A Hitchhiker's Guide to Search-Based Software Engineering for Software Product Lines

Roberto E. Lopez-Herrejon
Johannes Kepler University
Linz, Austria
rlopez@jku.at

Javier Ferrer
Universidad de Málaga,
Andalucía Tech, Spain
ferrer@lcc.uma.es

Francisco Chicano
Universidad de Málaga,
Andalucía Tech, Spain
chicano@lcc.uma.es

Lukas Linsbauer
Johannes Kepler University
Linz, Austria
lukas.linsbauer@jku.at

Alexander Egyed
Johannes Kepler University
Linz, Austria
alexander.egyed@jku.at

Enrique Alba
Universidad de Málaga,
Andalucía Tech, Spain
eat@lcc.uma.es

ABSTRACT

Search Based Software Engineering (SBSE) is an emerging discipline that focuses on the application of search-based optimization techniques to software engineering problems. The capacity of SBSE techniques to tackle problems involving large search spaces make their application attractive for Software Product Lines (SPLs). In recent years, several publications have appeared that apply SBSE techniques to SPL problems. In this paper, we present the results of a systematic mapping study of such publications. We identified the stages of the SPL life cycle where SBSE techniques have been used, what case studies have been employed and how they have been analysed. This mapping study revealed potential venues for further research as well as common misunderstanding and pitfalls when applying SBSE techniques that we address by providing a guideline for researchers and practitioners interested in exploiting these techniques.

Keywords

Software Product Lines, Search Based Software Engineering, Systematic Mapping Study

1. INTRODUCTION

Search Based Software Engineering (SBSE) is an emerging discipline that focuses on the application of search-based optimization techniques to software engineering problems [12]. Among the techniques SBSE relies on are: evolutionary computation techniques[1](e.g. genetic algorithms) and basic local searches (e.g. hill climbing, simulated annealing or random search) [18]. These techniques are generic, robust, and have been shown to scale to large search spaces. These capacities make their application attractive for *Software Product Line (SPL)* problems.

In recent years, several publications have appeared that explored applications of SBSE techniques to concrete SPL problems. This fact prompted us to carry out a systematic mapping study to provide an overview of this research area [20]. Our general goal is to identify the quantity and the type of research and results available, and thus highlight possible open research problems and opportunities. The research questions our study addresses are:

- ***RQ1. In what phases of the SPL life cycle have SBSE techniques been used?.*** SBSE has been applied throughout the entire life cycle of single systems, so our interest is finding out if SBSE has or can be applied also throughout the entire life cycle of SPLs.
- ***RQ2. What SBSE techniques have been used?.*** There are a vast number of SBSE techniques available in literature. Our goal here is cataloguing their use for SPLs problems and analyse if there are common trends in their application.
- ***RQ3. What type of comparative analysis is used?.*** Search-based techniques commonly rely on randomness, so adequate statistical analysis is necessary for the results to be useful and meaningful. Here our objective is to gauge at the adequacy of this type of analysis depending on the techniques and problems used.
- ***RQ4. What evaluation case studies are used?.*** Here our focus is on cataloguing the type, number, and provenance of the case studies analysed. We believe that identifying common case studies and their sources could lead to establishing community-wide benchmarks for certain problems.

Our study corroborated the increasing interest in applying SBSE techniques in SPLs. We found that the most common application is testing at the Domain Engineering level, and the most common technique being genetic algorithms with an increasing interest in multi-objective optimization. We identified a need to improve the empirical evaluation with a more adequate statistical analysis, and some common pitfalls when dealing with multi-objective optimization algorithms, and provide a short guideline to address them.

[1]Evolutionary computation is an area of computer science, artificial intelligence more concretely, that studies algorithms that follow Darwinian principles of evolution [9].

2. SYSTEMATIC MAPPING STUDY

Evidence-Based Software Engineering (EBSE) is an emerging software engineering area whose goal is *"to provide the means by which current best evidence from research can be integrated with practical experience and human values in the decision making process regarding the development and maintenance of software"* [14]. One of the approaches advocated by EBSE is systematic mapping studies whose goal is to provide an overview of the results available within an area by categorizing them along criteria such as type, forum, frequency, etc. [20]. In this section we describe how we carried out the standard systematic mapping process and summarized the results we obtained.

2.1 Process

We carried out the standard five steps of systematic mapping studies as described by Petersen et al. [20].

Step 1. Definition of Research Questions. These are the questions we put forward in Section 1. In short, our driving goal was to find out in what parts of the SPL life cycle which SBSE techniques have been used and to gauge the adequacy of the analysis performed and the types and provenance of case studies employed.

Step 2. Conduct Search for Primary Studies. As a first step in our search we selected the search terms and categorized them in SPL and SBSE terms. Table 1 shows the list of terms used. We should point out the search terms we used for SBSE are an extended version of the ones employed by Harman et al. in a recent survey of SBSE [12].

To perform our search we proceeded in two stages. At the first stage, we performed queries in two specialized repositories, the Search Based Software Engineering Repository[2], and the Bibliography on Genetic Programming[3]. At the second stage, we relied on general search engines: ScienceDirect, IEEExplore, ACM Digital Libray, SpringerLink, and Google Scholar. The queries we performed took all the combinations of one term from the SPL list and one or more terms of the SBSE terms depending on the querying functionality of the search engines or repository. For example, the following is a query used in the IEEExplore engine:

```
("product line") AND ("search-based" OR "search
based" OR "optimization"  OR  "multi-objective
optimization" OR "multiobjective optimization"
OR "genetic algorithm" OR "GA" OR  "genetic
programming" OR "GP" OR "hill climbing" OR
"simulated annealing")
```

In addition, because SBSE is considered to have started with the seminal paper by Harman and Jones in 2001 [11], we trimmed our search to include only publications on or after that year. We performed the pertinent queries from March 12th to March 31st, 2014. The queries yielded a total of 1,326 hits that we sieved as described next.

Step 3. Screening of Papers for Inclusion and Exclusion. This step was straightforward. We looked for the search terms it the title, abstract and keywords and whenever necessary at the introduction or at other places of the paper. The sole criteria for inclusion in our mapping study was that a clear application of SBSE techniques to SPL was described. This resulted in a total of 42 articles whose details are summarized in Table 2, presented in the order they were found. We should point out that many of the hits were either applications of SBSE techniques to single software systems (i.e. not in the realm of SPLs), or for product lines in the general manufacturing or marketing sense but not for software.

