Applied Soft Computing xxx (2016) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

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

Deriving products for variability test of Feature Models with a hyper-heuristic approach

Andrei Strickler, Jackson A. Prado Lima, Silvia R. Vergilio, Aurora T.R. Pozo*

DInf – Federal University of Paraná, CP: 19081, CEP 19031-970 Curitiba, Brazil

ARTICLE INFO

Article history:
Received 26 December 2015
Received in revised form 17 June 2016
Accepted 28 July 2016
Available online xxx

Keywords:
Software Product Line
Software testing
Hyper-heuristic

ABSTRACT

Deriving products from a Feature Model (FM) for testing Software Product Lines (SPLs) is a hard task. It is important to select a minimum number of products but, at the same time, to consider the coverage of testing criteria such as pairwise, among other factors. To solve such problems Multi-Objective Evolutionary Algorithms (MOEAs) have been successfully applied. However, to design a solution for this and other software engineering problems can be very difficult, because it is necessary to choose among different search operators and parameters. Hyper-heuristics can help in this task, and have raised interest in the Search-Based Software Engineering (SBSE) field. Considering the growing adoption of SPL in the industry and crescent demand for SPL testing approaches, this paper introduces a hyper-heuristic approach to automatically derive products to variability testing of SPLs. The approach works with MOEAs and two selection methods, random and based on FRR-MAB (Fitness Rate Rank based Multi-Armed Bandit). It was evaluated with real FMs and the results show that the proposed approach outperforms the traditional algorithms used in the literature, and that both selection methods present similar performance.

1. Introduction

A Software Product Line (SPL) is generally defined as a set of common products from a particular market segment or domain [1]. Such products share some features, which represent a functionality, or a system capability that is relevant and visible to the end user. The features can be common to all products derived from the SPL, but they can also be variable, being found only in some of them. To ease feature management, most SPL methodologies adopt the Feature Model (FM) [2] to represent all the SPL commonalities and variabilities. In such model, the features are represented in a hierarchical arrangement through a tree.

The FM is the customized representation of all the SPL products, and for this reason, it has been used by many testing approaches for the variability test of SPLs. In such kind of test, the goal is to check whether the products derived from the FM meet their requirements. However, the number of products that can be derived from the FM grows exponentially, according to the number of features, and to test all products may be infeasible in practice [3]. Then, some works in the literature propose the use of a test criterion to select the best products. Some criteria are based on combinatorial testing [4], and require that products which include some combinations of features are tested. For example, the pairwise testing requires that all the pairs of features are included in at least one selected product [3,5–7]. Other criteria, such as mutation testing [8,9], are based on common faults that can be present in the FM and require that the selected products are capable to reveal such faults. Another factor to be considered is related to cost. For example, it is necessary to derive a small set of products, by eliminating those ones that can be redundant with respect to the criterion coverage.

We can observe that deriving a set of products for the variability test of FMs is an optimization problem, impacted by many factors, which has been successfully addressed in the literature by multi-objective approaches, in the field named Search-Based Software Engineering (SBSE) [10,11]. Among them we can mention the work of Lopez-Herrejon et al. [12] that proposes an approach to reach high pairwise coverage with low number of test cases, and the work of Matnei Filho and Vergilio [13] that has as objective to satisfy the mutation testing of FM with a minimum number of test cases. In such cases, algorithms such as Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [14], Strength Pareto Evolutionary Algorithm (SPEA2) [15], and Indicator-Based Evolutionary Algorithm (IBEA) [16] are successfully used.

However, to implement a search-based solution to this and other similar problems is not an easy task. This happens because most existing solutions are generally domain dependent and in many cases the tester does not have a deep knowledge in the optimization field. Existing solutions are generally domain dependent. Then, the tester needs to choose the operators (mutation and

* Corresponding author.
E-mail addresses: astricker@inf.ufpr.br (A. Strickler), japlima@inf.ufpr.br (J.A. Prado Lima), silvia@inf.ufpr.br (S.R. Vergilio), aurora@inf.ufpr.br (A.T.R. Pozo).

http://dx.doi.org/10.1016/j.asoc.2016.07.059
1568-4946/

crossover), to configure corresponding parameters, and so on. To help in this task, some authors have used hyper-heuristics. A hyper-heuristic (HH) is defined by Burke et al. [17] as a methodology that automates the design and configuration (tuning) of heuristic algorithms to solve computationally hard problems. It can be used to automatically determine which operator should be applied in the optimization problem, at a given moment.

In the literature, we can find few works that use hyper-heuristics for solving software engineering problems [18–21]. However, such works do not address variability test of SPLs.

Considering the benefits for the tester, this paper proposes and evaluates a HH approach for deriving a test set of products from the FMs. The approach dynamically (on-line) selects evolutionary operators, herein called Low Level Heuristics (LLHs), to be applied during the execution of a Multi-Objective and Evolutionary Algorithm (MOEA). Specifically, a LLH is a combination of crossover and mutation.

