ARTICLE IN PRESS

JID: INFSOF [m5G;August 28, 2017;14:45]

Information and Software Technology 000 (2017) 1–15

Contents lists available at ScienceDirect

Information and Software Technology

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

Is seeding a good strategy in multi-objective feature selection when feature models evolve?

Takfarinas Saber[a,*], David Brevet[b], Goetz Botterweck[c], Anthony Ventresque[a,*]

[a] Lero@UCD, School of Computer Science, University College Dublin, Dublin 4, Ireland
[b] Institut Supérieur d'Informatique, de Modélisation et de leurs Applications, Clermont-Ferrand, France
[c] Lero@UL, University of Limerick, Ireland

A R T I C L E I N F O

Article history:
Received 7 February 2017
Revised 20 August 2017
Accepted 21 August 2017
Available online xxx

Keywords:
Software product lines
Multi-objective
Genetic algorithm
Evolution

A B S T R A C T

Context: When software architects or engineers are given a list of all the features and their interactions (i.e., a Feature Model or FM) together with stakeholders' preferences – their task is to find a set of potential products to suggest the decision makers. *Software Product Lines Engineering* (SPLE) consists in optimising those large and highly constrained search spaces according to multiple objectives reflecting the preference of the different stakeholders. SPLE is known to be extremely skill- and labour-intensive and it has been a popular topic of research in the past years.

Objective: This paper presents the first thorough description and evaluation of the related problem of *evolving* software product lines. While change and evolution of software systems is the common case in the industry, to the best of our knowledge this element has been overlooked in the literature. In particular, we evaluate whether seeding previous solutions to genetic algorithms (that work well on the general problem) would help them to find better/faster solutions.

Method: We describe in this paper a benchmark of large scale evolving FMs, consisting of 5 popular FMs and their evolutions – synthetically generated following an experimental study of FM evolution. We then study the performance of a state-of-the-art algorithm for multi-objective FM selection (SATIBEA) when seeded with former solutions.

Results: Our experiments show that we can improve both the execution time and the quality of SATIBEA by feeding it with previous configurations. In particular, SATIBEA with seeds proves to converge an order of magnitude faster than SATIBEA alone.

Conclusion: We show in this paper that evolution of FMs is not a trivial task and that seeding previous solutions can be used as a first step in the optimisation - unless the difference between former and current FMs is high, where seeding has a limited impact.

1. Introduction

Software Product Lines (SPL) is a branch of Software Engineering that aims at designing software products based on a composition of pre-defined software artefacts, increasing the reusability and personalisation of software products [3,42]. Software architects, when they design new products or adapt existing products, navigate a set of features in a Feature Model (FM). Each of these features represents an element of a software artefact that is of importance to some stakeholders. Through its structure and additional constraints, each FM describes all possible products as combinations of features. One of the issues with FMs is that they can be very large – for instance in our study we work with FMs composed of more than 13,000 features and of nearly 300,000 constraints. *Optimising FM Configurations*, i.e., *selecting the set of features* that could lead to potential real products, is then a difficult problem [31]. This problem is also called SPL configuration as it consists in configuring products from the FMs.

In theory, software architects use SPL engineering to find one product – the product that matches their needs the most and does not violate any of the Feature Model's constraints. But in practice, the notion of the 'best' product is controversial, as there are different perspectives on what is a good product. For instance, some stakeholders may consider that energy consumption of the products is the most important objective to optimise, while for others it can be the cost of licensing the features; or some stakeholders see the reliability as the key element (for instance if they run critical

* Corresponding authors.
E-mail addresses: takfarinas.saber@ucdconnect.ie (T. Saber), david.brevet@isima.fr (D. Brevet), goetz.botterweck@lero.ie (G. Botterweck), anthony.ventresque@ucd.ie (A. Ventresque).

http://dx.doi.org/10.1016/j.infsof.2017.08.010
0950-5849/

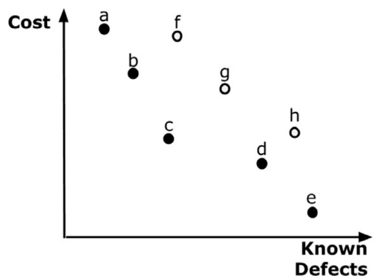

Fig. 1. Possible products in 2-dimensions. In black are the non-dominated solutions (the good ones) and in white the dominated solutions (the bad ones).

applications), while other stakeholders have a strict performance policy and they need assurance that the selected features follow some guidelines. Fig. 1 shows an example of SPL configuration according to two dimensions: number of known defects and cost (the lower the better for both dimensions). Possible products, found by an SPL optimisation algorithm, are represented as coloured circles. The good products, i.e., those that are better than any other one in a combination of objectives are represented by black circles. Product f, for instance, is not considered a good one, as product b is better than f in both dimensions. Similarly, product a, while worse than f in terms of cost, is better than all the others in terms of known defects - it is then considered a good product. Those good products form a set, called *Pareto set* or *Pareto front*.