SPL terms: product line, software family, feature model, variability, variant, commonality, variability-intensive system, highly-configurable system
SBSE terms: search based, optimization, genetic algorithm (GA), multiobjective optimization, multi-objective optimization, genetic programming (GP), hill climbing, simulated annealing, local search, integer programming, ant colony optimization (ACO), particle swarm optimization (PSO), scattered search, artificial immune systems (AIS), evolutionary algorithm, evolutionary strategy, greedy, greedy search, memetic algorithm, evolutionary programming, grammatical evolution, variable neighborhood search, iterative local search (ILS), GRASP, tabu search, path relinking, harmony search, imperial competitive algorithm, bee colony, fire fly, constraint handling, mutation testing

Table 1: Summary of SPL and SBSE Search Terms

Step 4. Paper Classification. For the classification of the stages of the SPL life cycle we used Pohl et al.'s SPL engineering framework (see [21]), which defines four subprocesses for both *Domain Engineering (DE)* and *Application Engineering (AE)*. We regard each sub-process as a stage. In addition to the eight stages of this framework, we considered two more classifications categories: one to cover all maintenance and evolution issues of SPLs, and one to contain publications that have tooling support as one of their main contributions. In summary, our classification terms are as follows and we will refer to them henceforth by their shorthand names in parenthesis:

- *Domain Requirements Engineering (DRE)* is the subprocess of DE where the common and variable requirements of the product line are defined, documented in reusable requirements artefacts, and continuously managed.
- *Domain Design (DD)* is the sub-process of DE where a reference architecture for the entire software product line is developed.
- *Domain Realisation (DR)* is the sub-process of DE where the set of reusable components and interfaces of the product line is developed.
- *Domain Testing (DT)* is the sub-process of DE where evidence of defects in domain artefacts is uncovered and where reusable test artefacts for application testing are created.

[2] http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/repository.html
[3] http://liinwww.ira.uka.de/bibliography/Ai/genetic.programming.html

- *Application Requirements Engineering (ARE)* is the sub-process of AE dealing with the elicitation of stakeholder requirements, the creation of the application requirements specification, and the management of application requirements.

- *Application Design (AD)* is the sub-process of AE where the reference architecture is specialised into the application architecture.

- *Application Realisation (AR)* is the sub-process of AE where a single application is realised according to the application architecture by reusing domain realisation artefacts.

- *Application Testing (AT)* is the sub-process of AE where domain test artefacts are reused to uncover evidence of defects in the application.

- *Maintenance and Evolution (ME)* refers to all the maintenance and evolution of all the artefacts developed across the entire life cycle of SPLs. Reverse engineering artefacts or bug fixing are examples of activities that fall in this category.

- Tool support (TOOL) used to classify the publications that have tooling support as one of their main contributions (e.g. tool papers or demos).

Step 5. Data Extraction and Mapping Studies. To perform the data extraction of our mapping study, the authors formed three distinct groups: one with more expertise in SBSE, one with more expertise in SPLs, and one with mixed background. Each group independently filled out a spreadsheet with the following data: *i)* SPL Stage (as defined in Step 4), *ii)* artefacts employed, *iii)* rationale for the categorization if any, *iv)* SBSE techniques, *v)* analysis performed (e.g. statistical test), *vi)* number of case studies evaluated, *vii)* type of case studies (e.g. artefact used), *viii)* provenance of the case studies, and *ix)* a general field for any remarks.

Once the data was independently gathered, it was consolidated during a joint revision session of the three groups where the data of each article was discussed until a consensus was reached. A summary of the results obtained are shown in Table 2. It should be pointed out that it is customary in SBSE articles to compare techniques against local or random searches. For our categorization, we report only the main SBSE technique put forward by each paper, regardless of other techniques used for comparison purposes.

2.2 Results

The first interesting result of our mapping study is the growth in number of publications as shown in Figure 1. From 2007 to 2009 the growth was steady. Then a sharp increase followed from 2010 to 2013 where each year almost doubled the number of publications of the previous year. There are early signs that an increasing trend will also continue in 2014.

Regarding the publication fora, we can derive from Table 2 that the most popular venue was the ICSE conference with 7 publications, considering short and workshop papers. A close second was the SPLC conference with 6 publications. On third place there was a tie with 3 publications between

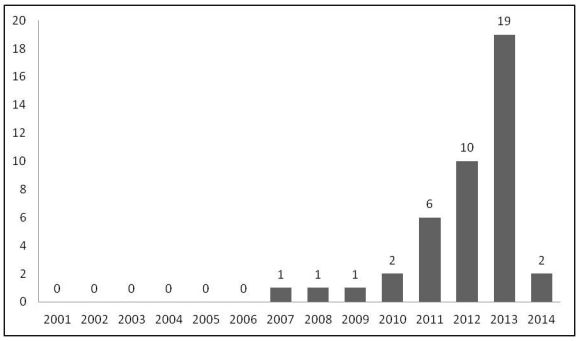
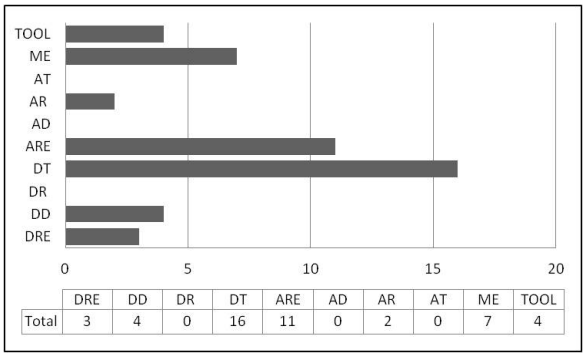


Figure 1: Publications per year since 2001

VaMoS workshop and the Symposium on Search-Based Software Engineering (SSBSE). On fourth place it was Computing Research Repository (CoRR) with 2 papers. The rest of the publications were distributed across 21 distinct venues from journals and conferences to a book chapter (BC) and a technical report (TR).

