given by the selected databases, four filtering stages were applied. These stages have also been used in other literature studies [10], [15]. First, the related papers were identified in the selected databases (i.e., IEEE Xplore, ScienceDirect, ACM Digital Library, Scopus, SpringerLink). As mentioned previously, the outcome of this stage is 2,668 papers. It should be mentioned here that different papers were shared between these databases. Hence, “Filter 1” stage was performed in which all the duplicated titles, proceeding abstracts, and PowerPoint presentations were excluded. In the “Filter 2” stage, the papers were analyzed by reading the titles, abstracts, and if necessary the introduction sections of the papers. The selected papers were based on the exclusion and inclusion criteria provided earlier. In the “Filter 3” stage, the authors read the full-text of the chosen papers. Here, another set of papers was excluded due to multiple reasons. For example, many conference papers described an idea for research but do not include study results. In addition, the focus of some published papers was not entirely in software engineering and also not falling within the scope of the research questions. Finally, a snowballing stage was conducted by checking the references of the selected studies to not miss any relevant papers. Here, 14 more papers were added as an outcome of this stage. Hence, at the last stage, 103 papers were selected to answer the research

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

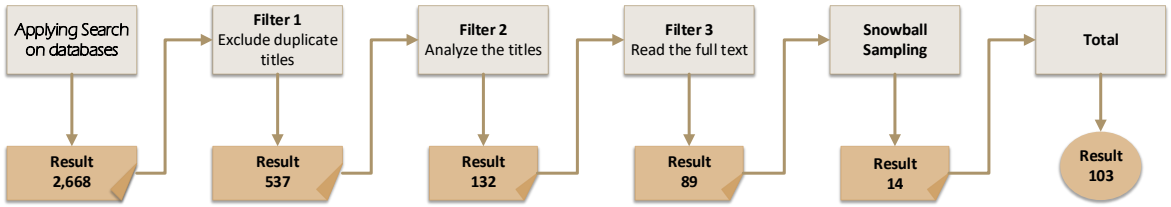

Fig. 2. Number of included papers in the selection and filtering process

TABLE III
DATA EXTRACTION TEMPLATE

Data item	Value
Study ID	Integer
Paper Title	Name of the paper
Author Name	Name of author(s)
Year of Publication	Calendar year
Venue	Name of publication venue
Country	Name of the country for each participated author
Area of research	Knowledge area of research
Research topic	Main topic or theme addressed by the study
Research problem	Research problem addressed by the study
Proposal	Proposed solution to the problem
Contribution	Main contribution of the paper
Challenges	Challenges addressed in the paper
Evaluation process	Which benchmark adopted for evaluation?
Case study	Which case study used?

questions by extracting information from them. Appendix A lists the studied papers. Appendix A shows the references for these papers along with their full names and the publication years.

D. Data Extraction and Analysis

This phase aims to extract data from the selected studies and analyze them to answer the research questions. A spreadsheet was created to retrieve the required data from these identified studies. The template developed by [14], [16] was followed and adopted to construct the sheet. More fields of data were also added to the table. Table III shows the template that was used. General information about the paper was recorded, including paper ID, publication title, publication year, authors' names and countries, venue, and area of research. More specific data was extracted by including the research approach for the study, evaluation process, case study, applied techniques and challenges addressed. The data extraction process helped to understand each paper's aim better and to get answers to the posed research questions. At the end of the process, the frequencies of papers were also calculated.

Each paper has a table with the information specified in Table III. To extract and analyze the information for all papers, a reliable method was followed. The information is extracted first by the first author and then double checked by the other authors separately. For better reliability, automatic text analyzer also used to verify the obtained information.

IV. RESULTS

This section is dedicated to answering the research questions in detail. Each research question from section III-A is addressed here individually. A short title is used for each section that is extracted from the main research questions in section III-A.

A. Frequency of Publications (RQ1)

The identified studies were analyzed over the last decade (2005–2016) to know the frequency and evolution of the number of publications. Figure 3 shows the results of this analysis process. As mentioned earlier, 103 publications were considered in which the average number of publications per year is of 10 papers. The average number is influenced strongly by the growth of publication numbers after 2009.

It should be mentioned here that the first model of constrained interaction for interaction testing purposes is proposed in a Ph.D. thesis in 2004 by Cohen [17]. However, this study did not appear in any database and was only published on the author (and university) website. In 2005, Hnich *et al.* [18] showed how to model handle constraints in Covering Array (CA); however, they did not explicitly address the constraints among the values of the input parameters, and they mention this as a problem to be solved in the future. In 2006, Bryce *et al.* [19] formalized the constrained interaction testing with CA mathematical object, while Hnich *et al.* [20] also defined and formalized the constraint models for CA in the same year. In the same time, Cohen *et al.* [21] tried to investigate the application of constrained interaction testing for SPL testing.

The interest in constrained interaction testing moderately increased between 2005–2007, whereas, a significant increase of research can be observed in 2008 and beyond. This increase in the publication number is an indication of the increasing interest in researching constrained interaction testing in the software engineering community. Another potential reason behind this increase is a shift of constrained interaction testing from theory to practice, with more and more papers investigating the application of this testing technique in different case studies.

Regarding the type of the publication venue, Figure 4 shows this information. Majority of publications (around 70%) are conference publications, about 22% are journal publications, and around 8% are workshop publications. It should be mentioned here that some conference publications are ultimately published as book chapters; however, their original venues, which are conferences, were considered.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

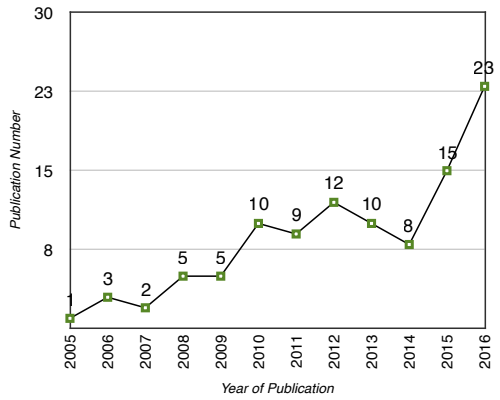

Fig. 3. Publication per year

Given these results, it is also important to know the popular peer-reviewed journal, conference and workshop venues for constrained interaction testing. Figure 5 shows those active peer-reviewed journals where relevant papers are published. Journals names are given in abbreviations of Thomson Reuters Science Citation Index[1] due to their long names. The full journal names corresponding to those abbreviations are given in Appendix B. Respectively, Figure 6 shows active conferences involved in constrained interaction testing research. The conference names are given in abbreviations, and the full names can be found in Appendix C.

Figures 5 and 6 gives a clear picture of targeted venues for publication by authors of the considered studies. Looking at the journal publications specifically, it is clear that "Software Quality," "Information and Software Technology," "IEEE Transactions on Software Engineering," and "Systems and Software," journals are the most active and top four journals in terms of publication target (more than 52% of the journal publications). Considering the conference publications, it be can noted that many papers have been published in individual conferences, however, "Software Testing, Verification and Validation (ICST)" and "Software Product Lines (SPLC)" are the most targeted venues for the authors (more than 36% of conference papers; 19 in ICST and 7 in SPLC). However, if we consider those conferences which published more than two papers, more than 68% of the papers were published in annual conferences. The remainder of the papers were published in 30 individual conferences (represented as others in Figure 6).

B. Active Individuals, Organizations and Countries (RQ2)

This research question aims to identify active researchers publishing studies about constrained interaction testing. A quick analysis reveals that many researchers are engaged in researching constrained interaction testing. In fact, it is clear that many authors were participating in a single research paper. Most active researchers were designated as those who author/co-author more than three research papers. Such researchers are producing more than 83% (86/103) of the publications. Figure 7 shows the ranking of these researchers based on them being authors or co-authors of the papers. From the ranking, it is clear that "Myra B. Cohen", "Angelo Gargantini" and "Andrea Calvagna" are top three researchers with 13, 10 and 6 published papers respectively. These three authors participated in almost more than 28% (29/103) of the total publications.

Table IV shows the ranking of the organizations and active research groups based on the published papers. The name of the group, participating researchers, reference to the published papers and the total number of papers is also presented in the table. The table complements the observations from Figure 7. In addition to the organizational ranking, the table also shows the collaboration among the authors. From the analysis of the table, it is clear that the research group from University of Nebraska - Lincoln (a collaboration between "Myra B. Cohen" and "Matthew B. Dwyer") is the most active research group in constrained interaction testing. In addition, that the group from Chinese Academy of Sciences (a collaboration between Jian Zhang and Feifei Ma) and the group from Technischen Universität Darmstadt (a collaboration between Malte Lochau and Sebastian Oster) are second active research groups since they are collaborating with other researchers in publishing 7 papers for each group. There are two active groups in the third rank. The group from Università di Bergamo (a collaboration between "Angelo Gargantini", and "Paolo Vavassori"), and University of Catania (active author "Andrea Calvagna") have participated in 6 published papers.

Another analysis can be drawn from the active countries in published papers. Figure 8 shows the most active countries in the publishing of constrained interaction testing papers. The figure shows the participation of each country in published papers based on the organizational affiliation of the authors. It is clear that USA, Germany, and Italy are top three countries in publications, with 28, 14, and 12 publications respectively. For example, 13 papers out of those published from the USA comes from a collaboration between "Myra B. Cohen" and "Matthew B. Dwyer". Germany is the second most active country in constrained interaction testing publications. This comes from the collaboration of three German universities with other groups. These organizations are, "Technischen Universität Darmstadt" (active researchers: Malte Lochau and Sebastian Oster), "Technischen Universität Braunschweig" (active researcher: Thomas Thüm) and the "University of Magdeburg" (active researchers: Gunter Saake and Mustafa Al-Hajjaji).

Important observations can be made from the output of this research question. Although the topic is important and promising, few researchers participate in constrained interaction testing publications. For instance, 31 active authors are participating in more than two research papers, and they are responsible for more than 83% (86/103) of the publications. In fact, many more authors are participating in combinatorial interaction testing without constraint's support.

C. Distribution of the Studies and Topics Addressed (RQ3)

Full-text of the considered papers was scanned carefully to answer this research question. Different topics and subjects

[1]https://apps.webofknowledge.com

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

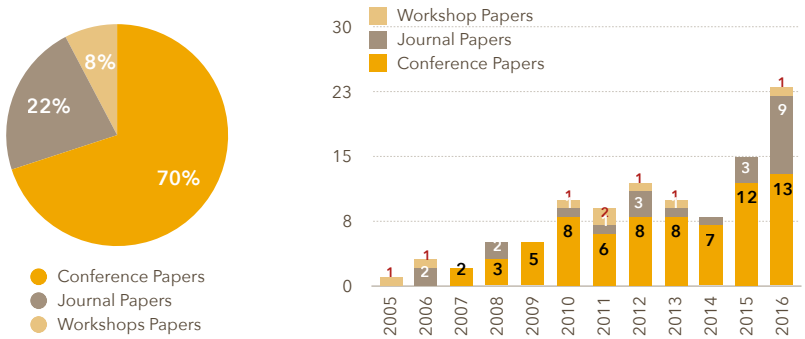

Fig. 4. Publication ratio and number categorized by source

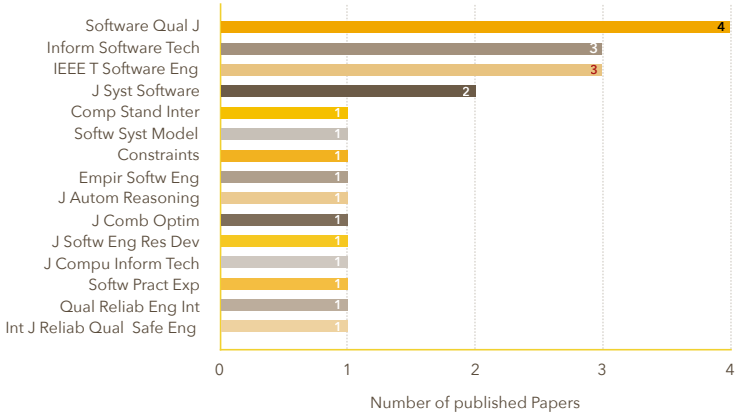

Fig. 5. Number of Published Papers vs. Journal Name

TABLE IV
RESEARCHERS AND ORGANIZATIONS INVOLVED IN CONSTRAINED INTERACTION TESTING RESEARCH

Organization	Author(s)	Papers Published	Total
University of Nebraska - Lincoln	Myra B. Cohen and Matthew B. Dwyer	[21]–[33]	13
Chinese Academy of Sciences	Jian Zhang and Feifei Ma	[34]–[40]	7
Technische Universität Darmstadt	Malte Lochau and Sebastian Oster	[41]–[47]	7
Università di Bergamo	Angelo Gargantini, and Paolo Vavassori	[48]–[53]	6
University of Catania	Andrea Calvagna	[30], [49], [51], [52], [54], [55]	6
University of Luxembourg	Yves Le Traon , Jacques Klein, Christopher Henard, and Mike Papadakis	[44], [56]–[59]	5
IRISA/INRIA	Sagar Sen and Benoit Baudry	[44], [57], [60]–[62]	5
University of Oslo and SINTEF ICT	Martin Fagereng Johansen	[63]–[66]	4
University Sains Malaysia / University Malaysia Pahang	Kamal Z. Zamli	[67]–[70]	4
SINTEF ICT	Øystein Haugen	[63]–[66]	4
University of Namur	Gilles Perrouin	[44], [57], [59], [71]	4
University College London	Justyna Petke and Mark Harman	[5], [29], [31]	3
Technische Universität Braunschweig	Thomas Thüm	[41], [43], [72]	3
University of Magdeburg	Gunter Saake and Mustafa Al-Hajjaji	[41], [43], [72]	3
Johannes Kepler University Linz	Roberto E. Lopez-Herrejon and Alexander Egyed	[73]–[75]	3
Arizona State University	Charles J. Colbourn	[19], [76], [77]	3
National Institute of Advanced Industrial Science and Technology-Japan	Cyrille Artho, and Takashi Kitamura	[78]–[80]	3
Simula Research Laboratory	Dusica Marijan	[60], [61], [81]	3

