Information and Software Technology 71 (2016) 129–146

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Practical minimization of pairwise-covering test configurations using constraint programming

Aymeric Hervieu [a], Dusica Marijan [b,*], Arnaud Gotlieb [b], Benoit Baudry [a]

[a] INRIA Rennes Bretagne Atlantique, France
[b] Simula Research Laboratory, Norway

ARTICLE INFO

Article history:
Received 23 January 2015
Revised 23 October 2015
Accepted 19 November 2015
Available online 2 December 2015

Keywords:
Variability testing
Highly-configurable software systems
Constraint programming

ABSTRACT

Context: Testing highly-configurable software systems is challenging due to a large number of test configurations that have to be carefully selected in order to reduce the testing effort as much as possible, while maintaining high software quality. Finding the smallest set of valid test configurations that ensure sufficient coverage of the system's feature interactions is thus the objective of validation engineers, especially when the execution of test configurations is costly or time-consuming. However, this problem is NP-hard in general and approximation algorithms have often been used to address it in practice.

Objective: In this paper, we explore an alternative exact approach based on constraint programming that will allow engineers to increase the effectiveness of configuration testing while keeping the number of configurations as low as possible.

Method: Our approach consists in using a (time-aware) minimization algorithm based on constraint programming. Given the amount of time, our solution generates a minimized set of valid test configurations that ensure coverage of all pairs of feature values (a.k.a. pairwise coverage). The approach has been implemented in a tool called PACOGEN.

Results: PACOGEN was evaluated on 224 feature models in comparison with the two existing tools that are based on a greedy algorithm. For 79% of 224 feature models, PACOGEN generated up to 60% fewer test configurations than the competitor tools. We further evaluated PACOGEN in the case study of an industrial video conferencing product line with a feature model of 169 features, and found 60% fewer configurations compared with the manual approach followed by test engineers. The set of test configurations generated by PACOGEN decreased the time required by test engineers in manual test configuration by 85%, increasing the feature-pairs coverage at the same time.

Conclusion: Our experimental evaluation concluded that optimal time-aware minimization of pairwise-covering test configurations is efficiently addressed using constraint programming techniques.

1. Introduction

Motivation. Modern software systems often contain a base functionality common to a set of related products and a number of specific functionalities and features that introduce variability within a set of software configurations. *Highly-configurable systems* are thus those systems that can be configured in many different ways and adapted to diverse user needs by a configuration process. Testing highly-configurable systems is challenging due to a large number of possible test configurations that must be minimized in order to reduce the testing effort, while not compromising the effectiveness of the testing process. If validation engineers ignore commonalities within the products when selecting test configurations, the overall testing effort/time usually exceeds the available testing time. A practical approach to cope with this problem is to abruptly cut off the number of test configurations, given the available testing time. This approach may cause missing serious software faults that are due to the interaction of specific software features. Moreover, in highly-configurable systems, particular failures can be detected only by testing specific software configurations [1], as different combinations of features lead to different functionality/products.

The MVPF problem. In a highly-configurable software setting, adopting a systematic testing methodology is of paramount importance for ensuring software quality. This means adopting well-recognized

* Corresponding author. Tel.: +47 45517996.
E-mail address: dusica@simula.no (D. Marijan).

http://dx.doi.org/10.1016/j.infsof.2015.11.007
0950-5849/

testing criteria to build a software testing process based on rational choices. In our work, we aim at selecting the *smallest* subset of valid test configurations ensuring that each possible pair of feature values is covered at least once. We call this problem the *Minimum Valid Pairwise-Feature-values coverage (MVPF) problem*. Through a specific constraint optimization model, MVPF problem aims at finding true optima (not an approximation to the optimal solution) of a configuration sampling problem, which is one differentiating characteristic of our approach over most of the existing work. Furthermore, MVPF problem incorporates the requirement for the validity of test configurations, satisfying the constraints existing among the features (including physical and business constraints).

Solving the MVPF problem is regarded as a minimum requirement when testing highly-configurable systems. Indeed, as observed in several studies [2,3], having covered all 2-way feature interactions increases the confidence of validation engineers in the quality of tested products (assuring, at least, against 2-way errors).

Solving the MVPF problem also means finding a set (not necessarily unique!) of valid test configurations of minimum cardinality. The rationale is to minimize the overall test effort by minimizing the number of considered test configurations, while maintaining their ability to detect failures.