2.2.1 Results RQ1 – SPL Life Cycle Stages

Figure 2 shows the number of publications classified as described in Section 2.1. The most frequent stage where SBSE techniques are used is DT, testing at Domain Engineering level, with 16 publications. The majority of these publications focus on computing test suites that cover certain types (e.g. 2-wise or 3-wise) of features combinations that are derived from feature models. The second place was ARE, requirements engineering in the Application Engineering phase, with 11 publications. Many of these publications dealt with optimizing product configuration or derivation with different characteristics and attributes. Category ME, maintenance and evolution applications, came third place with 7 publications. Among the applications were reverse-engineering and fixing inconsistencies (e.g. of feature models). In fourth place with 4 publications there were TOOL and DD. For the former the tools supported analysis and generation of feature models and visualization, whereas for the latter the focus was on software architecture. In fifth place with 3 publications was DRE where the main focus was on the correct definition of variability models. Lastly with 2 publications was AR where the main concern was on runtime adaptation.



	DRE	DD	DR	DT	ARE	AD	AR	AT	ME	TOOL
Total	3	4	0	16	11	0	2	0	7	4

Figure 2: SPL Life Cycle Use

ID	Authors	Forum	Year	Technique	Stage	Analysis	Studies
[S36]	Wang, Shaukat, Gotlieb	GECCO	2013	GA	DT	Stat.Tests	6,FM[1,5]
[S4]	Colanzi, Vergilio	SSBSE	2012	MOEA	DD	None	1,CD[4]
[S23]	Lopez-Herrejon, Galindo, Benavides, Segura, Egyed	SSBSE	2012	GA	ME	Basic	59,FM[1]
[S29]	Sayyad, Menzies, Ammar	ICSE	2013	MOEA	ARE	Basic	2,FM[1]
[S3]	Colanzi	ICSE	2012	MOEA	DD	None	None
[S9]	Guo, White, Wang, Li, Wang	JSS	2011	GA	ARE	Basic	900,FM[2]
[S21]	Lopez-Herrejon, Egyed	SSBSE	2011	DS	ME	Basic	60,FM[1]
[S37]	Wu, Tang, Kwong, Chan	JISTDM	2011	IP	DRE	Undefined	1,AH[5]
[S2]	Cohen, Dwyer, Shi	TSE	2008	GRE	DT	Undefined	4,AH[3]
[S35]	Ullah	TR	2009	GA	ME	Undefined	1,AH[3]
[S8]	Garvin, Cohen, Dwyer	ESE	2011	SA	DT	Stat.Tests	35,AH[2,3]
[S1]	Cohen, Dwyer, Shi	ISSTA	2007	GRE, SA	DT	Basic	2,AH[3]
[S31]	Segura, Parejo, Hierons, Benavides, Ruiz-Cortés	ESA	2014	GA	TOOL	Stat.Tests	5000,FM[2]
[S15]	Henard, Papadakis, Perrouin, Klein, Le Traon	SPLC	2013	GA	TOOL, DT	None	None
[S11]	Haslinger, Lopez-Herrejon, Egyed	VaMoS	2013	GRE	DT	Basic	146,FM[1]
[S22]	Lopez-Herrejon, Egyed	VaMoS	2012	DS	ME	Basic	45,FM[1]
[S39]	Xu, Cohen, Motycka, Rothermel	SPLC	2013	GA	DT	Stat.Tests	2,FM[4]
[S14]	Henard, Papadakis, Perrouin, Klein, Le Traon	SPLC	2013	GA	DT	Stat.Tests	8,FM[1]
[S16]	Henard, Papadakis, Perrouin, Klein, Le Traon	ICSE	2013	LS	ME	Basic	1,FM[1]
[S34]	Tan, Lin, Ye, Zhang	ACSC	2013	SA, GA	ARE	Basic	1,FM[4]
[S25]	Pascual, Pinto, Fuentes	SEAMS	2013	GA	ARE, AR	Basic	1,AH[4]
[S32]	Serajzadeh, Shams	SPLC	2011	PSO	DRE	None	1,AH[4]
[S17]	Johansen, Haugen, Fleurey	SPLC	2012	GRE	DT	Basic	19,FM[1,4]
[S18]	Johansen, Haugen, Fleurey, Eldegard, Syversen	MODELS	2012	GRE	DT	None	2,FM[3,5]
[S24]	Murashkin, Antkiewicz, Rayside, Czarnecki	SPLC	2013	AEMOO	TOOL, ARE	None	None
[S5]	Colanzi, Vergilio	ICSE	2013	MOEA	DD	None	None
[S20]	Lopez-Herrejon, Chicano, Ferrer, Egyed, Alba	ICSM	2013	AEMOO	DT	Basic	118,FM[1]
[S38]	Wu, Tang, Wang	SEDM	2010	ADHS	DD, ARE	Undefined	None
[S6]	Cruz,Neto, Britto, Rabelo, Ayala, Soares, Mota	CEC	2013	MOEA	ARE	Undefined	1,AH[4]
[S28]	Sayyad, Ingram, Menzies, Ammar	ASE	2013	MOEA	ARE	Stat.Tests	7,FM[1,4]
[S19]	Karimpour, Ruhe	ICSE	2013	MOEA	ME	Undefined	1,FM[1]
[S27]	Sayyad, Ingram, Menzies, Ammar	ICSE	2013	MOEA	ARE	Stat.Tests	1,FM[1]
[S26]	Sanchez, Moisan, Rigault	ICSE	2013	DS	ARE, AR	None	100,FM[2]
[S33]	Shi, Guo, Wang	PIC	2010	GRE	ARE	Basic	200,FM[2]
[S41]	Yu,Duan, Lei, Kacker, Kuhn	HASE	2014	CH	DT	Basic	12,FM[1]
[S40]	Yi, Zhang, Zhao, Jin, Mei	RE	2012	GA	ME	Basic	2,FM[1]
[S7]	Ensan, Bagheri, Gasevic	CAiSE	2012	GA	DT	Basic	8,FM[1]
[S42]	Zhang, Haiyan, Mei	BC	2011	DS	DRE	Basic	32,FM[2]
[S13]	Henard, Papadakis, Perrouin, Klein, Le Traon	ICST	2013	LS	DT	Stat.Tests	12,FM[1,4]
[S30]	Segura, Galindo, Benavides, Parejo, Ruiz-Cortés	VaMoS	2012	GA	TOOL	None	None
[S12]	Henard, Papadakis, Perrouin, Klein, Le Traon	CoRR	2012	GA	DT	Basic	124,FM[1,5]
[S10]	Haslinger, Lopez-Herrejon, Egyed	CoRR	2013	GRE	DT	Stat.Tests	133,FM[1]