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

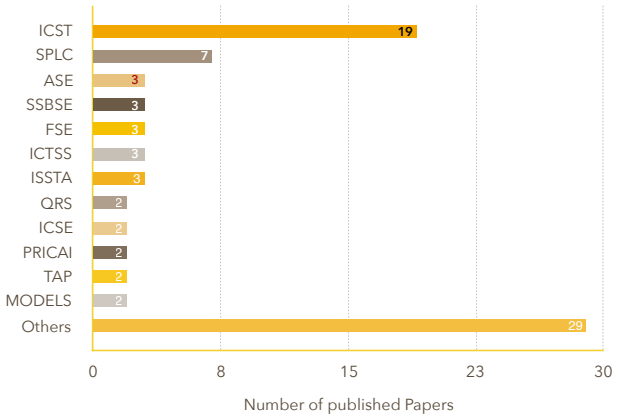

Fig. 6. Amount of Published Papers vs. Conference Venues

have been addressed. Although the topics are broad in range, this study distributed the topics into four main high-level categories. Within each category, many topics can be discussed. Table VI shows these four high-level categories and refers to the papers in each category. Based on a careful analysis, the studies are placed into the following categories:

- **Constraint Test Generation Studies:** Papers in this category discuss the generation strategies, methods, and approaches for constrained interaction testing. The papers are addressing the problem of generation by using different approaches including, exact methods, computational algorithms, meta-heuristic algorithms, and constraint solvers.
- **Application Studies:** Papers in this category consider only the application of constrained interaction testing for a specific domain of research (such as Graphical User Interface (GUI) testing).
- **Generation and Application Studies:** Papers in this category shows those studies that are considering a combination of generation and application. Here, the research papers are introducing a generation approach for a specific application(s).
- **Model Validation Studies:** Papers in this category either introduce models of constrained interaction testing, formalize the constrained interaction testing mathematically, or introduce models of problems that can be solved by constrained interaction testing.

Figure 9 shows the detail distribution of the papers according to the classified topics. The following sub-sections illustrate these categories and the papers related to them in detail.

1) Constraint Test Generation Studies: As can be seen from Table VI, 36 papers are proposing and evaluating different generation strategies of constrained interaction test suites. In addition, there are also 21 papers proposing customized generation strategies for specific applications. Hence, in total, there are 57 papers containing generation methods in some form. Here, it is essential to know the used methods for solving and dealing with constraints and also the generation strategies. In fact, the generation strategies will be addressed in RQ4, while this section addresses the constraint solving methods.

As can be seen from Figure 10, different methods have been used to solve the constraints during the generation of constrained interaction test suites. By analyzing those 57 papers, a classification for those constraint solving methods can be made. Out of those 57 papers, 43 papers (75.43%) used a constraint solver package. Those constraint solvers are, SAT, SMT, CSP, PBO and Clasp solvers. SAT solver has the highest usage rate 69.7% (30/43) [5], [18], [20], [22]–[26], [28], [30], [31], [34], [38], [40], [41], [44], [45], [56]–[59], [65], [72], [73], [78], [79], [85], [95], [100], [113] then SMT solver 13.9% (6/43) [33], [44], [51], [54]–[56], then CSP solver 9.3% (4/43) [5], [42], [60], [99], and then PBO solver 4.6% (2/43) [37], [39]. Recently, one paper uses Clasp solver also [36]. The use of SAT solver seems to attract researchers because of its performance, accuracy, and simplicity for solving the constraints. Henard *et al.* [56] validated this result recently by conducting a comparative study between SAT and SMT solvers in case of flattening the CIT into a Boolean model. The research found that the SAT solver can process the flattening models faster than the SMT solver.

Another reported way of solving the constraints in the literature is by excluding the constraints from the search process. For instance, 4 papers [19], [42], [87], [98] of the generation papers follow this method. The method removes those tuples which are related to the constraints, in other words, the meaningless tuples. In this way, there is no need for a constraint solver. However, this method just considers the exclusion of constraints rather than their inclusion. For real applications, in some cases, there could be inclusion constraints. For example, some input parameters might come exclusively with a specific set of parameters.

Binary decision diagram [86], [115], [118] and graph theory [27], [88], [89] methods are also used to deal with the constraints. Within the considered papers for generation approach, there are three papers for each of these methods. The binary decision diagram prevents the appearance of constraints in the final test suite by considering some form of cause-effect graph relationships. By considering this relationship, the generation method prevents the generation path from passing through those constraints. The graph theory method follows the well-known graph algorithms like graph coloring to generate test small test suites without violating the specified constraints.

Constraint programming [61], [62], [81] and linear programming [74] methods are also used rarely in the literature to deal with the constraints in combinatorial interaction testing. Constraint programming is mainly dedicated for solving hard combinatorial problems. Not much different from this definition, constraint programming is used within constrained interaction testing as an exact method to generate test suites without violating the constraints. Although it can be used with generation strategies in general, this method has been used only with SPL testing. Linear programming is also used with some similarities with this approach to solving the constraints but follows certain mathematical models. In general, linear programming is used with a mathematical model when the inputs have a linear relationship. Lopez-Herrejon *et al.* [74] followed this approach by using zero-one mathematical linear

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

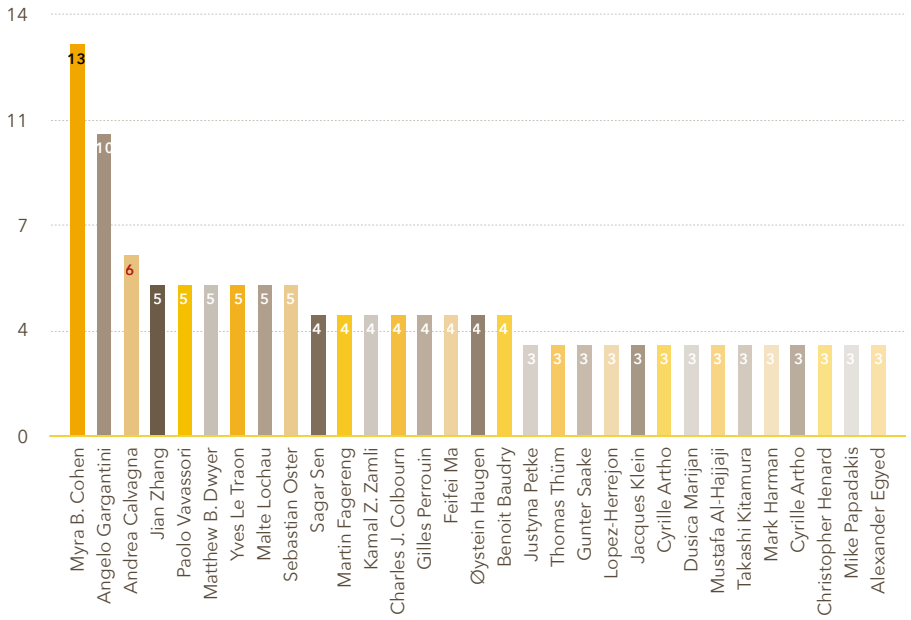

Fig. 7. Active Researchers

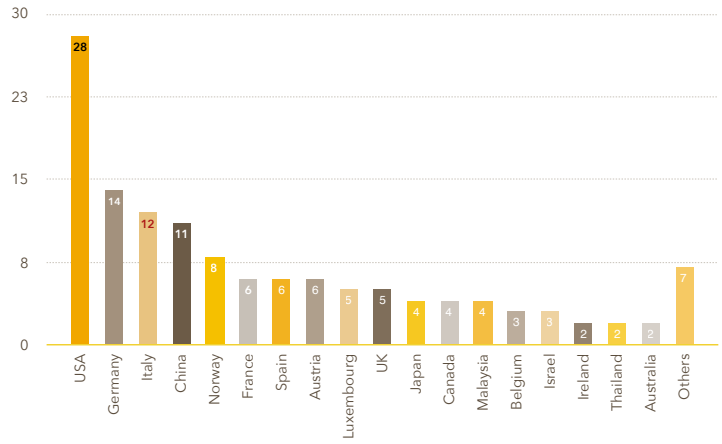

Fig. 8. Active Countries

TABLE VI
DISTRIBUTION OF PAPERS BY RESEARCH TYPE (HIGH LEVEL CLASSIFICATION)

Research Type	Number	Papers
Constraint Test Generation Studies	36	[18]–[20], [22]–[26], [34]–[38], [40], [48]–[51], [56], [67]–[69], [76]–[79], [82]–[91]
Application Studies	37	[27]–[30], [41], [42], [44], [45], [47], [52], [57], [58], [62]–[64], [71], [73], [80], [92]–[110]
Generation and Application Studies	21	[5], [31]–[33], [39], [43], [59]–[61], [65], [66], [70], [72], [74], [75], [81], [111]–[115]
Model Validation Studies	9	[21], [46], [54], [55], [116]–[120]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

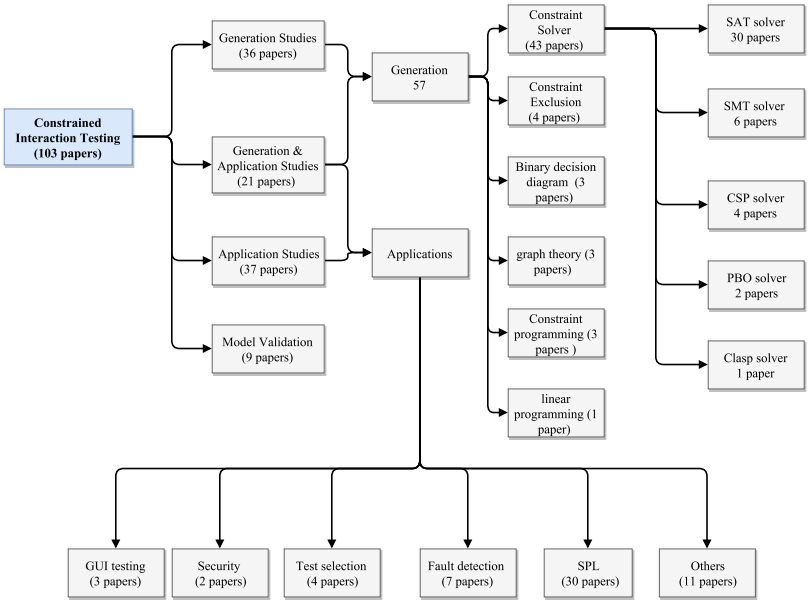

Fig. 9. Distribution of papers according to classification

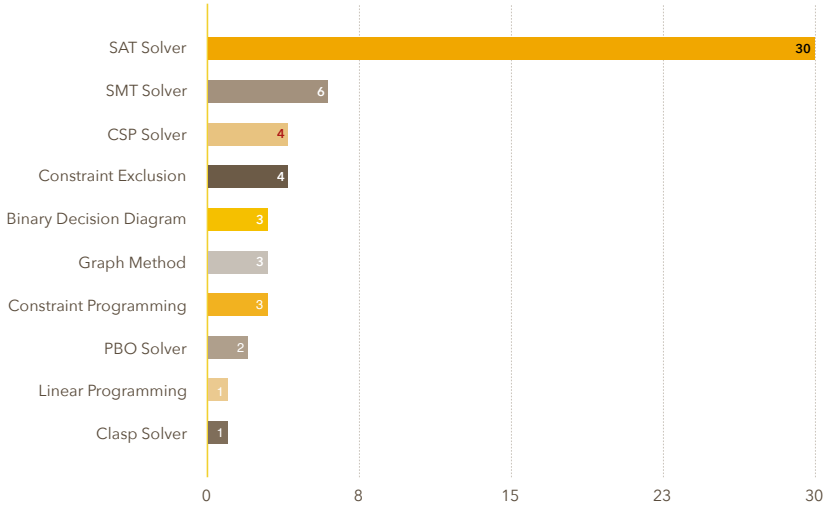

Fig. 10. Constraint Solving Methods

program with a multi-objective algorithm to solve the constraint problem for SPL feature models. In fact, this approach is difficult to follow for big and general constrained interaction testing strategies as it is a non-deterministic polynomial (NP) hard problem [4], [121].