NP-hardness of the MVPF problem. Unfortunately, as soon as constraints are involved among the features, the problem of generating exactly the minimum number of test configurations that cover all pairs of feature values is NP-hard. This is shown by the reduction to the boolean satisfiability problem [4]. There are several classes of approaches to such problems [5]. *Greedy algorithms* [6–9] consider one by one the configurations that cover the most pairs, checking their validity. This means that once a configuration has been included in the current set of valid configurations, it cannot be deleted and replaced by another configuration covering fewer but different pairs. Being approximations, these algorithms can only seldom reach the true minimum number of valid test configurations. In domains where test configurations take a long time to execute, finding an optimal number of test configurations becomes essential to decrease the overall testing effort. *Meta-heuristic searches* [10], as more sophisticated heuristics, can often find good solutions with less computational effort than simple heuristics, but still not guaranteeing a globally optimal solution. *Mathematical construction* based on covering arrays are exact approaches [11], but they do not handle cross-tree constraints, and often they are not well suited for a large number of parameters [12].

However, by checking afterwards the validity of the generated configurations, they can be used to solve the MVPF problem even if this may require in the worst case to generate all the possible covering arrays.

For a given instance of MVPF, providing the validation engineer with a certificate of minimality is only possible with exact methods but it may require an exponential amount of time in the worst case because of the NP-hardness of the problem. However, even if it is strongly desirable to get a set of valid configurations of minimal cardinality, obtaining this certificate of minimality is not required in industrial settings.

Solving an industrial problem. The MVPF problem was identified in the domain of large highly-configurable networking software by Cisco Systems Norway, an industrial partner of the Certus Centre.[1] Testing highly-configurable networking systems in a continuous integration environment requires not only selecting the relevant configurations to test (configurations that underwent changes), but also fitting the testing process for a limited testing time. For our partner, the ability to control (at any stage of the testing process) the time needed to complete testing is mandatory. As test configuration execution often requires manual hardware setup, which may take up to 30 min, finding an optimal set of valid test configurations is of paramount importance.

The proposed solution. To deal with this problem, we propose a novel Constraint Programming (CP) approach that manages test configurations from feature models. Feature modelling is a common approach for representing variability within a software product line, while Constraint Programming is a well-known programming paradigm dedicated to the resolution of hard combinatorial problems. Given a contract of time, our approach seeks for the minimum set of valid configurations solving the MVPF problem. A special data structure composed of finite-domain variables is used to encode a set of valid configurations. We introduce a dedicated combinatorial constraint (called *pairwise global constraint*) to enforce feature pairwise coverage over the data structure, while other logical constraints are used to capture hierarchical and cross-tree links between features. We also introduce special heuristics to order the feature pairs and to select the variable to enumerate first (*Pair-ordering and Variable-selection heuristics*). Finally, a constraint optimisation procedure, called *branch-and-bound*, is used to find a subset of test configurations.

By using this well-established and proved optimization procedure, our approach is an exact method to find a candidate set of minimum cardinality. When sufficient time is allocated to the method, it can provide a minimality certificate, proving that there is no smaller set. However, as the problem is NP-hard, it is necessary to control the time allocated to the method by setting a time-contract. When the time of the contract is exhausted while the method is still running, then the method returns the best found candidate and no certificate of minimality. This approach proved to be successful in solving the MVPF problem in practice, because the time spent in test generation has to be balanced with the time required to test a single configuration.

Implementation and experimental results. We implemented the approach in a freely available tool called PACOGEN.[2] The tool results from a two man-year development effort and incorporates subtle optimizations that are discussed in detail in Section 4 and evaluated in Section 6. We performed the experimental comparative study with the two existing greedy approaches that aim at solving the same problem, MosoPolite tool [7,13] and SPLCAT tool [9,14]. The comparison was performed using a large benchmark consisting of 224 feature models from SPLOT [15]. Our results show that PACOGEN produces up to 60% fewer configurations on average, respecting the MVPF problem, for 79% of the benchmark feature models for one approach, and up to 42% fewer test configurations for 6 of 7 feature models for the another approach. Furthermore, we applied and validated PACOGEN on an industrial case study provided by Cisco. The feature model of a Cisco video software system consists of 169 features and more than 10^9 possible configurations (valid and invalid). The experimental results show that PACOGEN produces 60% fewer configurations, compared with the manual approach followed by Cisco, while substantially increasing 2-way feature coverage. The automatically generated set of configurations decreased the time required by validation engineers in manual configuration specification by 85%.

Contributions. The idea of using CP to solve the MVPF problem was introduced by the authors in [16]. We have further enhanced the initial idea by introducing a dedicated algorithm for filtering invalid pairs (Section 4.3), and by proposing new search heuristics

[1] Certus Centre, hosted by SIMULA Research Laboratory, is a research-based innovation centre aiming at developing methods and tools in the domain of validation and verification of software-intensive systems, in collaboration with industrial and public administration partners.

[2] http://people.rennes.inria.fr/Arnaud.Gotlieb/resources/Pacogen/Pacogen.html.