Study type: **FM** feature model, **CD** class diagram, **AH** ad hoc model
Provenance Superscripts: **1** SPLOT, **2** Random, **3** Open source project, **4** Academic, **5** Industrial

Table 2: Primary Sources Summary Table

Acronym	Description	No.
MOEA	Multi-Objective Evolutionary Algorithm	8
GA	Genetic Algorithm	14
LS	Local Search	2
IP	Integer Programming	1
GRE	Greedy algorithm	7
SA	Simulated Annealing	3
PSO	Particle Swarm Optimization	1
AEMOO	Ad hoc Exact Multi-Objective Optimization	2
ADHS	Ad hoc Heuristic Search	1
DS	Other Deterministic Search	4
CH	Constraint Handling	1

Table 3: SBSE techniques used and Frequency

2.2.2 Results RQ2 – SBSE techniques used

Table 3 summarizes what SBSE techniques were used and in how many publications. Not surprisingly genetic algorithms came first with 14 publications. We believe their popularity might be because they are among the most basic evolutionary algorithms. On second place there was the multi-objective evolutionary algorithms with 8 publications. We should also point out that we draw a distinction with other forms of multi-objective optimization besides evolutionary, in our study we found two ad hoc exact approaches (AEMOO). While performing the mapping study, we noticed in some publications a common misunderstanding in distinguishing between genetic algorithms with one fitness function (i.e. single-objective) and multi-objective algorithms. We address this finding in further detail in Section 3.

The third place was greedy algorithms with 7 publications. The fourth place with 4 publications was deterministic searches (e.g. based on breadth first search). The fifth place was simulated annealing with 3 publications, followed by local search with 2. The rest of the publications employed an array of different techniques.

2.2.3 Results RQ3. Type of Comparative Analysis

We divided the publications in four categories depending on the type of analysis performed: *i) Undefined* whenever we were not able to clearly discern what type of analysis was performed, *ii) None* when there was clearly no analysis presented, *iii) Basic* when some basic statistical tools were used (i.e. medians, average, standard deviation), and *iv) Stat.Tests* when any of the standard statistical analysis test (as sketched in Section 3) was used.

The category of *Basic* was the most frequent with 18 publications, followed by *None* and *Statistical Tests* with 9 publications each, and *Undefined* with 6. As mentioned before, because of the randomness involved in most of the SBSE techniques, using the adequate statistical analysis is of utmost importance for the results obtained to be reliable and meaningful. Our findings help to raise awareness of the need to employ adequate statistical analysis so that it could be addressed by the SPL community in large. To contribute to this effort, we outline in Section 3 a standard experimental methodology followed when using SBSE techniques.

2.2.4 Results RQ4. Evaluation Case Studies

We categorize the artefacts employed in the publications in four types: *i) None* when there was no clear artefact employed, *ii)* Class Diagrams (*CD*), *iii)* Feature Models (*FM*), and *iv)* Ad hoc (*AH*) when special format artefacts were used in particular cases.

For the provenance of the artefacts we use five categories: *i)* from the SPLOT repository, *ii)* when artefacts are generated randomly, *iii)* when the artefacts come from open source projects, *iv)* when the artefacts come from academia, and *v)* when the artefacts belong to actual industrial cases.

The trends are clear in both aspects. The great majority of artefacts used are feature models (28 publications), with ad hoc artefacts (that range from specialized constraint formats to cost models) in second place with 8 publications. In terms of provenance, SPLOT was the main source with 19 publications, followed by academic case studies with 9 publications. In third place was random generation with 6 publications, followed closely by open source projects (5 publications), and industrial cases (4 publications).

An interesting finding was the low number of feature models employed, in some cases as low as 1 or 2 feature models, which raises the issue of how generalizable the results of those studies are. Another finding is that the majority of publications in the area of domain testing (DT) employed SPLOT feature models. This suggests the possibility of extracting a common benchmark of feature models for assessing the different testing techniques.

2.3 Threats to validity

We faced similar validity threats to any other systematic mapping study. The selection of the search queries was carefully chosen to include common terms in both SPLs and SBSE. For the latter we used an extended the search terms already employed in a survey of SBSE techniques. In our search we employed two SBSE paper repositories and 5 standard bibliography search engines. For our classification terms, we employed as a basis a framework that is well-known in the SPL community. For the data extraction we performed three independent classifications by three groups with different background, which were subsequently validated and consolidated. Certainly, any of these aspects could be refined and improved to obtained a better snapshot of the application of SBSE to SPLs. We definitely intend to do so with the feedback of both communities. In addition we will expand our search by looking into the references of the papers already collected.

3. SBSE GUIDELINES

In this section we summarize the common pitfalls and misunderstandings in the use of SBSE techniques for SPL that were revealed by our mapping study. We put forward a set of basic guidelines that researchers interested in applying SBSE techniques can follow to avoid them.

3.1 Experimental Methodology

When one wants to compare the performance of a set of algorithms in a task there are roughly two options: provide theorems stating the relative performance of the algorithms or perform an empirical study. Theorems are desirable since they are more general and, provided that their hypotheses hold, the claim will be necessarily true. However, proving such theorems is very difficult (if not impossible) since they

require mathematical tools that are not developed or far from trivial. On the other hand, empirical studies can be more easily done, since they only require to run the algorithms over the case studies.

An empirical study is always incomplete and the conclusions we obtain from it can always be biased by, at least, the selection of the case studies, the parameters of the algorithms, the experiment design and the analysis of the data. It should be clear that the conclusions obtained could be wrong. However, it is possible to reduce the probability of claiming wrong conclusions. In particular, we will focus in this section on how to apply an appropriate experimental methodology and statistical analysis of the results to increase the confidence on the conclusions.