2) Application Studies : From the Table VI, it is clear that 37 published papers are dealing with the application of constrained interaction testing without addressing the details of the generation process. The majority of these strategies used well-known and established tools and algorithms for generating constrained test suites. Those generation tools and algorithms are either adopted from other studies or used without much focus on them in the published studies due to the more significant focus on the application itself. For generating the test suites, there is an essential need to represent the system-under-test as a model to recognize the input parameters and constraints. These inputs are then used to generate test cases. Figure 12 shows those adopted methods in the application studies. Out of those 37 published papers in the application direction, eight papers are using SAT solver directly to generate the test cases and for solving the constraints. Eight papers used ACTS[2] tool for the generation since it is a well-established, efficient, and an excellent performance tool. Specifically, those published papers used the IPO-family algorithms inside the ACTS tool.

As can be seen from Figure 11, the applications are distributed into six main areas. Constrained interaction testing is frequently used within five of those areas actively. Those active areas (with the citations to them) are, SPL testing [27], [41], [42], [44], [45], [47], [52], [57], [58], [62]–[64], [71], [99], [101], [104], [109], [110], fault detection and characterization [28], [29], [80], [95], [105], test selection [96], [97], [107],

[2]http://csrc.nist.gov/groups/SNS/acts/index.html

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

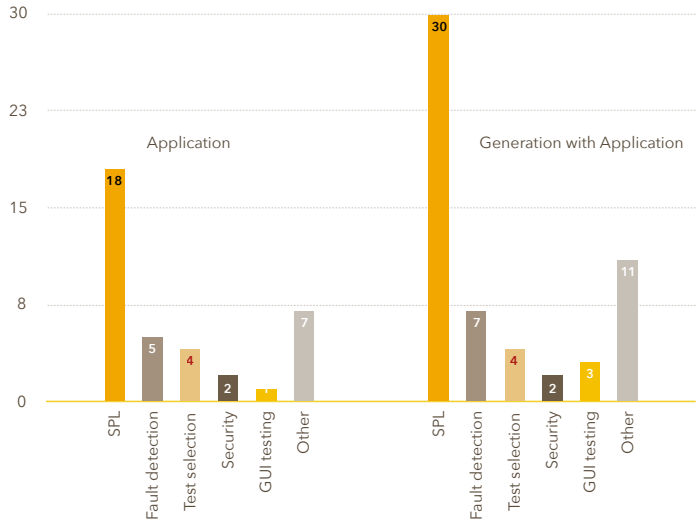

Fig. 11. Application Areas of constrained Interaction Testing

[108], security [93], [102], and GUI testing [114]. The left-hand side of the graph shows the results of those research papers which are presenting the application focus. It should be mentioned here that the papers are showing the results of the testing process without much detail about the generation process. In fact, most of the research papers use one or more generation tools either from their previous research or other research. For instance, 15 research papers show the use and the benefits of the constrained interaction testing for the SPL. In total, 29 published papers are focusing on the use of constrained interaction testing for SPL.

Fault detection and characterization is another active research direction here. Fault detection has been examined in the literature with normal interaction testing without constraints. Within constrained interaction testing, fault detection and characterization is used for different purposes. In fact, there are five research papers published in this direction without the generation details and two more papers with the test generation details. In total, there are seven published papers, but there are no frequent author names in them.

Another active research direction for application of constraint testing is the test selection direction. From Figure 11, it is clear that in total there are four published papers here. The test selection research originated from the normal combinatorial interaction testing. Again, there are no frequent names in these publications. This direction of research seems to catch less attraction by the researchers, but it still could be a productive future research direction.

Constrained interaction testing has been used for testing security issues. Security application is investigated first for normal combinatorial interaction testing for few studies (e.g., [122], [123]). In case of constraints, so far, two studies investigated its use in security. Bozic *et al.* [93] assessed the concept of constrained interaction for web security testing and Bouquet *et al.* [102] assessed it with model-based testing.

Combinatorial interaction testing has been used effectively to generate test cases for applications from the GUI point of view. Here, the constrained interaction testing is used to solve invalid test cases in the final generated test suite. In total, there are three published papers in this direction.

In addition to those aforementioned frequent areas, constrained interaction testing has been used in total within 11 different studies to solve real-life problems. More details on the above mentioned active applications and other areas are given while answering RQ6 in subsection IV-F.

3) Generation and Application Studies: Another category of research area within constrained interaction testing belongs to the "generation and application studies." As can be noted from Table VI, 24 published papers are dealing with the application of constrained interaction testing and also addressing the generation process. It is clear from the right-hand side of Figure 11, there are still six frequent main areas for research. The right-hand side of the figure extends the left side by adding more papers to the application categories.

More papers are dealing with the SPL application but with more details about the generation of test cases. Here, the papers considered the generation of test cases for the SPL as a special case of constrained interaction testing due to different constraints in the feature models used within SPL. In addition, researchers tried to integrate the generation algorithms with the testing tools. Some of those algorithms were depending mainly on constraint solving solutions. For example, Model-based Software Product Line Testing framework (MoSo-PoLiTe) [66] uses CSP solver for solving the constraints in the feature models of the SPLs. Other tools deployed well-known algorithms for constrained interaction testing to generate test cases for the SPLs. CITLAB tool [52] deployed IPO-family algorithms that are available in ACTS [91] tool to generate test cases for SPL. In fact, the publication in this direction aimed to increase the efficiency of the generation algorithms. Other researchers used special algorithms to reach this aim. For instance, Al-Hajjaji *et al.* [72] proposes an incremental approach for product sampling for pairwise interaction testing (called IncLing), which enables

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

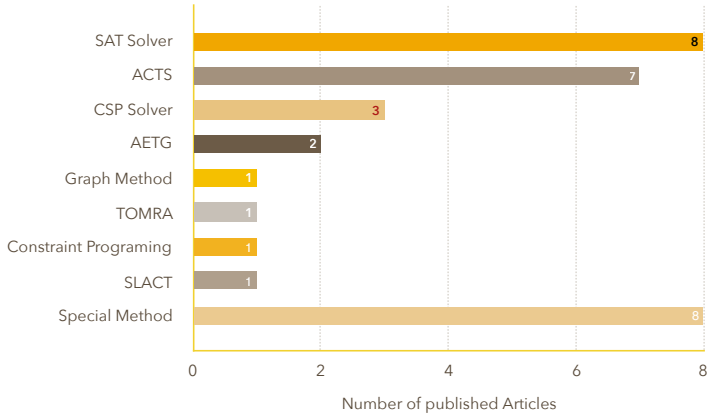

Fig. 12. Adopted Generation Methods for Applications

developers to generate samples on demand in a step-wise manner. Here, the performance of the generation algorithm is also improved by this incremental approach since there is no need to wait until the generation algorithm produces all the configurations. Lopez-Herrejon *et al.* [75] proposed Parallel Prioritized product line Genetic Solver (PPGS) algorithm, a parallel genetic algorithm for the generation of prioritized pairwise suites for SPLs. The study compared PPGS with the greedy algorithm prioritized-ICPL and in most cases, PPGS was found to be producing better results.

D. Marijan [61] has also introduced an algorithm to generate a minimal set of valid test cases for SPL that covers 2-wise feature interactions for a given time interval. Mainly, the algorithm is based on constraint solver and constraint programming. The study identifies the generation of a minimal configuration set for SPL from the feature model as a constraint optimization problem. The algorithm can capture the relationships among the features in the input feature model by a constraint model that identify these relationships as constraints for the solvers. The algorithm then tries to generate a set that covers the $2-wise$ interactions of the features and also satisfy the constraints.

4) Model Validation Studies: Model validation studies form another category of published research papers on constrained interaction testing. From Table VI, it is clear that seven published papers ([21], [46], [54], [55], [116]–[120]) address this issue for constrained interaction testing. The papers in this category deal with input models of an application under test. These models are then used to generate test cases accordingly. The rationale behind this research direction is the deployment obstacles to find the correct set of parameters and values with the restrictions (i.e., constraints) among them for real-life applications.

Generally, in combinatorial interaction testing, a model is composed of parameters and values for each of them. Constrained interaction testing adds some complex entity to the model validation process which is the constraint model among the parameters. The test generation strategies start from this model to construct the final test suite. Having a good model for the system under test will improve the quality of the produced test suites by the strategy and also it could help in early defect discovery. The model will be more useful when we want to validate the produced test suites and assess their quality.

Calvagna and Gargantinia [55] presented a new approach to construct constrained $2-wise$ test suites. More parts of the paper discuss the models of input domains and how the constraints should be presented in the strategy. The paper presented a model also to generate customized test suites by exclusion and inclusion of ad-hoc combinations of input parameters. The research also formalized constrained interaction testing as a logic problem. A model checker tool is presented in the paper to validate the model. In line with the approach of the paper, it presented a prototype tool for implementation. In another paper later, A. Calvagna and A. Gargantini [54] extended this approach to include more formalization of the model with more extensive evaluation process for the generation and validation process.

Segall *et al.* [118] then suggested a different approach to handle the complex relationship models between the parameters. The research suggested two different construction models. The first construction model is the type counter parameter, which is a special type of parameter that counts the number of specific values appearance for each input parameter in each test case. The second construction model is the type value property, which specifies the number of properties related to each parameter and values. The research showed that these two approaches will reduce the modeling complexity needed significantly. The research validated the two approaches on two real-life case studies. Although the approach is new in this direction, significant experimental results for validation could not be recognized.

Arcaini *et al.* [116] proposed a model validation approach for the constrained interaction testing, focusing more on the validation of the constraints rather than the interactions of parameters. The approach checks the consistency of the constraint among the input parameters, to be sure that they are not contradicted by each other, and hence the input parameters and their values are necessary for the inclusion in the test suite. This has been done by checking whether the produced

test suites are consistent with the provided requirements in the beginning for the input parameters, values, and constraints among them. The research also provides a technique to identify potential causes for any appeared error and how to fix it. The research suggested four sets of checkpoints that each constrained interaction test suite must have. Those checkpoints are, the consistency of the constraints, the usefulness of the parameters and values, the correctness of the final test suite in term of constraints violation, and finally the importance of each test case in term of coverage. These checkpoints were validated by the benchmarks available in the CITLAB tool [49] and the constraints were validated with the help of SMT solver.

Tzoref-Brill and Maoz [119] explore the evolution of constrained interaction test space modeling. The research showed that the Boolean semantics are inadequate for constrained interaction model evolution. The research extends the Boolean semantics to a lattice-based semantics to provide consistency of the interpretation for model changes. This has been done via Galois connection to establish the connection among the elements in an abstract domain. The research provides an extensive formulation of the models without showing any experimental results.

In contrast with the approaches mentioned above of general model validation, M. Spichkova and A. Zamansky [120] proposes a formal framework for model and validation framework of constrained interaction testing at multiple levels of abstraction between elements. The framework is human-centric in which it provides queries to testers for helping to analyze and assess the quality of specific test plans, models, and constraints to check if they are complete and valid. The study presented a formal analysis of the framework without an extensive experimental evaluation.

While the model validation frameworks and approaches are feasible for constrained interaction testing, S. Khalsa, Y. Labiche [117] introduced the base choice criteria to account for constraints. Specifically, the researchers discovered two extensions for the base choice to address complex constraints for complex systems. The criteria and models were evaluated on industrial and academic cases studies using different generated test suites by various testing tools. The evaluation was based on cost and effectiveness of the test suites in term of code coverage and fault detection.

D. Existing Generation Strategies, Tools and Techniques (RQ4)

For the answering of RQ4, this study is interested in knowing available generation strategies and tools. By answering this question, it would also be possible to understand the features and drawbacks of each strategy. The generation process and the efficient algorithms of the strategies are also essential to show here. It is clear from discussions mentioned above that each generation strategy of constrained interaction testing needs a mechanism to deal with the constraints. In SectionIV-C1, it is illustrated that to solve the constraints, the researchers usually either using a constraint solver through constraint programming or they exclude the constraints. They may also avoid the inclusion of constraints by following a particular algorithm. Having discussed those constraint issues, this Section is more about to know the algorithms used to derive the interactions for the final constrained interaction test suite.

There are many generation approaches for normal combinatorial interaction testing, with few of them supporting the inclusion of constraints. From the examined papers, in general, there are three approaches for the generation. The first approach is to use a computational algorithm to construct and optimize the test suites; the second approach is to use a meta-heuristic algorithm to optimize the test suite. With these two approaches, a constraint handling mechanism (explained in Section IV-C1) is used to handle the constraints in the final test suite. The third approach is to use a constraint solver to construct the final test suite directly without violating the constraints. The presented generation methods in the considered studies are falling under one of these approaches.