The hypothesis is that the use of HH leads to better results than the traditional algorithms. The proposed approach works with two different HH algorithms: one with a random selection, chosen as a baseline, and a second one, based on Fitness Rate Rank based Multi-Armed Bandit (FRR-MAB) [22]. FRR-MAB was chosen due to its performance reported in [23]. FRR-MAB is an Upper Confidence Bound (UCB) [24] based algorithm. UCB algorithms are among the best ones to deal with Multi-Armed Bandit (MAB) problems. In MAB context, as well as in HH, the key for a succeed algorithm is to find a good trade-off between exploration and exploitation, in other words, to decide which strategy to apply: the best LLH (exploitation), or other LLH (exploration), since the performance of a LLH changes at different search stages.

We evaluated our hypothesis by implementing the HH approach and the traditional algorithms mentioned above, and by using bases associated to real FMs. Multi-objective quality indicators, such as hypervolume and effect size, are used. Evaluation results are presented and show that the HH, independently of the adopted selection method, outperforms the best traditional algorithm, and that the random selection can obtain results as good as FRR-MAB.

The paper is organized as follows. Section 2 reviews related work on HH and MAB, and applications in the SBSE field. Section 3 contains basic concepts about FM and criteria used in this work: pairwise and mutation testing. Section 4 reviews the area of hyper-heuristics and selection methods. Section 5 introduces the proposed HH approach: population representation, fitness used, and implementation aspects of the evolution process. Section 6 describes how the experiments were conducted: research questions, used FMs, quality indicators and parameters configuration. Section 7 presents and analyses the obtained results. Finally, Section 8 concludes the work and presents future research directions.

2. Related work

SBSE is a field that could benefit from hyper-heuristics, and some authors have pointed hyper-heuristics as a trend and future research topic for SBSE [25–27]. However, very few studies explore hyper-heuristics for solving software engineering problems. In fact, as far as we know, only four works deal with these subjects [18,20,21,28].

Kumari et al. [28] proposed a multi-objective algorithm called *Fast Multi-objective Hyper-heuristic Genetic Algorithm* (MHypGA) to solve the module clustering problem. The proposed HH selects low-level heuristics while the optimization is being executed. Each low-level heuristic is composed by a selection operator, a mutation operator and a crossover operator. The authors empirically evaluated MHypGA in six real-world problems. MHypGA outperformed a conventional evolutionary algorithm in all problems.

Basgalupp et al. [18] applied an offline HH to evolve an algorithm for the generation of effort-prediction decision trees. The authors concluded that the algorithm created by their HH was able to obtain better results than some state-of-the-art algorithms and other traditional heuristics.

Jia et al. [21] proposed a single online HH algorithm to intelligently learn and apply combinatorial interaction testing strategies. The goal is to obtain better solutions and to provide an algorithm more generally applicable. Their experimental evaluation compares the results of their HH approach to the results of state-of-the-art techniques and to the best known results of the literature. The authors concluded that their HH performed well on constrained and unconstrained problems in several instances.

The work of Guizzo et al. [20] addresses the integration and test order problem, using the NSGA-II with hyper-heuristics. The HH obtained best results in comparison to a conventional NSGA-II. Its work used two selection functions: Choice Function (CF) and Multi-Armed Bandit (MAB) in a pull of 9 combined operators (3 of mutation and 3 of crossover). In their study, a variation of the original MAB method which is called *Sliding Multi-Armed Bandit (SlMAB)* proposed in [22] was used.

Recently, Li et al. [29] have proposed the integration of a MAB algorithm, called FRR-MAB with Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D). The authors reported results with different experiments using continuous multi-objective benchmark functions. These results point out that FRR-MAB MOEA/D was the most competitive algorithm.

These works motivated the study herein proposed: hyper-heuristics are a promising technique that can help SBSE researches; FRR-MAB is a state of art MAB. Moreover, in their work, Li et al. stated that it was not easy to quantify the improvement caused by an operator in most Pareto dominance-based MOEAs, the use of MAB in these algorithms posed a big challenge. Therefore, our work is based on the approach of Guizzo et al. to quantify such improvement.

3. Deriving test products from the FM

The FM is the most common representation for the features (variabilities and commonalities) of an SPL, allowing a hierarchical order of features in a tree, as shown in Fig. 1. Such figure contains the FM of the SPL AGM [30] from the game domain.

The mandatory features are represented by a full circle. They should be obligatorily present to derive a product. For example, the features `play` and `pause` in the sub-tree below the feature `services` of Fig. 1 are mandatory. The optional features are represented by an empty circle and may not be present in a product, such as the feature `save`. The group of exclusive alternative features is represented by interconnected edges and connected by an empty bow. From this group, only one sub-feature can be selected to compose a product. If it is possible to select more than one feature, this is represented by interconnected edges and connected by a full bow. Hence, we can see that different products can be generated by selecting different combination of features. In addition to this, there may be some dependence relations between the features. If a feature A is present in a product p, a feature B should also be present, or otherwise, B should not be included in p.

With the growing adoption of SPLs in the industry, we have noticed a demand for specific SPL testing techniques. An important activity in this context is the feature testing. This activity tests if the products that can be derived from a FM match their requirements. To ensure this, all products should be tested. However, this is impractical in terms of resources and time of execution [3]. Consequently, a way to select only the representative products is necessary, i.e., a testing criteria should be used. A testing criteria

Please cite this article in press as: A. Strickler, et al., Deriving products for variability test of Feature Models with a hyper-heuristic approach, Appl. Soft Comput. J. (2016), http://dx.doi.org/10.1016/j.asoc.2016.07.059