One of the first things we should think about when designing the empirical study is how many and what case studies we will use. If the number of case studies is too low (one or two) the conclusions cannot be generalized, there is not enough evidence to claim that the results will be the same if new case studies are added to the empirical study. A serious empirical study should contain a large number of case studies. The exact number will depend on the availability of case studies or the budget limit to do the experiments. We should also take into account the No Free Lunch Theorem [26], which claims that all the algorithms perform the same when all the problems are considered. An empirical study is focused on one single problem (not all of them), but even in this case the Focused No Free Lunch Theorems [25] suggest that we could find that all the algorithms perform the same as the number of case studies increases.

If stochastic algorithms are used in the experimental study then the performance measure of the algorithm for a case study is not a single number, but a probability distribution. The result of a single run of the algorithm contains little information about this distribution and several independent runs should be done. Arcuri and Briand [2] suggest 1,000 independent runs of the algorithm. There is, however, no reason to use exactly this number, it usually depends on the time limit for doing the experiments. The larger the number of independent runs, the higher the confidence we have on the results. In the case of deterministic algorithms, where the result of the performance measure does not change in different runs, one single execution of the algorithm is enough. If the performance measure is time, even a deterministic algorithm would require several runs, since the wall clock time depends on the load of the system in which the algorithm runs.

Once all the results of the different independent runs of the algorithms over the case studies are collected, a statistical analysis of the data is necessary. Common questions we want to answer in this phase are: Do the algorithms perform all the same?, Which is the best algorithm for this task? Data contain several values for the performance of the algorithms on the case studies (one for each independent run). Aggregating these values into a single one, e.g., the average, to compare the algorithms is not enough, since the aggregated value (whatever it is) is a random variable itself. For example, if the average execution time of algorithm A for case study X is lower than the average execution time of algorithm B we cannot claim that A is faster than B, since it could be a matter of chance. It is necessary to apply statistical tests to check if the observed differences are really significant or not. There are many statistical tests that can be applied depending on the kind of data and the assumptions that can be made on them. The work of Sheskin [23] is a good reference to find the appropriate test. In short, any statistical test formulates a base hypothesis, called the *null hypothesis* H_0, and it computes the probability of having the observed data provided that H_0 is true. This computed value is the so-called p-value. If the p-value is low enough we can safely reject the null hypothesis, meaning that probably it is not true. In the example of the execution time the null hypothesis could be H_0: "the average of the execution time of algorithms A and B over case study X is the same". Let's say that an appropriate statistical test (a t-test, for example) provides a p-value of 0.01. Then we can claim with significance level $\alpha = 1\%$ that the mean execution time of algorithm A over all the possible runs (potentially infinite) is lower than the mean execution time of algorithm B. The probability of failing in our conclusion, that is, making a type I error, is 0.01.

Statistical tests can be classified into parametric and non-parametric. The former assume that the data are samples of a given distribution (e.g., normal, binomial, etc.). One salient example of parametric tests is the Student t-test. This test assumes a normal distribution of the random variables and the same standard deviation in both populations. In general, the probability distribution of the data we collect is unknown, so it is difficult to justify the application of parametric tests. In other cases, the assumptions of the parametric tests are clearly violated. Taking again the example of the run time of the algorithms, the time does not follow a normal distribution because it cannot be negative, while a normal distribution would require a nonzero probability of having negative values. If the assumptions are violated or it is unknown if they are, a non-parametric test should be used. These tests do not assume a distribution of the data, so they can always be applied. A popular non-parametric test to compare two samples is the Mann-Whitney U-test, which checks of the median if the ranks of the samples are equal or not.

Care should be taken if many statistical tests are done during the statistical analysis. For example, it is common to compare many algorithms by applying two-samples tests for all the possible pairs of algorithms. In this case, the global p-value obtained of the experiment is higher than the p-values for each particular comparison. The probability of type I errors are accumulated. Thus, using a significance level of α in each pairwise comparison is not the same as having a significance level of α in the global experiment, the latter is higher, and this could affect the confidence on the conclusions. In order to bound the significance level of the global experiment by a known value, it is required to reduce the significance level of the pairwise comparisons. There are many proposals for this. One well-known approach is the Bonferroni correction [19].

3.2 Multi-objective Optimization

In our survey we found two main kinds of multi-objective techniques: the ones that use a mono-objective technique with a weighted sum of the objectives as fitness functions (e.g. [S14, S36]) and the ones that use multi-objective algorithm to find the entire Pareto front. The former technique has several drawbacks. If the weights of the aggregative function are not systematically varied during the optimization, a single trade-off solution is obtained. In addition,

even if the weights are changed during the search, it is not possible to obtain all the points in the Pareto front if it is concave downwoards [6]. The use of weighted sum of objectives could be useful if the preferences of all the objectives are clear. When this is not the case, multi-objective algorithms should be used, since they are able to obtain the Pareto front [4]. These algorithms provide an approximated Pareto front, which is a set of non-dominated solutions. One important issue is how to compare these fronts, since they are not just a single number. This is the role of quality indicators as explained next.

3.2.1 Quality Indicators

Three different issues are normally considered for assessing the quality of the results computed by a multi-objective optimization algorithm [27]: *i)* To minimize the distance of the computed solution set by the proposed algorithm to the optimal Pareto front (convergence towards the optimal Pareto front), *ii)* To maximize the spread of solutions found, so that we can have a distribution as smooth and uniform as possible (diversity), and *iii)* To maximize the number of elements of the Pareto optimal set found.

A number of quality indicators have been proposed in the literature trying to capture the three issues indicated above, but for the moment, there is not a single metric which captures all of them. Consequently, researchers should use more than one to measure different aspects of the solutions generated by the multi-objective techniques. Among them, we can distinguish between *Pareto compliant* and *non Pareto compliant* indicators [16]. Given two Pareto fronts, A and B, if A dominates B, the value of a Pareto compliant quality indicator is higher for A than for B; meanwhile, this condition is not fulfilled by the non–compliant indicators. Thus, the use of Pareto compliant indicators should be preferable. To apply these quality indicators, it is usually necessary to know the optimal Pareto front. However, the location of the optimal front is usually unknown. Therefore, the front composed of all the non–dominated solutions computed by all analyzed approaches is used to obtain a reference Pareto front. Many quality indicators have been proposed in the literature. Next we highlight the advantages and disadvantages of some of the most common ones:

Number of Pareto optimal solutions. This non–compliant indicator is very simple, it computes the number of solutions that are included in the optimal Pareto front. Its main advantage lies in the fact that it is very easy to compute. In contrast, the disadvantages are the lack of information about the diversity of solutions and the requirement of knowing the optimal Pareto front.