Following the first approach, Yu *et al.* [91] presented ACTS tool for generating different instances of combinatorial interaction test suites including the support for constraint handling. In fact, ACTS is a composition of many algorithms, mainly the IPO-family algorithms like IPOG [124], IPOG-D [125], and IPOG-F [126]. To handle the constraints, the ACTS tool relies on the Choco[3] solver. The tool can handle large configurations with a large set of constraints and large values of interaction strength, especially in the command line script. As mentioned previously, the tool has been used as a base for many applications in the literature.

Garvin *et al.* [25] followed the second approach and presented an improved implementation of the original simulated annealing-based (SA) constrained test suite generator [26] called CASA. The tool includes the original implementation of the SA generator algorithm with some improvements. The improvement is concentrated in the long run time for highly configurable systems with large constraint support. The research reformulated the search algorithm in the SA to efficiently incorporate the constraints. The tool uses SAT solver to handle the constraints. Through the evaluation process in the study, the tools perform well for the 35 benchmarks used. However, the tool becomes slow, and the performance degrades as the configuration benchmarks become large and it is not able to generate test cases for some complex configurations. In addition, it could be noted that the tool does not scale well when the interaction strength grows. As mentioned earlier, the tool has been used in other studies to investigate the application of constrained interaction test suites for real-world problems.

Remaining with the second approach, A. Kalaee and V. Rafe [115] presented an algorithm to generate constrained interaction test suites using the PSO algorithm. The constraints are handled by an ROBDD graph that helps to prevent them appearing during the construction. In the same way, Alsariera *et al.* [70] presented an algorithm to generate the test suites using the Bat-inspired algorithm for SPL. The study handled the constraints by excluding them from the final test suite.

[3]http://www.choco-solver.org/

Both studies need more extensive evaluation, and also their performance is not comparable with ACTS and CASA.

Following the third approach, Banbara *et al.* [113] presented an algorithm and a method to generate constrained interaction test suites using SAT solver directly. The study proposed modern CDCL SAT solvers with two encodings, one is order encoding while the other is a combination of order encoding. The use of these encodings helps to enhance the efficiency of generating better constrained interaction test suites in terms of size. The evaluation experiment showed this enhancement by generating better test suites sizes as compared with well-known results in the literature.

E. Used Benchmarks for Evaluation (RQ5)

By analyzing the considered papers for this study, it is clear that different studies are using standard benchmarks or what one can call "custom" benchmarks. As in the case of any benchmark, the aim of using it is to evaluate a particular approach within constrained interaction testing. In addition, it is also used to compare different approaches. Mainly, there are two types of benchmarks: benchmarks used to evaluate constrained interaction test generation efficiency in general and benchmarks used to evaluate the constrained interaction test generation efficiency for SPL.

Cohen *et al.* [26] used a set of experiments that have been used as a benchmark to evaluate a strategy for generating constrained interaction testing suites. Later in her study [23], Cohen has extended the set to include more real-world configurations generated from well-known software systems. These benchmarks are input models of Apache, GCC, Bugzilla, SPIN simulator and SPIN verifier. Apache HTTP Server[4] is an open source software system that works on different platforms to feed web services. GCC[5] is a multiple input language infrastructure for supporting many programming languages like C, C++, Java, and Fortran. Bugzilla[6] is a defect tracker from Mozilla that comes in open source to be used by software developers. SPIN model checker [127] is a software tool that is used as a case study for evaluation. In addition to these systems, 30 more system configurations were generated from the above systems to be used as benchmarks. These models are used as configuration benchmarks for 15 more published papers (see [22], [24]–[26], [36], [37], [54], [55], [68], [78], [79], [82], [85], [86], [90]). Other published papers rely on custom benchmarks, which are normally random configurations to simulate different scenarios of inputs.

Benchmarks for SPL test generation are commonly feature model files like XML files and contain different features and the constraints among them. These benchmarks may come from industry partners that use SPLs or may come from previously constructed sets for experimental use. In general, many papers use SPLOT[7], which is a website for SPL tools and repositories. In fact, 8 of the considered studies in this paper have been using the features models available on SPLOT for benchmarks. These papers are: [44], [45], [58], [59], [62], [75], [99], [101].

[4]httpd.apache.org/docs/2.2/mod/quickreference.html
[5]http://gcc.gnu.org/onlinedocs/gcc-4.1.1
[6]https://bugzilla.readthedocs.org/en/latest
[7]http://www.splot-research.org/

F. Applications of Constrained Interaction Testing (RQ6)

As can be seen from Figure 11, there are mainly five areas in which the constrained interaction testing has frequently been used. In addition, there are 11 different areas where the constrained interaction testing has shown significant results concerning the application. Hence, in total, there are 16 areas of the application so far for constrained interaction testing. The following subsections discuss these application areas with more details.

1) SPL : SPL engineering is getting more attention recently due to considerable industrial interest. The approach of SPL is to establish a platform for common products in a product line by identifying variability and commonality of features during the development of individual products [128]. The testing foundation in the SPL is to produce test assets that can be reused by the products during the process of product line development. The test case includes entities that form common parts related to a variety of possible products that must be realized by the domain engineer. The testing process of the SPL is getting more and more attention in the last decade. Many approaches have been published for testing SPL. Neto *et al.* [10] summarized different testing approaches for SPL in an extensive systematic literature study.

Due to huge features of modern products, it is very hard to generate a complete and feasible test suite and configurations. Here, combinatorial interaction testing can be a useful approach to construct an optimal and practical configuration for different potential products from a large set of features. However, in reality, there are different constraints among these features that tend to produce infeasible product configurations. Constrained interaction testing can be an excellent solution for this problem by generating an optimal configuration set without violating the constraints among the features.

To apply the constrained interaction testing for SPL, a set of activities need to be followed, as in the 29 papers identified earlier in Figure 11. These activities are the feature model adaptation including the constraints identification, interaction testing adaptation, and configuration set generation.

Feature model adaptation and constraint identification probably is the starting point in any approach. This step starts by taking a standard input file that can also be represented in a feature diagram. For instance, Perrouin *et al.* [57] formalized different types of potential constraints from the feature models. The study also proposed a toolset to generate the final test suite for SPL. Calvagna *et al.* [52] presented a more extensive and mature method to translate the feature models into combinatorial interaction models in a framework named CITLAB. The framework gives many advantages for a tool to generate constrained interaction testing such as editing facilities, seeding, and generation algorithms.

Interaction testing adaptation is an important stage to use the constrained interaction testing for SPL. In fact, the majority of the published papers are converting the feature models to the CA mathematical object using the combinatorial model

obtained by conversion. As previously mentioned, the CA consist of a set of $t-wise$ sub-array, where t indicates the strength of feature combination. The t is the number of features that we want to test in combination.

Because the number of possible features to be combined grows exponentially with the number of features designed for a particular SPL, there is a need for an efficient and practical strategy for $t-wise$ generation to get the most optimum set of combinations within an affordable testing cost. The adaptation takes care of four main entities for the CA which are input parameters, number of features, the constraints among the features and the combination strength. This stage is tied to the configuration set generation stage because the CA is an outcome of the stage. Here, different studies have proposed various algorithms for the generation. Majority of the studies used well-known algorithms for constrained interaction testing like ACTS, CASA, IPO-family algorithms and AETG. For example, Filho and Vergilio [101] developed a strategy that relies on the AETG for multi-objective test data generation for feature models' mutation testing. Following the same approach, Lamancha and Usaola, [104] proposed a strategy to generate $2-wise$ test products to cover all feature in the feature model using AETG. Johansen *et al.* [66] proposed a strategy for generating $t-wise$ test suites for SPLs from large feature models using CASA algorithms.

On the other hand, other studies proposed specialized tools for generating constrained interaction test suites for SPL. For example, PLEDGE is an editor and test generation tool developed by Henard and Papadakis [59] for SPL that rely on special algorithms based on SAT solver. PACOGEN [61], [81] is another tool for generating $2-wise$ test suites for SPL using constraint programming. Researchers are also using heuristic algorithms to generate optimum test suites. For example, Alsariera *et al.* [70] developed a strategy called SPLBA that relies on the bat-inspired algorithm to generate constrained interaction test suites. The strategy depends on constraint exclusion to resolve the constraints among the feature models. Jia *et al.* [31] developed a Hyperheuristic strategy to generate test suites for SPL. The strategy depends on SAT solver to resolve the constraints.

2) Fault detection and characterization: Here, the straightforward aim is to construct test cases for possible fault detection. The characterization process is used within fault detection for drawing a better understanding of the faults in the system and to enable fault avoidance in the future development. Another purpose is to design test cases for regression testing purposes. It could also be used for test case prioritization. Fault detection has been examined in the literature with normal interaction testing without constraints. For example, Yilmaz *et al.* [129] uses normal $t-wise$ approach with covering array to generate test cases to detect faults. The approach is integrated in the Skoll system [130] to allow parallel execution of test cases across a grid of computers. The results of the executed test cases were returned to a central server. The central server then classified the faults to cover and uncover faults. This classification of faults during the detection process helped the developers for providing descriptions of failing configurations to find the causes of the faults. In fact, this method can be used even with other test generation methods; however, Yilmaz found that interaction testing can produce better classification models.

3) Test selection: With the recent growing complexity of applications, it has appeared that constrained interaction testing is a more practical testing approach. The research in this direction considers the complex and large number of inputs and generating test cases from these inputs. Taking which values of these inputs and how to take them is a primary question here. Constrained interaction testing can sample these inputs systematically and take care of the constraints available between input values during the generation of the test cases. As previously mentioned, the test selection applications are more assessed within the normal combinatorial interaction testing. More recently, the focus of research has also shifted to consider the constraints among the input values due to the large and complex input variables and complex structures (e.g., JSON) in the real-world systems. For large scale modern software systems like commercial applications, selecting input for a test case is a difficult task due to the large input domain. In this case, exhaustive testing usually is impractical and not possible. One possible solution is to select one test design method (like boundary-value analysis or cause and effect) to select the inputs for test cases. However, these test design methods are not applicable for every situation. Constrained interaction testing could form another test design method that is suitable for sampling test cases based on their interactions among them while also considering the constraints among them.

Zhong *et al.* [108] generated constrained interaction test suites for large and complex software systems using comKorat, which is a combination of Korat and ACTS algorithms. Four systems were used to test the effectiveness of the generated test suites. These four systems are large-scale software systems developed at eBay and Yahoo!. The approach detected almost 59 bugs that were not detected by other approaches.

Nakornburi and T. Suwannasart [107] use a different approach to select entities necessary for test generation like the input parameters and values and then identify the constraints among them. The approach depends on the statistical profile of the software's user for selecting the entities. By selecting those entities, the study proposes a flow to generate an optimal $2-wise$ test suite for the software under test by eliminating the unrealistic combinations from the final test suite. However, the study did not mention the algorithm for the generation, and the presented approach is just for resolving constraints by excluding them. The study shows preliminary evaluation results for small size software under testing, and there is still need to show an extensive experiment.

For the case of large and complex software under test, C. Yilmaz [97] introduced a new combinatorial interaction object called test case-aware covering array, which is a refinement to the original covering array mathematical object used traditionally with constrained interaction testing but with different coverage criteria of the interactions. The study selects the input configurations of the software under test and considers a set of test cases for these algorithms to satisfy all test case-specific constraints. The study presented three algorithms,

two algorithms for test generation and minimization and the other algorithm is dedicated to minimize the number of test runs. The evaluation experiments used two highly configurable software systems (Apache and MySQL). The results of the evaluation showed that the test case-aware covering array is practically better than the traditional covering array due to the consideration of handling real-world and test case-specific constraints.

Finally, Kruse *et al. [96]* presented in a short paper an approach to handle constraints in the numerical constrained interaction test suites using the classification tree method. The study showed a prototype for the implementation without giving a complete description of the final solution.

4) Security: As previously mentioned, two studies were focusing on the application of constrained interactions for security testing. These studies are by Bozic *et al.* [93] and Bouquet *et al.* [102].

Bozic *et al.* [93] assessed the concept of constrained interaction for web security testing using the IPO-family algorithm. Specifically, the author evaluated IPOG and IPOG-F algorithms which are available within the ACTS tool. The algorithms generate test cases for exploring and detecting injection attacks which are remote exploits that can lead to security breaches. Specifically, the study focused on the cross-site scripting (XSS) vulnerabilities detection. Here, the inputs have been modeled using XSS attack vectors' modeling method to identify the number of inputs. The model is also used to identify the relations and constraints among the input parameter values. Four test sets are generated using ACTS tool, each two of these test sets were generated for IPOG and IPOG-F respectively. Then, these generated test suites are applied on a set of web applications that are used as cases studies. These applications are included in Open Web Application Security Project (OWASP)[8] and in the Exploit Database Project[9]. In general, the results of the study indicate that constrained interaction test suite can significantly reveal security leaks in web applications. The consideration of constraints during the modeling process of attack grammars will increase the number of test cases that can detect more vulnerabilities by examining more security breaches. The study also investigated that the quality of the test suites generated by IPOG-F is slightly better as compared to IPOG.