Since it is unlikely in practice that only one dimension would be considered when optimising the feature configuration, the SPL engineering problem can be interpreted as a *multi-objective optimisation problem* [33,48]. In fact, software architects tend to favour tools that allow them to manipulate good products, i.e., possible products that are better than every other possible product on a particular combination of objectives.

Another related problem that has only been addressed in the literature recently [13], is *feature selection in a multi-objective context when the FMs evolve*. Software requirements and artefacts evolve constantly; customers and other stakeholders change their opinions about what an application should do and how it should achieve that. Such changes can be reflected in Feature Models [44]: for instance, we have seen in our study that a large FM (such as the one behind the Linux kernel) evolves regularly and substantially (every few months a new version is released with up to 7% difference from the previous one). In this context, it seems odd to generate random bootstrapping populations for the state-of-the-art genetic algorithms, such as SATIBEA. It is tempting on the contrary to use the fact that FMs have evolved and that the SPL configurations generated previously, while not totally applicable, are close and can be adapted. This is a strategy called *seeding* and our intuition is that this could prove helpful in the context of Feature Model selection - especially since SATIBEA (and the other evolutionary algorithms) is very dependent on the initial population.

Seeding for search-based software engineering is not a novel idea as such (e.g., see papers by Fraser and Arcuri [26] and Alshahwan and Harman [2]). However, our approach is novel for various reasons:

- usually seeding is done by taking a few good/previous solutions that are inserted in the initial population - while in this paper, we take all the previous solutions that we adapt to create a starting population.
- the data sets we use for our experiments in this paper are large and very constrained, which is not always the case in search-based software engineering contexts for which seeding is known to work. This, of course, calls for a proper evaluation that we report here.
- we also studied the performance of seeding against a large variety of data sets, of different size and demographics (varying in their numbers and ratios of features and constraints). This gives us some more assurance that our conclusions are correct.

From a more general perspective, our contributions in this paper are the following:

- We propose a benchmark[1] for the analysis of evolving SPL; this data set has been generated following a study of the demographics and evolution of a large SPL (Linux kernel). This data set is important to provide a good evaluation of the different algorithms under different evolution scenarios;
- We propose *eSATIBEA* which is a modification of the state-of-the-art SATIBEA [33] for evolving SPL. eSATIBEA adapts previous solutions to new FMs to improve and speed-up the results of SATIBEA;
- We evaluate SATIBEA and eSATIBEA on the evolving SPL problem and show that eSATIBEA improves both the execution time and the quality of SATIBEA. In particular, eSATIBEA converges an order of magnitude faster than SATIBEA alone.

The rest of this paper is organised as follows: Section 2 defines the problem of multi-objective features selection when Feature Models evolve. Section 3 presents a large study of the evolution of a Feature Model: 20 versions of the Linux kernel (up to 13,000+ features and nearly 300,000 constraints). This study of the demographics of the Feature Model helps us to create the synthetic evolutions of 5 large and popular FMs. In particular, we are able to create evolved data sets using two parameters representing the evolution in terms of features and constraints. Section 4 describes SATIBEA, the state-of-the-art resolution algorithm for multi-objective SPL problems and the seeding mechanism for SPL configuration. In particular, we present a modification of SATIBEA that we call eSATIBEA – for SATIBEA in the context of Evolution. Section 5 describes the hardware set-up and presents the various metrics we use to compare algorithms. Those metrics are standard in the community and are classified as quality and diversity metrics. Section 6 evaluates SATIBEA and eSATIBEA against the 5 evolved data sets – and with different degrees of evolution. We show that eSATIBEA performs better in terms of quality and converges faster than SATIBEA (an order of magnitude faster). Section 7 presents threats to the validity of the results. Section 8 describes the related work. Section 9 concludes our study and proposes some future directions that we would like to explore.

Note that the study that we report in this paper follows a previous work [13] published at SSBSE 2016, the symposium dedicated to Search Based Software Engineering. In the SSBSE paper, we introduced the problem of optimisation of evolving Feature Models and provided some preliminary results using one data set and one metric. In the current paper, we extend the study to 5 data sets and 5 metrics, and we describe the data sets and the techniques in depth.

2. Problem definition

In this section, we present the three elements that define the problem in our paper.

- Software Product Line Engineering, in particular how to describe variations of software applications as configurations of a Feature Model.
- Multi-objective optimisation; picking features can lead to many products for which the quality can be seen from different per-

[1] Available here: http://hibernia.ucd.ie/EvolvingFMs/ upon acceptance of this paper.