Hypervolume (HV) [28]. This Pareto–compliant indicator calculates the volume (in the objective space) covered by members of a non-dominated set of solutions. For each solution of the set, a hypercube is constructed with a reference point and the solution as the diagonal corners of the hypercube. The main advantages of the hypervolume are that it considers the convergence as well as the diversity of the solutions, and it doesn't require the optimal Pareto front. A drawback is that it depends on the reference point selected. Different reference points produce different results. This could be critical to compare the results with existing approaches in the literature. Therefore, it should be always reported what is the reference point used for computing the hypervolume.

Spread (SD) [8]. It is a diversity quality non–compliant indicator that measures the distribution of individuals over the non-dominated region. This measure is based on the distance between solutions, so Pareto fronts with a smaller value of Spread are more desirable. The main advantage of this measure is that it summarizes the diversity of a Pareto front in one single scalar value. The main disadvantage is that it does not consider the other two quality aspects, i.e., the solution set could be very well distributed, but the solutions could be far from the optimal Pareto front. Other quality indicators should be used to complement the Spread.

Generational Distance (GD) [24]. The generational distance is a non–compliant indicator. It measures how far the elements in the approximated Pareto front are from those in the optimal Pareto front. It considers the distance of the approximated Pareto front obtained to the reference front. Pareto fronts with a smaller value of GD are more desirable. The advantages of the generational distance are the ease of understanding and calculation, and the possibility to use different kinds of distance functions. In contrast, it does not take into account the diversity of the solutions found, i.e., a front with only one solution in the optimal Pareto front will obtain an ideal value of generational distance.

Epsilon (Multiplicative) [29]. This Pareto–compliant indicator measures, in one single scalar value, how badly approximated the worst approximated solution of the Pareto front is. The approximation quality of solutions is the ratio between the optimal value and the best value found. The main advantage of this quality indicator is that it allows us to compare the quality of solutions between different functions, different population sizes, and even different dimensions. In addition, it measures convergence of the algorithm, but it does not depend on a chosen reference point like the hypervolume. In contrast, its main disadvantage is that it only considers part of the front, namely the worst solution.

The previous indicators have the advantage of summarizing an entire front into one single scalar value that allows the performance of different algorithms to be compared. However, from the point of view of a decision maker, knowing about a single number is not enough, because it gives no information about the shape of the front.

In the related literature, the trade-off between the different objectives is usually presented by showing one of the approximated Pareto fronts obtained in one single run of a given algorithm. However, if the optimization algorithm used is stochastic there is no warranty that the same result is obtained after a new run of the algorithm. We need a way of representing the results of a multi-objective algorithm that allows us to observe the expected performance and its variability, in the same way as the average and the standard deviation are used in the single-objective case. For this reason, the concept of *Empirical Attainment Function (EAF)* [15] is used. In short, the EAF is a function α from the objective space $\mathbb{R}^n$ to the interval $[0, 1]$ that estimates for each vector in the objective space the probability of being dominated by the approximated Pareto front of one single run of the multi-objective algorithm. Given the r approximated Pareto fronts obtained in the different runs, the EAF is defined as:

$$\alpha(z) = \frac{1}{r} \sum_{i=1}^{r} I(A^i \preceq \{z\}) \quad (1)$$

where A^i is the i-th approximated Pareto front obtained

with the multi-objective algorithm and I is an indicator function that takes value 1 when the predicate inside it is true, and 0 otherwise. The predicate $A^i \preceq \{z\}$ means A^i dominates solution z. Thanks to the attainment function, it is possible to define the concept of k%-attainment surface [15]. The attainment function α is a scalar field in $\mathbb{R}^n$ and the k%-attainment surface is the level curve with value $k/100$ for α.

Informally, the 50%-attainment surface is analogous to the median in the single-objective case. In a similar way, the 25%- and 75%-attainment surfaces can be used as the first and third "quartile fronts" and the region between them could be considered a kind of "interquartile region".

The attainment surfaces provides engineers with a tool for evaluating the variability of an algorithm for the problem at hand. The variability in the results of one multi-objective algorithm is not reduced to a scalar (as in the single-objective case). This variability depends on the region observed in the objective space. We can find a rich range of possibilities when considering variability in the multi-objective domain. Using attainment surfaces the engineer can analyze and explore this range of possibilities. From a practical point of view, in the SPL domain, this tool helps the engineer to decide on the more suitable multi-objective algorithm for her/his requirements.

4. RELATED WORK

In this section we briefly summarize the salient surveys and studies carried out in either SPLs or in SBSE. The survey by Harman et al. presents a general overview of SBSE techniques and the areas where it has been employed [12]. Freitas et al. performed a bibliometric analysis of SBSE [7]. Their goal was to identify trends in the number of publications, the publication fora, the authorship and collaborations amongst members of the SBSE community. In contrast with our work, they have a different focus, namely general software engineering. Ali et al. performed a systematic review of empirical investigation of search-based test case generation techniques [1]. Similar to ours their review assessed how the focused techniques were empirically evaluated, but in contrast their work focused exclusively on testing and for non-SPL software systems.

In the area of SPL there are two recent systematic mapping studies in SPL testing [5, 10], both of which attest that the application of SBSE techniques for SPL testing is an area ripe for research that needs to be further explored. Laguna et al. performed a systematic mapping study on SPL evolution [17], where they made an assessment of the maturity level of techniques to migrate individual systems or groups of software variants into SPLs. Rabiser et al. performed a systematic review of requirements for supporting product configuration [22], whereas Holl et al. carried out a systematic review of the capabilities to support multi product lines [13]. Chen et al. performed a systematic review of variability management [3]. In contrast with all these SPLs studies our work has a novel and distinct focus.