Bouquet *et al.* [102] discussed constrained interaction testing with model-based testing used for security testing. Although the paper is a general study without specific experiments, many important issues are discussed regarding the security test generation using model-based approach. In fact, the paper illustrated and discussed many cases studies in security where constrained interaction testing could be applied. Such case studies can very well form a stable base for experimental investigations.

5) GUI testing: Constrained interaction testing principles have been used in its simple form for GUI testing. Recently, the functional testing of GUIs has shifted to an advanced process by using graph theory and modeling concepts. The basic idea is to convert all events and positions on the GUI into nodes and edges and consider all possible sequences of events. This model has been used later as an input to the generation algorithms to generate test suites that simulate the event sequences. In fact, this method is initiated and solved by normal combinatorial interaction testing. However, with the normal combinatorial interaction testing, it has appeared that many of the generated test cases were not valid since the sequences were not visible for execution on the actual application's GUI. Huang *et al.* [32] recognized and illustrated this situation. Huang *et al.* investigated that there are constraints for the events and some events may not be available for execution. The study proposes a method to repair the generated test suites by avoiding those invalid combination of events and generated new feasible test cases, which in other words, solve the constraints. To evolve new test cases from the repaired test suites, a genetic algorithm is used to utilized event flow graph (EFG) for generating new test cases. The approach is tested using different synthetic programs that contain different GUIs with different constraints among the events. Although the use of EFG has been investigated in other research, it is not complete, and there is a need for manual verification in different cases.

Go *et al.* [106] introduced a different approach to apply $2-wise$ constrained interaction testing to GUI testing by analyzing system specifications. The analysis process of the specification will identify the class partitions of the input data to be tested. Here, they categorized the constraints into data, semantic rules, and GUI constraints. The data constraints are identified from the system inputs; semantic rules constraints are identified from system specification documents; while the GUI constraints can be collected from initial GUI requirements. The constraints have been solved by using conventional equivalent class partitioning without utilizing any constraint solvers. Here, the study did not mention the $2-wise$ test suite generation algorithm since the application used is not complex, and the test suite can be generated even by conventional methods. The study evaluated the approach on a real-world problem. The approach showed significant results concerning fault detection as compared to the conventional random testing approach.

Finally, Bauersfeld *et al.* [114], tried to complement and build the approach started by Huang *et al.* [32] for identifying and generating test sequences for applications with GUI. The study treated the test sequence generation of GUIs as an optimization problem and employed ant colony optimization (ACO) algorithm to solve it optimally. The study tries to avoid the limitations of EFG use by using a new metric called MCT (Maximum Call Tree) and avoids the inclusion of those invalid interactions from the beginning thus preventing repairing process of the test suites later. The approach is tested using an implemented Java SWT environment for application testing. A graphical editor is used to test the effectiveness of the approach. The study showed that the approach can generate better sequences as compared with the random approach. The study claimed that the approach is better than the use of EFG; however, there is no evidence for this claim.

[8]https://www.owasp.org/index.php/OWASP
[9]https://www.exploit-db.com/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2017.2771562, IEEE Access

6) Other applications: In addition to the applications mentioned above of constrained interaction testing, other applications have been referred to in one published paper. As in the case of above applications, the test cases were generated either using well-known tools or by developing specific algorithms for the generation. In fact, the application domains are broad in range.

For instance, Sherwood [92] assessed the application of constrained interaction testing for embedded functions and anticipated benefits of programming languages. Here, PHP language is used for its flexibility and prevalence. An individual model is employed in the study to identify and validate the constraints and generate test cases. In the same way, Salecker *et al.* [112] applied a similar approach for grammar-based test selection and generation.

Li *et al.* [94] applied the constrained interaction testing to big data applications. Here, two real-world big data ETL applications are used. The ETL (Extract, Transform, and Load) applications are common type applications in big data employed by the developers to report and analyze data by writing scripts in Hive, SQL, or Pig. To generate the test cases for the SUT from the input domain models, the study used ACTS tool. The study proved the effectiveness of the constrained interaction testing by using a minimized sizes of the test suites but detecting all the faults found in the original data source. Fischer *et al.* [73] performed an empirical case study to explore the application of constrained interaction testing. Seven Java and C programs are used as subjects of the case study in which they composed over nine million lines of codes in total. SAT solver is used to generate and resolve the constraints in the test suites.

Palacios *et al.* [98] addressed the use of constrained interaction testing for testing Service Level Agreements (SLAs). In this study, the classical combinatorial interaction testing has been combined with the Classification Tree Method to generate test suites and resolve input constraints. For this generation process, the study presented an automated tool called SLACT (SLA Combinatorial Testing) that has been applied successfully to an eHealth case study.

Arrieta *et al.* [100] used constrained interaction testing to generate configurations for cyber-physical systems (CPSs). CPSs are systems to integrate physical processes with digital technologies. In the study, simulation-based test cases were generated to test different configurations of CPSs. To resolve the constraints, SAT solver is used with other approaches for the generation to form a tool called ASTERYSCO. The tools are used successfully with a cases study for configurable Unmanned Aerial Vehicle.

Calvagna *et al.* [30] assessed the application of constrained interaction testing in the context of random and combinatorial effectiveness. The study uses this approach to know the differences between random and combinatorial concerning effectiveness for conformance testing. Here, the conformance testing used to verify components of Java Virtual Machine (JVM). To generate test cases model checking is applied to the considered specification.

Zhong *et al.* [108] applied constrained interaction testing on four large-scale software applications. The software systems were developed by eBay and Yahoo companies. To generate the test cases, the study presented a newly developed tool called comKorat which is an integration between Korat and ACTS generation tools. By applying the test suites generated by comKorat, more faults were detected in the SUT. Specifically, 59 new faults were detected while other test suites did not detect them.

In another study, Grieskamp *et al.* [33] investigated and proved the application of constrained interaction testing for path coverage of software systems. To generate the test suites, the study used SMT solver. The solver is also sued to resolve the constants among the paths in the software.

Finally, Kalaee, and Rafe [115] used the graph theory and Particle Swarm Optimization (PSO) to generate constrained interaction test suites. The developed generation algorithm is applied to a case study for a boiler system.

G. Current Limitations and Challenges (RQ7)

Fundamentally, the limitations and challenges in the application of constrained interaction test generation in practice related to four main factors: the parameters (P), the values (v), the interaction strength (t) as well as the constraints (C).

In line with the advancement of new technologies such as the Internet of Things, big data analytic as well as cloud computing, the intertwined dependencies and constraints between components and their sub-systems can be massive. To put this issue into perspective, Microsoft Windows source code was merely around 4.5 million LOC in 1993. In 2003, after ten years, Microsoft Windows source code increased to over 50 million LOC [131], a tremendous increase. Here, the growth of parameters (P) and the values (v) and the constraints (C) can be problematic (although such effect many be controllable as far as (t) is concerned as empirical evidence suggest that most faults are detectable at $t = 6$) [132]–[134]).

Two potential stumbling blocks can be attributed to the above issues. Firstly, as the parameters (P), values (v) and constraints (C) increase, the coverage of tuples for consideration also increases tremendously. Putting constraints (C) aside for simplicity, consider a scenario when $P = 1000$, $v = 5$ and $t = 2$. Here, the search space for exhaustive testing is 5^{1000} and the search tuple is equal to 1.

$$\binom{p}{t}v^t = \frac{p!}{t!(p-t)}v^t = \frac{1000!}{2!(1000-2)}5^2 \tag{1}$$

Similarly, when v is large (even with small P and t), the search space can also be large. Consider another scenario when $v = 1000$, $P = 2$, and $t = 2$. In this case, the search space for exhaustive testing is 1000^5 and the search tuple is equal to 2

$$\binom{p}{t}v^t = \frac{p!}{t!(p-t)}v^t = \frac{2!}{2!(2-2)}1000^2 \tag{2}$$

Here, if both P and v are large so do the required search space and the required tuples (even if t is bounded at $t = 6$). To this end, no known strategy can handle both large parameters (P) and large values (v) with rich constraints (C). Currently, there is a limit on the size of the search space as well as the

number of tuples to be processed owing to limited hardware and computational resources, rendering the need for much research as far as scalability is concerned. Concerning SPL as a special case of constrained interaction testing, while the parameter (P) can be large with rich constraints (C), the values (v) are always limited to two Boolean (true or false).

The second stumbling block relates to the process of finding and specifying constraints for large parameters and values. The use of feature diagram and constraints solver appears widespread for SPL; however, such an approach has not been sufficiently adopted for the general constrained interaction testing. If general constrained interaction testing is to be adopted for large parameters (P) and large values (v) with rich constraints (C), existing strategies need to integrate and embed constraints solver (or constraints programming) as part of their implementations.

H. Possible Research Directions for Future (RQ8)

Constrained interaction testing can be formulated as an optimization problem, resulting in the adoption of meta-heuristic based strategies in the literature. Meta-heuristic based strategies offer superior solutions (i.e., concerning test suite size) compared to existing approaches (i.e., owing to the systematic exploration and exploitation of the search space). Although useful, much-existing meta-heuristic based strategies have explored single meta-heuristic algorithm. As such, the exploration and exploitation of existing strategies have been limited based on the (local and global) search operators derived from a particular meta-heuristic algorithm. In this case, choosing a proper combination of search operators (termed as hybridization) can be the key to achieving good performance (as hybridization can capitalize on the strengths and address the deficiencies of each algorithm collectively and synergistically).

Hyper-heuristics have recently received considerable attention to addressing some of the hybridization as mentioned earlier issues [135], [136]. Specifically, hyper-heuristic represents an approach of using (meta)-heuristics to choose (meta)-heuristics to solve the optimization problem in hand. Unlike traditional meta-heuristics, which directly operates on the solution space, hyper-heuristics offer flexible integration and adaptive manipulation of the complete (low-level) meta-heuristics or merely partial adoption of a particular meta-heuristic search operator through non-domain feedback. In this manner, the hyper-heuristic can evolve its heuristic selection and acceptance mechanism in the search for a high-quality solution.

Apart from adopting hybridization with hyper-heuristic, the integration with machine learning appears to be a viable approach to improve the state-of-the-art of existing meta-heuristic algorithms for constrained interaction testing [137]. Machine learning relates to the study of fundamental laws that govern the computer learning process (i.e., concerning how to build systems that can automatically learn from experience). Machine learning techniques can be classified into three types: supervised, unsupervised, and reinforcement. Supervised learning involves learning the direct functional input-output mapping based on some set of training data and being able to perform a prediction on new data (e.g., deep learning approaches). Unlike supervised learning, unsupervised learning does not require precise training data. Specifically, unsupervised learning involves learning by drawing inferences (e.g., clustering) on the input datasets. Reinforcement learning relates to learning that allows mapping between states and actions to maximize reward signal by experimental discovery. This type of learning differs from supervised learning by the fact that it relies upon punish and reward mechanism and never corrects pairs of input-output data (even when dealing with sub-optimal responses).

To maximize the effectiveness of the fault finding efficiency, the constrained interaction testing can also be made as a multi-objective problem. Apart from avoiding forbidden tuples and maximizing the tuples coverage to obtain the most minimum test suite size, one possibility is to include minimizing the similarity among the test cases within the test suite. Here, the effectiveness of similarity distance metrics such as Euclidean distance, Hamming distance, Manhattan distance, Cosine similarity, Jaccard index and its variants can be investigated further. With a more diverse set of test cases, the fault detection rate may increase but potentially at the expense of a slight increase of test suite size. Additionally, from a different perspective, the same approach of using similarity metrics can also be used to prioritize existing test suite.

Finally, to leverage on the need for supporting large parameters (P), values (v), with rich constraints (C), there is a need to explore the new hybrid strategy based on cloud-based service-oriented architecture. In this manner, the user can exploit the computing power of the cloud to generate the constrained test suite. Additionally, the user can also explore the generation of constrained test suite as a service without having to purchase the actual tool.

V. Threats to validity

Like any other literature study, this study has threats to its validity. Many threats were eliminated by following well-known recommendations and advice on conducting literature studies.

First, as can be seen in the paper, several search strings were tried to assure maximum coverage of related papers. Although most of the related works have been selected here, 100% coverage of the papers cannot be guaranteed. There could be some uncovered papers by the search string; however, to overcome this threat a pilot search for several papers and also snowball sampling search was conducted.

Second, systematic literature studies could suffer from single-author bias during data extraction. To eliminate this threat, a double-checking and reviewing process by all authors individually were conducted. Besides, automatic mining and extraction tools were also tried in the spreadsheet to verify the results of the data extraction process.

Third, several papers during the study were excluded. The selection criteria are discussed explicitly in SectionIII-C. Some studies may also get excluded due to the scope of the paper itself. This study is dedicated to combinatorial interaction testing with constraints. Hence, those papers that cover combinatorial interaction testing without constraint support are excluded. The rationale behind this selection is that several studies cover these papers (see [2], [7], [138]). In addition, the constraint support in combinatorial testing was covered because it is an important aspect for the practical application of combinatorial interaction testing. The papers not published electronically were excluded; however, any contribution in the area of constrained interaction testing that is not published electronically is not expected.

VI. Conclusion

In this paper, an extensive literature study of 103 research papers published on constrained interaction testing between 2005 and 2016 is presented. In the study, the selected papers were analyzed from different perspectives based on a set of research questions. The results indicate that there is an increasing trend to address constrained interactions for the testing purpose. There is a mix of contributions in conferences, workshops, and journals with the majority of papers being published in conferences. The active authors and research groups in the field are further highlighted. Based on the analysis, the contributions in constrained interaction testing are grouped into four categories of constrained test generation studies, application studies, generation and application studies and model validation studies. The study found that to solve constraints, the researchers usually either use a constraint solver or exclude the constraints. The study further showed that the applications of constrained interaction testing are within software product lines, fault detection and characterization, test selection, security and GUI testing, among others. The study ends with a discussion of limitations, challenges, and areas for future research for constrained interaction testing.

Acknowledgment

The work undertaken in this paper is supported by the Fundamental Research Grant: Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite. Generation (grant no: RDU170103) from the Ministry of Higher Education Malaysia.

REFERENCES

[1] W. Afzal, R. Torkar, and R. Feldt, "A systematic review of search-based testing for non-functional system properties," *Information and Software Technology*, vol. 51, no. 6, pp. 957–976, 2009, 1518545.

[2] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys*, vol. 43, no. 2, pp. 1–29, 2011.

[3] R. E. Lopez-Herrejon, S. Fischer, R. Ramler, and A. Egyed, "A first systematic mapping study on combinatorial interaction testing for software product lines," pp. 1–10, 13-17 April 2015.

[4] B. S. Ahmed, L. M. Gambardella, W. Afzal, and K. Z. Zamli, "Handling constraints in combinatorial interaction testing in the presence of multi objective particle swarm and multithreading," *Information and Software Technology*, vol. 86, pp. 20 – 36, 2017.

[5] J. Petke, M. Cohen, M. Harman, and S. Yoo, "Practical combinatorial interaction testing: empirical findings on efficiency and early fault detection," *IEEE Transactions on Software Engineering*, vol. PP, no. 99, pp. 1–1, 2015.

[6] J. Petke, "Constraints: the future of combinatorial interaction testing," in *2015 IEEE/ACM 8th International Workshop on Search-Based Software Testing (SBST)*. IEEE, 2015, Conference Proceedings, pp. 17–18.

[7] V. Kuliamin and A. Petukhov, "A survey of methods for constructing covering arrays," *Programming and Computer Software*, vol. 37, no. 3, pp. 121–146, 2011.

[8] B. S. Ahmed and K. Z. Zamli, "A review of covering arrays and their application to software testing," *Journal of Computer Science*, vol. 7, no. 9, pp. 1375–1385, 2011.

[9] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 742–762, 2010.

[10] P. A. da Mota Silveira Neto, I. d. Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira, "A systematic mapping study of software product lines testing," *Information and Software Technology*, vol. 53, no. 5, pp. 407–423, 2011.

[11] S. Zein, N. Salleh, and J. Grundy, "A systematic mapping study of mobile application testing techniques," *Journal of Systems and Software*, vol. 117, pp. 334–356, 2016.

[12] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," EBSE Technical Report, Report, 2007.

[13] T. Dyba, T. Dingsoyr, and G. K. Hanssen, "Applying systematic reviews to diverse study types: an experience report," pp. 225–234, 2007.

[14] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1–18, 2015.

[15] N. S. R. Alves, T. S. Mendes, M. G. de Mendonca, R. O. Spinola, F. Shull, and C. Seaman, "Identification and management of technical debt: A systematic mapping study," *Information and Software Technology*, vol. 70, pp. 100–121, 2016.

[16] C. V. C. de Magalhaes, F. Q. B. da Silva, R. E. S. Santos, and M. Suassuna, "Investigations about replication of empirical studies in software engineering: A systematic mapping study," *Information and Software Technology*, vol. 64, pp. 76–101, 2015.

[17] M. B. Cohen, "Designing test suites for software interaction testing," PhD Thesis, 2004.

[18] B. Hnich, S. Prestwich, and E. Selensky, *Constraint-based approaches to the covering test problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 172–186.

[19] R. C. Bryce and C. J. Colbourn, "Prioritized interaction testing for pairwise coverage with seeding and constraints," *Information and Software Technology*, vol. 48, no. 10, pp. 960–970, 2006.

[20] B. Hnich, S. D. Prestwich, E. Selensky, and B. M. Smith, "Constraint models for the covering test problem," *Constraints*, vol. 11, no. 2, pp. 199–219, 2006.

[21] M. B. Cohen, M. B. Dwyer, and J. Shi, *Coverage and adequacy in software product line testing*. New York: ACM, 2006.

[22] M. B. Cohen, M. B. Dwyer, and J. shi, "Exploiting constraint solving history to construct interaction test suites," in *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*, 2007, pp. 121–132.

[23] 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," *IEEE Transactions on Software Engineering*, vol. 34, no. 5, pp. 633–650, 2008.

[24] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "An improved meta-heuristic search for constrained interaction testing," pp. 13–22, 2009.

[25] B. Garvin, M. Cohen, and M. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 61–102, 2011.

[26] M. B. Cohen, M. B. Dwyer, and J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," in *Proceedings of the 2007 International Symposium on Software Testing and Analysis*, ser. ISSTA '07. New York, NY, USA: ACM, 2007, pp. 129–139.

[27] I. Cabral, M. B. Cohen, and G. Rothermel, "Improving the testing and testability of software product lines," in *Proceedings of the 14th International Conference on Software Product Lines: Going Beyond*, ser. SPLC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 241–255.

[28] S. Fouché, M. B. Cohen, and A. Porter, "Incremental covering array failure characterization in large configuration spaces," in *Proceedings of the Eighteenth International Symposium on Software Testing and Analysis*, ser. ISSTA '09. New York, NY, USA: ACM, 2009, pp. 177–188.

[29] J. Petke, S. Yoo, M. B. Cohen, and M. Harman, "Efficiency and early fault detection with lower and higher strength combinatorial interaction testing," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: ACM, 2013, pp. 26–36.

[30] A. Calvagna, A. Fornaia, and E. Tramontana, "Random versus combinatorial effectiveness in software conformance testing: A case study," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. New York, NY, USA: ACM, 2015, pp. 1797–1802.

[31] Y. Jia, M. B. Cohen, M. Harman, and J. Petke, "Learning combinatorial interaction test generation strategies using hyperheuristic search," pp. 540–550, 2015.

[32] S. Huang, M. B. Cohen, and A. M. Memon, "Repairing gui test suites using a genetic algorithm," in *3rd IEEE International Conference on Software Testing, Verification and Validation*. IEEE Computer Society, 2010, Conference Proceedings, pp. 245–254.

[33] W. Grieskamp, X. Qu, X. Wei, N. Kicillof, and M. B. Cohen, *Interaction coverage meets path coverage by SMT constraint solving*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 97–112.

[34] J. Lin, C. Luo, S. Cai, K. Su, D. Hao, and L. Zhang, "Tca: An efficient two-mode meta-heuristic algorithm for combinatorial test generation (t)," in *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, Conference Proceedings, pp. 494–505.

[35] Y. Zhao, Z. Zhang, J. Yan, and J. Zhang, "Cascade: A test generation tool for combinatorial testing," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, 2013, Conference Proceedings, pp. 267–270.

[36] Z. Zhang, J. Yan, Y. Zhao, and J. Zhang, "Generating combinatorial test suite using combinatorial optimization," *Journal of Systems and Software*, vol. 98, pp. 191–207, 2014.

[37] H. Liu, F. Ma, and J. Zhang, *Generating covering arrays with pseudo-boolean constraint solving and balancing heuristic*. Cham: Springer International Publishing, 2016, pp. 262–270.

[38] F. Ma and J. Zhang, *Finding orthogonal arrays using satisfiability checkers and symmetry breaking constraints*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 247–259.

[39] J. Zhang, F. Ma, and Z. Zhang, *Faulty interaction identification via constraint solving and optimization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 186–199.

[40] F. Ma, "Constraint solving techniques for software testing and analysis," in *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ser. ICSE '10. New York, NY, USA: ACM, 2010, pp. 417–420.

[41] M. Al-Hajjaji, T. Thüm, M. Lochau, J. Meinicke, and G. Saake, "Effective product-line testing using similarity-based product prioritization," *Software and Systems Modeling*, pp. 1–23, 2016.

[42] M. Lochau, S. Oster, U. Goltz, and A. SchÃŒrr, "Model-based pairwise testing for feature interaction coverage in software product line engineering," *Software Quality Journal*, vol. 20, no. 3, pp. 567–604, 2012.

[43] M. Al-Hajjaji, T. Thüm, J. Meinicke, M. Lochau, and G. Saake, "Similarity-based prioritization in software product-line testing," in *Proceedings of the 18th International Software Product Line Conference - Volume 1*, ser. SPLC '14. New York, NY, USA: ACM, 2014, pp. 197–206.

[44] G. Perrouin, S. Oster, S. Sen, J. Klein, B. Baudry, and Y. le Traon, "Pairwise testing for software product lines: comparison of two approaches," *Software Quality Journal*, vol. 20, no. 3, pp. 605–643, 2012.

[45] S. Oster, F. Markert, and P. Ritter, *Automated incremental pairwise testing of software product lines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 196–210.

[46] H. Cichos, S. Oster, M. Lochau, and A. Schürr, *Model-Based Coverage-Driven Test Suite Generation for Software Product Lines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 425–439.

[47] S. Oster, M. Zink, M. Lochau, and M. Grechanik, *Pairwise feature-interaction testing for SPLs: potentials and limitations*. Munich: ACM, 2011.

[48] A. Gargantini and P. Vavassori, "Citlab: A laboratory for combinatorial interaction testing," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, 2012, Conference Proceedings, pp. 559–568.

[49] A. Calvagna, A. Gargantini, and P. Vavassori, "Combinatorial interaction testing with citlab," in *IEEE Sixth International Conference on Software Testing, Verification and Validation*, 2013, Conference Proceedings, pp. 376–382.

[50] A. Gargantini and P. Vavassori, *Efficient combinatorial test generation based on multivalued decision diagrams*. Cham: Springer International Publishing, 2014, pp. 220–235.

[51] A. Calvagna and A. Gargantini, *Combining satisfiability solving and heuristics to constrained combinatorial interaction testing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 27–42.

[52] A. Calvagna, A. Gargantini, and P. Vavassori, "Combinatorial testing for feature models using citlab," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, 2013, Conference Proceedings, pp. 338–347.

[53] A. Gargantini and G. Fraser, "Generating minimal fault detecting test suites for general boolean specifications," *Information and Software Technology*, vol. 53, no. 11, pp. 1263–1273, 2011.

[54] A. Calvagna and A. Gargantini, "A formal logic approach to constrained combinatorial testing," *Journal of Automated Reasoning*, vol. 45, no. 4, pp. 331–358, 2010.

[55] A. Calvagna and a. Gargantini, "A logic-based approach to combinatorial testing with constraints," in *Proceedings of the 2Nd International Conference on Tests and Proofs*, ser. TAP08. Berlin, Heidelberg: Springer-Verlag, 2008, Conference Proceedings, pp. 66–83.

[56] C. Henard, M. Papadakis, and Y. L. Traon, "Flattening or not of the combinatorial interaction testing models?" in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2015, Conference Proceedings, pp. 1–4.

[57] G. Perrouin, S. Sen, J. Klein, B. Baudry, and Y. le Traon, *Automated and scalable t-wise test case generation strategies for software product lines*. Paris: IEEE, 2010.

[58] C. Henard, M. Papadakis, and Y. Le Traon, *Mutation-based generation of software product line test configurations*. Cham: Springer International Publishing, 2014, pp. 92–106.

[59] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon, "Pledge: A product line editor and test generation tool," in *Proceedings of the 17th International Software Product Line Conference Co-located Workshops*, ser. SPLC '13 Workshops. New York, NY, USA: ACM, 2013, pp. 126–129.

[60] S. Sen, S. D. Alesio, D. Marijan, and A. Sarkar, "Evaluating reconfiguration impact in self-adaptive systems - an approach based on combinatorial interaction testing," in *41st Euromicro Conference on Software Engineering and Advanced Applications*, 2015, Conference Proceedings, pp. 250–254.

[61] D. Marijan, A. Gotlieb, S. Sen, and A. Hervieu, "Practical pairwise testing for software product lines," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: ACM, 2013, pp. 227–235.

[62] A. Hervieu, B. Baudry, and A. Gotlieb, "Pacogen: Automatic generation of pairwise test configurations from feature models," in *Proceedings of the 2011 IEEE 22Nd International Symposium on Software Reliability Engineering*, ser. ISSRE '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 120–129.