5. CONCLUSIONS AND FUTURE WORK

In this paper we present the results of the first systematic mapping study on the application of SBSE techniques to SPL problems. Our study corroborates the increasing interest in applying this type of techniques as shown by the number of recent publications. The most common application is for testing at the Domain Engineering level, for example in computing test suites in combinatorial interaction testing. The most common technique used is genetic algorithms with an increasing interest in multi-objective optimization problems. We identify a need to improve empirical evaluations with a more adequate statistical analysis, and some common pitfalls when dealing with multi-objective optimization algorithms. To address these two issues we provide a short guideline section to serve as a reference basis for researchers and practitioners interested in exploiting SBSE techniques.

Our work also revealed research areas and possible opportunities. For example, we inadvertently found that there is a plethora of work on product line scoping and design in the area of manufacturing and marketing that relies on SBSE techniques. This begs the question if any of the research done in those areas can be applicable to SPLs. We found no applications in domain realisation, application design, and application testing. For the first two there is work in SBSE on the generation of software artefacts including architectural models that could be leveraged, the challenge is how to effectively express and cope with variability issues.

6. ACKNOWLEDGEMENTS

This research is partially funded by the Austrian Science Fund (FWF) projects P25289-N15, P25513-N15, and Lise Meitner Fellowship M1421-N15, the Spanish Ministry of Economy and Competitiveness and FEDER under contract TIN2011-28194 and fellowship BES-2012-055967. It is also partially founded by project 8.06/5.47.4142 in collaboration with the VSB-Technical University of Ostrava and Universidad de Málaga, Andalucía Tech.

7. References

[1] 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 Trans. Soft. Eng.*, 36(6):742–762, 2010.

[2] A. Arcuri and L. Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verif. and Reliab.*, 24:219–250, 2014.

[3] L. Chen and M. A. Babar. A systematic review of evaluation of variability management approaches in software product lines. *Inform. & Software Tech.*, 53(4):344–362, 2011.

[4] C. Coello Coello, G. B. Lamont, and D. A. Veldhuizen. *Evolutionary Algorithms for Solving Multi Objective Problems*. 2007.

[5] P. A. da Mota Silveira Neto, I. do 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 & Software Technology*, 53(5):407–423, 2011.

[6] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, Aug. 1997.

[7] F. G. de Freitas and J. T. de Souza. Ten years of search based software engineering: A bibliometric analysis. In *SSBSE*, pages 18–32, 2011.

[8] K. Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, Inc., Aug. 2001.

[9] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer Verlag, 2003.

[10] E. Engström and P. Runeson. Software product line testing - a systematic mapping study. *Inform. & Software Tech.*, 53(1):2–13, 2011.

[11] M. Harman and B. F. Jones. Search-based software engineering. *Inform. & Software Tech.*, 43(14):833–839, 2001.

[12] M. Harman, S. A. Mansouri, and Y. Zhang. Search-based software engineering: Trends, techniques and applications. *ACM Comput. Surv.*, 45(1):11, 2012.

[13] G. Holl, P. Grünbacher, and R. Rabiser. A systematic review and an expert survey on capabilities supporting multi product lines. *Inform. & Software Tech.*, 54(8):828–852, 2012.

[14] B. Kitchenham, T. Dybaa, and M. Jorgensen. Evidence-based software engineering. In *ICSE*, pages 273–281. IEEE CS Press, 2004.

[15] J. Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *ISDA*, pages 552–557. IEEE Computer Society, 2005.

[16] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. TIK Report 214, ETH Zurich, February 2006.

[17] M. A. Laguna and Y. Crespo. A systematic mapping study on software product line evolution: From legacy system reengineering to product line refactoring. *Sci. Comput. Program.*, 78(8):1010–1034, 2013.

[18] S. Luke. *Essentials of Metaheuristics*. Lulu, 2009. http://cs.gmu.edu/∼sean/book/metaheuristics/.

[19] S. Nakagawa. A farewell to bonferroni: the problems of low statistical power and publication bias. *Behavioral Ecology*, 15(6):1044–1045, 2004.

[20] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *EASE*, pages 68–77. British Computer Society, 2008.

[21] K. Pohl, G. Bockle, and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.

[22] R. Rabiser, P. Grünbacher, and D. Dhungana. Requirements for product derivation support: Results from a systematic literature review and an expert survey. *Inform. & Software Tech.*, 52(3):324–346, 2010.

[23] D. J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman & Hall/CRC; 4 edition, 2007.

[24] D. A. Van Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. PhD thesis, Air Force Institute of Technology, USA, 1999. AAI9928483.

[25] D. Whitley and J. Rowe. Focused no free lunch theorems. In *GECCO*, pages 811–818, 2008.

[26] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Trans. on Evol. Comp.*, 4:67–82, 1997.

[27] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–95, Jan. 2000.

[28] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans. Evol. Comp.*, 3(4):257–271, 1999.

[29] E. Zitzler, L. Thiele, M. Laumanns, F. C. M., and d. F. V. G. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evol. Comp.*, 7:117–132, 2003.

APPENDIX

A. Primary Sources Cites

[S1] M. B. Cohen, M. B. Dwyer, and J. Shi. Interaction testing of highly-configurable systems in the presence of constraints. In *ISSTA*, pages 129–139, 2007.

[S2] 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 Trans. Software Eng.*, 34(5):633–650, 2008.

[S3] T. E. Colanzi. Search based design of software product lines architectures. In *ICSE*, pages 1507–1510, 2012.

[S4] T. E. Colanzi and S. R. Vergilio. Applying search based optimization to software product line architectures: Lessons learned. In *Proceedings of SSBSE*, volume 7515 of *LNCS*, pages 259–266, 2012.

[S5] T. E. Colanzi and S. R. Vergilio. Representation of software product line architectures for search-based design. In *Proceedings of CMSBSE@ICSE*, pages 28–33, 2013.

[S6] J. Cruz, P. S. Neto, R. Britto, R. Rabelo, W. Ayala, T. Soares, and M. Mota. Toward a hybrid approach to generate software product line portfolios. In *IEEE Congress on Evolutionary Computation*, pages 2229–2236, 2013.

[S7] F. Ensan, E. Bagheri, and D. Gasevic. Evolutionary search-based test generation for software product line feature models. In *CAiSE*, pages 613–628, 2012.

[S8] B. J. Garvin, M. B. Cohen, and M. B. Dwyer. Evaluating improvements to a meta-heuristic search for constrained interaction testing. *Empirical Software Engineering*, 16(1):61–102, 2011.