[63] M. F. Johansen, O. y. Haugen, F. Fleurey, A. G. Eldegard, and T. r. Syversen, *Generating better partial covering arrays by modeling weights on sub-product lines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 269–284.

[64] M. F. Johansen, O. y. Haugen, F. Fleurey, E. Carlson, J. Endresen, and T. Wien, *A technique for agile and automatic interaction testing for product lines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 39–54.

[65] M. F. Johansen, O. y. Haugen, and F. Fleurey, *Properties of realistic feature models make combinatorial testing of product lines feasible*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 638–652.

[66] M. F. Johansen, O. Haugen, and F. Fleurey, "An algorithm for generating t-wise covering arrays from large feature models," in *Proceedings of the 16th International Software Product Line Conference - Volume 1*, ser. SPLC '12. New York, NY, USA: ACM, 2012, pp. 46–55.

[67] R. R. Othman and K. Z. Zamli, "Input-input relationship constraints in t-way testing," in *2011 IEEE Symposium on Industrial Electronics and Applications*, 2011, Conference Proceedings, pp. 527–531.

[68] A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," *Information and Software Technology*, vol. 54, no. 6, pp. 553–568, 2012.

[69] A. A. Al-Sewari and K. Z. Zamli, *Constraints dependent T-Way test suite generation using harmony search strategy*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–11.

[70] Y. A. Alsariera, M. A. Majid, and K. Z. Zamli, "Splba: An interaction strategy for testing software product lines using the bat-inspired algorithm," in *4th International Conference on Software Engineering and Computer Systems (ICSECS)*, 2015, Conference Proceedings, pp. 148–153.

[71] X. Devroey, G. Perrouin, A. Legay, M. Cordy, P.-Y. Schobbens, and P. Heymans, *Coverage criteria for behavioural testing of software product lines*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 336–350.

[72] M. Al-Hajjaji, S. Krieter, T. Thüm, M. Lochau, and G. Saake, "Incling: Efficient product-line testing using incremental pairwise sampling," in *GPCE 2016 - Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, co-located with SPLASH 2016*, 2016, pp. 144–155.

[73] S. Fischer, L. Linsbauer, R. E. Lopez-Herrejon, and A. Egyed, "A source level empirical study of features and their interactions in variable software," in *IEEE 16th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2016, Conference Proceedings, pp. 197–206.

[74] R. E. Lopez-Herrejon, F. Chicano, J. Ferrer, A. Egyed, and E. Alba, "Multi-objective optimal test suite computation for software product line pairwise testing," pp. 404–407, 2013.

[75] R. E. Lopez-Herrejon, J. Javier Ferrer, F. Chicano, E. N. Haslinger, A. Egyed, and E. Alba, "A parallel evolutionary algorithm for prioritized pairwise testing of software product lines," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14. New York, NY, USA: ACM, 2014, pp. 1255–1262.

[76] W.-T. Tsai, C. J. Colbourn, J. Luo, G. Qi, Q. Li, and X. Bai, "Test algebra for combinatorial testing," pp. 19–25, 2013.

[77] C. J. Colbourn and D. W. McClary, "Locating and detecting arrays for interaction faults," *Journal of Combinatorial Optimization*, vol. 15, no. 1, pp. 17–48, 2008.

[78] A. Yamada, T. Kitamura, C. Artho, E. H. Choi, Y. Oiwa, and A. Biere, "Optimization of combinatorial testing by incremental sat solving," in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, Conference Proceedings, pp. 1–10.

[79] A. Yamada, A. Biere, C. Artho, T. Kitamura, and E.-H. Choi, "Greedy combinatorial test case generation using unsatisfiable cores," pp. 614–624, 2016.

[80] E. H. Choi, S. Kawabata, O. Mizuno, C. Artho, and T. Kitamura, "Test effectiveness evaluation of prioritized combinatorial testing: a case study," in *IEEE International Conference on Software Quality, Reliability and Security (QRS)*, 2016, Conference Proceedings, pp. 61–68.

[81] A. Hervieu, D. Marijan, A. Gotlieb, and B. Baudry, "Practical minimization of pairwise-covering test configurations using constraint programming," *Information and Software Technology*, vol. 71, pp. 129–146, 2016.

[82] L. Yu, F. Duan, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Constraint handling in combinatorial test generation using forbidden tuples," in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2015, Conference Proceedings, pp. 1–9.

[83] P. M. Kruse, "Test oracles and test script generation in combinatorial testing," in *IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2016, Conference Proceedings, pp. 75–82.

[84] S. Nakornburi and T. Suwannasart, "A tool for constrained pairwise test case generation using statistical user profile based prioritization," in

13th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2016, Conference Proceedings, pp. 1–6.

[85] J. Yuan, C. Jiang, and Z. Jiang, "Improved extremal optimization for constrained pairwise testing," in *International Conference on Research Challenges in Computer Science*, 2009, Conference Proceedings.

[86] E. Salecker, R. Reicherdt, and S. Glesner, "Calculating prioritized interaction test sets with constraints using binary decision diagrams," pp. 278–285, 2011.

[87] B. Pérez Lamancha, M. Polo, and M. Piattini, "Prow: A pairwise algorithm with constraints, order and weight," *Journal of Systems and Software*, vol. 99, pp. 1–19, 2015.

[88] S. Hallé, E. La Chance, and S. Gaboury, *Graph methods for generating test cases with universal and existential constraints*. Cham: Springer International Publishing, 2015, pp. 55–70.

[89] P. Danziger, E. Mendelsohn, L. Moura, and B. Stevens, "Covering arrays avoiding forbidden edges," in *International Conference on Combinatorial Optimization and Applications*. Springer Berlin / Heidelberg, Conference Proceedings, pp. 298–308.

[90] Y. Sheng, C. Wei, G. Wang, S. Jiang, and Y. Chen, "Constraint test cases generation based on particle swarm optimization," in *Proceedings - 22nd ISSAT International Conference on Reliability and Quality in Design*, 2016, Conference Proceedings, pp. 329–333.

[91] R. N. Kacker, D. Richard Kuhn, Y. Lei, and J. F. Lawrence, "Combinatorial testing for software: An adaptation of design of experiments," *Measurement*, vol. 46, no. 9, pp. 3745–3752, 2013.

[92] G. B. Sherwood, "Embedded functions in combinatorial test designs," in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Conference Proceedings, pp. 1–10.

[93] J. Bozic, B. Garn, D. E. Simos, and F. Wotawa, "Evaluation of the ipo-family algorithms for test case generation in web security testing," in *2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Conference Proceedings, pp. 1–10.

[94] N. Li, Y. Lei, H. R. Khan, J. Liu, and Y. Guo, "Applying combinatorial test data generation to big data applications," pp. 637–647, 2016.

[95] G. Fraser and A. Gargantini, "Generating minimal fault detecting test suites for boolean expressions," in *Third International Conference on Software Testing, Verification, and Validation Workshops*, 2010, Conference Proceedings, pp. 37–45.

[96] P. M. Kruse, J. Bauer, and J. Wegener, "Numerical constraints for combinatorial interaction testing," in *IEEE Fifth International Conference on Software Testing, Verification and Validation*, 2012, Conference Proceedings, pp. 758–763.

[97] C. Yilmaz, "Test case-aware combinatorial interaction testing," *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 684–706, 2013.

[98] M. Palacios, J. García-Fanjul, J. Tuya, and G. Spanoudakis, "Automatic test case generation for ws-agreements using combinatorial testing," *Computer Standards and Interfaces*, vol. 38, pp. 84–100, 2015.

[99] J. A. Galindo, H. Turner, D. Benavides, and J. White, "Testing variability-intensive systems using automated analysis: an application to android," *Software Quality Journal*, vol. 24, no. 2, pp. 365–405, 2016.

[100] A. Arrieta, G. Sagardui, L. Etxeberria, and J. Zander, "Automatic generation of test system instances for configurable cyber-physical systems," *Software Quality Journal*, pp. 1–43, 2016.

[101] R. A. Matnei Filho and S. R. Vergilio, "A multi-objective test data generation approach for mutation testing of feature models," *Journal of Software Engineering Research and Development*, vol. 4, no. 1, p. 4, 2016.

[102] F. Bouquet, F. Peureux, and F. Ambert, *Model-based testing for functional and security test generation*. Cham: Springer International Publishing, 2014, pp. 1–33.

[103] H. Zhong, L. Zhang, and S. Khurshid, *The comKorat tool: unified combinatorial and constraint-based generation of structurally complex tests*. Cham: Springer International Publishing, 2016, pp. 107–113.

[104] B. P. Lamancha and M. P. Usaola, *Testing product generation in software product lines using pairwise for features coverage*. Berlin: Springer, 2010.

[105] Y. Li, Z.-a. Sun, and J.-Y. Fang, "Generating an automated test suite by variable strength combinatorial testing for web services," *Journal of Computing and Information Technology*, vol. 24, no. 3, pp. 271–282, 2016.

[106] K. Go, S. Kang, J. Baik, and M. Kim, "Pairwise testing for systems with data derived from real-valued variable inputs," *Software - Practice and Experience*, vol. 46, no. 3, pp. 381–403, 2016.

[107] S. Nakornburi and T. Suwannasart, "Constrained pairwise test case generation approach based on statistical user profile," in *Lecture Notes in Engineering and Computer Science*, vol. 1, 2016, Conference Proceedings, pp. 445–448.

[108] H. Zhong, L. Zhang, and S. Khurshid, "Combinatorial generation of structurally complex test inputs for commercial software applications," in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, vol. 13-18-November, 2016, pp. 981–986.

[109] C. H. P. Kim, D. Batory, and S. Khurshid, "Eliminating products to test in a software product line," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 139–142.

[110] C. H. P. Kim, D. S. Batory, and S. Khurshid, "Reducing combinatorics in testing product lines," in *Proceedings of the Tenth International Conference on Aspect-oriented Software Development*, ser. AOSD '11. New York, NY, USA: ACM, 2011, pp. 57–68.

[111] R. Gao, L. Hu, W. E. Wong, H. L. Lu, and S. K. Huang, "Effective test generation for combinatorial decision coverage," in *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2016, Conference Proceedings, pp. 47–54.

[112] E. Salecker and S. Glesner, "Combinatorial interaction testing for test selection in grammar-based testing," pp. 610–619, 2012.

[113] M. Banbara, H. Matsunaka, N. Tamura, and K. Inoue, *Generating combinatorial test cases by efficient SAT encodings suitable for CDCL SAT solvers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 112–126.

[114] S. Bauersfeld, S. Wappler, and J. Wegener, *A metaheuristic approach to test sequence generation for applications with a GUI*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 173–187.

[115] A. Kalaee and V. Rafe, "An optimal solution for test case generation using robdd graph and pso algorithm," *Quality and Reliability Engineering International*, vol. 32, no. 7, pp. 2263–2279, 2016.

[116] P. Arcaini, A. Gargantini, and P. Vavassori, "Validation of models and tests for constrained combinatorial interaction testing," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops*, Conference Proceedings, pp. 98–107.

[117] S. K. Khalsa and Y. Labiche, "An extension of category partition testing for highly constrained systems," in *IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*, 2016, Conference Proceedings, pp. 47–54.

[118] I. Segall, R. Tzoref-Brill, and A. Zlotnick, "Simplified modeling of combinatorial test spaces," in *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, 2012, Conference Proceedings, pp. 573–579.

[119] R. Tzoref-Brill and S. Maoz, *Lattice-based semantics for combinatorial model evolution*. Cham: Springer International Publishing, 2015, pp. 276–292.

[120] M. Spichkova and A. Zamansky, "A human-centred framework for combinatorial test design," in *ENASE 2016 - Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, 2016, Conference Proceedings, pp. 228–233, export Date: 13 February 2017.

[121] K. C. Tai and Y. Lie, "In-parameter-order: a test generation strategy for pairwise testing," in *3rd IEEE International Symposium on High-Assurance Systems Engineering*. IEEE Computer Society, Conference Proceedings, pp. 254–261.

[122] J. Bozic, D. E. Simos, and F. Wotawa, "Attack pattern-based combinatorial testing," in *Proceedings of the 9th International Workshop on Automation of Software Test*, ser. AST 2014. New York, NY, USA: ACM, 2014, pp. 1–7.

[123] B. Garn, I. Kapsalis, D. E. Simos, and S. Winkler, "On the applicability of combinatorial testing to web application security testing: A case study," in *Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing*, ser. JAMAICA 2014. New York, NY, USA: ACM, 2014, pp. 16–21.

[124] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "Ipog: a general strategy for t-way software testing," in *4th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*. IEEE Computer Society, 2007, Conference Proceedings, pp. 549–556, 1253335549-556.

[125] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and j. Lawrence, "Ipog-ipog-d: Efficient test generation for multi-way combinatorial testing," *Softw. Test. Verif. Reliab.*, vol. 18, no. 3, pp. 125–148, Sep. 2008.

[126] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology*, vol. 113, no. 5, pp. 287–297, 2008.

[127] G. J. Holzmann, "The model checker spin," *IEEE Transactions on Software Engineering*, vol. 23, no. 5, pp. 279–295, 1997, 260902.
[128] J. Lee, S. Kang, and D. Lee, "A survey on software product line testing," in *Proceedings of the 16th International Software Product Line Conference - Volume 1*, ser. SPLC '12. New York, NY, USA: ACM, 2012, pp. 31–40.
[129] C. Yilmaz, M. B. Cohen, and A. A. Porter, "Covering arrays for efficient fault characterization in complex configuration spaces," *IEEE Transactions on Software Engineering*, vol. 32, no. 1, pp. 20–34, 2006.
[130] A. Memon, A. Porter, C. Yilmaz, A. Nagarajan, D. Schmidt, and B. Natarajan, "Skoll: distributed continuous quality assurance," in *26th International Conference on Software Engineering*. 999451: IEEE Computer Society, 2004, pp. 459–468.
[131] A. Deshpande and D. Riehle, *The Total Growth of Open Source*. Boston, MA: Springer US, 2008, pp. 197–209.
[132] D. R. Kuhn and M. J. Reilly, "An investigation of the applicability of design of experiments to software testing," in *27th Annual NASA Goddard Software Engineering Workshop (SEW-27'02)*. IEEE Computer Society, 2002, Conference Proceedings, p. 91, 83012391.
[133] D. R. Kuhn, D. R. Wallace, and J. Gallo, A.M., "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, pp. 418–421, 2004.
[134] R. C. Bryce and C. J. Colbourn, "Test prioritization for pairwise interaction coverage," in *Proceedings of the 1st International Workshop on Advances in Model-based Testing*, ser. A-MOST '05. New York, NY, USA: ACM, 2005, pp. 1–7.
[135] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, *A Classification of Hyper-heuristic Approaches*. Boston, MA: Springer US, 2010, pp. 449–468.
[136] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, Dec. 2003.
[137] K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, "An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation," *Information Sciences*, vol. 399, pp. 121 – 153, 2017.
[138] S. K. Khalsa and Y. Labiche, "An orchestrated survey of available algorithms and tools for combinatorial testing," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*, Nov 2014, pp. 323–334.

APPENDIX A. LIST OF STUDIED PAPERS

ID#	Ref.	Paper Title	Year
1	[52]	Combinatorial Testing for Feature Models Using CITLAB	2013
2	[76]	Test Algebra for Combinatorial Testing	2013
3	[74]	Multi-Objective Optimal Test Suite Computation for Software Product Line Pairwise Testing	2013
4	[116]	Validation of Models and Tests for Constrained Combinatorial Interaction Testing	2014
5	[5]	Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection	2015
6	[78]	Optimization of Combinatorial Testing by Incremental SAT Solving	2015
7	[92]	Embedded Functions in Combinatorial Test Designs	2015
8	[93]	Evaluation of the IPO-Family Algorithms for Test Case Generation in Web Security Testing	2015
9	[82]	Constraint Handling In Combinatorial Test Generation Using Forbidden Tuples	2015
10	[56]	Flattening or Not of the Combinatorial Interaction Testing Models?	2015
11	[31]	Learning Combinatorial Interaction Test Generation Strategies using Hyperheuristic Search	2015
12	[60]	Evaluating Reconfiguration Impact in Self-adaptive Systems	2015
13	[70]	SPLBA: An Interaction Strategy for Testing Software Product Lines Using the Bat-Inspired Algorithm	2015
14	[34]	TCA: An Efficient Two-Mode Meta-Heuristic Algorithm for Combinatorial Test Generation	2015
15	[117]	An extension of category partition testing for highly constrained systems	2016
16	[83]	Test oracles and test script generation in combinatorial testing	2016
17	[111]	Effective test generation for combinatorial decision coverage	2016
18	[79]	Greedy combinatorial test case generation using unsatisfiable cores	2016
19	[94]	Applying Combinatorial Test Data Generation to Big Data Applications	2016
20	[80]	Test effectiveness evaluation of prioritized combinatorial testing: a case study	2016
21	[84]	A tool for constrained pairwise test case generation using statistical user profile based prioritization	2016
22	[73]	A source level empirical study of features and their interactions in variable software	2016
23	[22]	Exploiting constraint solving history to construct interaction test suites	2007
24	[23]	Constructing interaction test Suites for highly-configurable systems in the presence of constraints: a greedy approach	2008
25	[24]	An improved meta-heuristic search for constrained interaction testing	2009
26	[85]	Improved extremal optimization for constrained pairwise testing	2009
27	[95]	Generating minimal fault detecting test suites for boolean expressions	2010
28	[57]	Automated and scalable t-wise test case generation strategies for software product lines	2010

29	[32]	Repairing GUI test suites using a genetic algorithm	2010
30	[86]	Calculating prioritized interaction test sets with constraints using binary decision diagrams	2011
31	[67]	Input-input relationship constraints in T-way testing	2011
32	[48]	CITLAB: A laboratory for combinatorial interaction testing	2012
33	[118]	Simplified modeling of combinatorial test spaces	2012
34	[112]	Combinatorial interaction testing for test selection in grammar-based testing	2012
35	[96]	Numerical constraints for combinatorial interaction testing	2012
36	[97]	Test case-aware combinatorial interaction testing	2013
37	[49]	Combinatorial interaction testing with CITLAB	2013
38	[35]	Cascade: A test generation tool for combinatorial testing	2013
39	[87]	PROW: A Pairwise algorithm with constRaints, Order and Weight	2015
40	[36]	Generating combinatorial test suite using combinatorial optimization	2014
41	[98]	Automatic test case generation for WS-Agreements using combinatorial testing	2015
42	[19]	Prioritized interaction testing for pair-wise coverage with seeding and constraints	2006
43	[68]	Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support	2012
44	[81]	Practical minimization of pairwise-covering test configurations using constraint programming	2016
45	[41]	Effective product-line testing using similarity-based product prioritization	2016
46	[20]	Constraint Models for the Covering Test Problem	2006
47	[25]	Evaluating improvements to a meta-heuristic search for constrained interaction testing	2011
48	[54]	A Formal Logic Approach to Constrained Combinatorial Testing	2010
49	[77]	Locating and detecting arrays for interaction faults	2008
50	[44]	Pairwise testing for software product lines: comparison of two approaches	2012
51	[42]	Model-based pairwise testing for feature interaction coverage in software product line engineering	2012
52	[99]	Testing variability-intensive systems using automated analysis: an application to Android	2016
53	[100]	Automatic generation of test system instances for configurable cyber-physical systems	2016
54	[101]	A multi-objective test data generation approach for mutation testing of feature models	2016
55	[58]	Mutation-Based Generation of Software Product Line Test Configurations	2014
56	[102]	Model-Based Testing for Functional and Security Test Generation	2014
57	[50]	Efficient Combinatorial Test Generation Based on Multivalued Decision Diagrams	2014
58	[119]	Lattice-Based Semantics for Combinatorial Model Evolution	2015
59	[88]	Graph Methods for Generating Test Cases with Universal and Existential Constraints	2015
60	[103]	The comKorat Tool: Unified Combinatorial and Constraint-Based Generation of Structurally Complex Tests	2016
61	[37]	Generating Covering Arrays with Pseudo-Boolean Constraint Solving and Balancing Heuristic	2016
62	[55]	A Logic-Based Approach to Combinatorial Testing with Constraints	2008
63	[89]	Covering Arrays Avoiding Forbidden Edges	2008
64	[38]	Finding Orthogonal Arrays Using Satisfiability Checkers and Symmetry Breaking Constraints	2008
65	[51]	Combining Satisfiability Solving and Heuristics to Constrained Combinatorial Interaction Testing	2009
66	[33]	Interaction Coverage Meets Path Coverage by SMT Constraint Solving	2009
67	[45]	Automated Incremental Pairwise Testing of Software Product Lines	2010
68	[27]	Improving the Testing and Testability of Software Product Lines	2010
69	[113]	Generating Combinatorial Test Cases by Efficient SAT Encodings Suitable for CDCL SAT Solvers	2010
70	[104]	Testing Product Generation in Software Product Lines Using Pairwise for Features Coverage	2010
71	[114]	A Metaheuristic Approach to Test Sequence Generation for Applications with a GUI	2011
72	[65]	Properties of Realistic Feature Models Make Combinatorial Testing of Product Lines Feasible	2011
73	[39]	Faulty Interaction Identification via Constraint Solving and Optimization	2012
74	[69]	Constraints Dependent T-Way Test Suite Generation Using Harmony Search Strategy	2012
75	[63]	Generating Better Partial Covering Arrays by Modeling Weights on Sub-product Lines	2012
76	[64]	A Technique for Agile and Automatic Interaction Testing for Product Lines	2012
77	[71]	Coverage Criteria for Behavioural Testing of Software Product Lines	2014
78	[18]	Constraint-Based Approaches to the Covering Test Problem	2005
79	[43]	Similarity-based Prioritization in Software Product-line Testing	2014
80	[66]	An Algorithm for Generating t-wise Covering Arrays from Large Feature Models	2012
81	[59]	PLEDGE: A Product Line Editor and Test Generation Tool	2013
82	[26]	Interaction Testing of Highly-Configurable Systems in the Presence of Constraints	2007
83	[28]	Incremental Covering Array Failure Characterization in Large Configuration Spaces	2009
84	[61]	Practical Pairwise Testing for Software Product Lines	2013
85	[40]	Constraint Solving Techniques for Software Testing and Analysis	2010
86	[75]	A Parallel Evolutionary Algorithm for Prioritized Pairwise Testing of Software Product Lines	2014
87	[30]	Random Versus Combinatorial Effectiveness in Software Conformance Testing: A Case Study	2015
88	[29]	Efficiency and Early Fault Detection with Lower and Higher Strength Combinatorial Interaction Testing	2013
89	[105]	Generating an Automated Test Suite by Variable Strength Combinatorial Testing for Web Services	2016
90	[90]	Constraint Test Cases Generation Based on Particle Swarm Optimization	2016
91	[120]	A Human-centred Framework for Combinatorial Test Design	2016
92	[106]	Pairwise testing for systems with data derived from real-valued variable inputs	2016
93	[107]	Constrained Pairwise Test Case Generation Approach based on Statistical User Profile	2016
94	[115]	An Optimal Solution for Test Case Generation Using ROBDD Graph and PSO Algorithm	2016
95	[72]	IncLing: Efficient Product-Line Testing using Incremental Pairwise Sampling	2016
96	[108]	Combinatorial Generation of Structurally Complex Test Inputs for Commercial Software Applications	2016
97	[91]	ACTS: A Combinatorial Test Generation Tool	2013
98	[62]	PACOGEN: Automatic Generation of Pairwise Test Configurations From Feature Models	2011
99	[110]	Reducing combinatorics in testing product lines	2011

100	[109]	Eliminating products to test in a software product line	2010
101	[21]	Coverage and adequacy in software product line testing	2006
102	[46]	Model-based coverage-driven test suite generation for software product lines	2011
103	[47]	Pairwise feature-interaction testing for SPLs: potentials and limitations	2011

APPENDIX B. LIST OF ACTIVE JOURNALS WITH ABBREVIATIONS

Acronym	Journal Full Name
IEEE T Software Eng	IEEE Transactions on Software Engineering
J Syst Software	The Journal of Systems and Software
Eur J Combin	European Journal of Combinatorics
Comp Stand Inter	Computer Standards & Interfaces journal
Inform Software Tech	Information and Software Technology Journal
Softw Syst Model	Software & Systems Modeling Journal
Constraints	Constraints Journal
Empir Softw Eng	Empirical Software Engineering journal
J Autom Reasoning	Journal of Automated Reasoning
J Comb Optim	Journal of Combinatorial Optimization
Software Qual J	Software Quality Journal
J Softw Eng Res Dev	Journal of Software Engineering Research and Development
J Compu Inform Tech	Journal of Computing and Information Technology
Softw Pract Exp	Software: Practice and Experience journal
Qual Reliab Eng Int	Quality and Reliability Engineering International Journal

APPENDIX C. LIST OF ACTIVE CONFERENCES WITH ABBREVIATIONS

Acronym	Conference Full Name
ICST	International Conference on Software Testing, Verification and Validation Workshops
ASE	International Conference on Automated Software Engineering
SSBSE	International Symposium on Search Based Software Engineering
FSE	International Symposium on Foundations of Software Engineering
SPLC	International Conference on Software Product Lines
ISSTA	International Symposium on Software Testing and Analysis
QRS	International Conference on Software Quality, Reliability and Security Companion
ICTSS	IFIP International Conference on Testing Software and Systems
ICSE	International Conference on Software Engineering
PRICAI	Pacific Rim International Conference on Artificial Intelligence
TAP	International Conference on Tests and Proofs
MODELS	International Conference on Model Driven Engineering Languages and Systems