[S9] J. Guo, J. White, G. Wang, J. Li, and Y. Wang. A genetic algorithm for optimized feature selection with resource constraints in software product lines. *Journal of Systems and Software*, 84(12):2208–2221, 2011.

[S10] E. N. Haslinger, R. E. Lopez-Herrejon, and A. Egyed. Improving casa runtime performance by exploiting basic feature model analysis. *CoRR*, abs/1311.7313, 2013.

[S11] E. N. Haslinger, R. E. Lopez-Herrejon, and A. Egyed. Using feature model knowledge to speed up the generation of covering arrays. In *VaMoS*, page 16, 2013.

[S12] C. Henard, M. Papadakis, G. Perrouin, J. Klein, P. Heymans, and Y. L. Traon. Bypassing the combinatorial explosion: Using similarity to generate and prioritize t-wise test suites for large software product lines. *CoRR*, abs/1211.5451, 2012.

[S13] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon. Assessing software product line testing

via model-based mutation: An application to similarity testing. In *ICST Workshops*, pages 188–197. IEEE, 2013.

[S14] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon. Multi-objective test generation for software product lines. In *Proceedings of SPLC*, pages 62–71, 2013.

[S15] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon. Pledge: a product line editor and test generation tool. In *SPLC Workshops*, pages 126–129, 2013.

[S16] C. Henard, M. Papadakis, G. Perrouin, J. Klein, and Y. L. Traon. Towards automated testing and fixing of re-engineered feature models. In *Proceedings of ICSE*, pages 1245–1248, 2013.

[S17] M. F. Johansen, Ø. Haugen, and F. Fleurey. An algorithm for generating t-wise covering arrays from large feature models. In *SPLC (1)*, pages 46–55, 2012.

[S18] M. F. Johansen, Ø. Haugen, F. Fleurey, A. G. Eldegard, and T. Syversen. Generating better partial covering arrays by modeling weights on sub-product lines. In *MoDELS*, pages 269–284, 2012.

[S19] R. Karimpour and G. Ruhe. Bi-criteria genetic search for adding new features into an existing product line. In *Proceedings of CMSBSE@ICSE*, pages 34–38, 2013.

[S20] 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. In *ICSM*, pages 404–407. IEEE, 2013.

[S21] R. E. Lopez-Herrejon and A. Egyed. Searching the variability space to fix model inconsistencies: A preliminary assessment. In *SSBSE 2011.*

[S22] R. E. Lopez-Herrejon and A. Egyed. Towards fixing inconsistencies in models with variability. In *Proceedings of VaMoS*, pages 93–100, 2012.

[S23] R. E. Lopez-Herrejon, J. A. Galindo, D. Benavides, S. Segura, and A. Egyed. Reverse engineering feature models with evolutionary algorithms: An exploratory study. In *Proceedings of SSBSE*, volume 7515 of *LNCS*, pages 168–182, 2012.

[S24] A. Murashkin, M. Antkiewicz, D. Rayside, and K. Czarnecki. Visualization and exploration of optimal variants in product line engineering. In *Proceedings of SPLC*, pages 111–115, 2013.

[S25] G. G. Pascual, M. Pinto, and L. Fuentes. Run-time adaptation of mobile applications using genetic algorithms. In *SEAMS*, pages 73–82, 2013.

[S26] L. E. Sanchez, S. Moisan, and J.-P. Rigault. Metrics on feature models to optimize configuration adaptation at run time. In *Proceedings of CMSBSE@ICSE*, pages 39–44, 2013.

[S27] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar. Optimum feature selection in software product lines: Let your model and values guide your search. In *Proceedings of CMSBSE@ICSE*, pages 22–27, 2013.

[S28] A. S. Sayyad, J. Ingram, T. Menzies, and H. Ammar. Scalable product line configuration: A straw to break the camel's back. In *ASE*, pages 465–474. IEEE, 2013.

[S29] A. S. Sayyad, T. Menzies, and H. Ammar. On the value of user preferences in search-based software engineering: a case study in software product lines. In *Proceedings of ICSE*, pages 492–501, 2013.

[S30] S. Segura, J. A. Galindo, D. Benavides, J. A. Parejo, and A. R. Cortés. BeTTy: benchmarking and testing on the automated analysis of feature models. In *Proceedings of VaMoS*, pages 63–71, 2012.

[S31] S. Segura, J. A. Parejo, R. M. Hierons, D. Benavides, and A. R. Cortés. Automated generation of computationally hard feature models using evolutionary algorithms. *Expert Syst. Appl.*, 41(8):3975–3992, 2014.

[S32] H. Serajzadeh and F. Shams. The application of swarm intelligence in service-oriented product lines. In *SPLC Workshops*, page 12, 2011.

[S33] R. Shi, J. Guo, and Y. Wang. A preliminary experimental study on optimal feature selection for product derivation using knapsack approximation. In *PIC*, pages 665–669, 2010.

[S34] L. Tan, Y. Lin, H. Ye, and G. Zhang. Improving product configuration in software product line engineering. In *36th Australasian Computer Science Conference*, ACSC '13, pages 125–133, 2013.

[S35] M. I. Ullah. Cope+: A method for design and evaluation of product variants. Technical Report SERG-2009-03, August 2009.

[S36] S. Wang, S. Ali, and A. Gotlieb. Minimizing test suites in software product lines using weight-based genetic algorithms. In *GECCO*, pages 1493–1500, 2013.

[S37] Z. Wu, J. Tang, C. K. Kwong, and C.-Y. Chan. An optimization model for reuse scenario selection considering reliability and cost in software product line development. *Intl. Jnl. of Inform. Tech. and Decision Making*, 10(5):811–841, 2011.

[S38] Z. Wu, J. Tang, and X. Wang. Integrated design of production strategy and reuse scenario for product line development. In *SEDM*, pages 69–74, June 2010.

[S39] Z. Xu, M. B. Cohen, W. Motycka, and G. Rothermel. Continuous test suite augmentation in software product lines. In *Proceedings SPLC*, pages 52–61, 2013.

[S40] L. Yi, W. Zhang, H. Zhao, Z. Jin, and H. Mei. Mining binary constraints in the construction of feature models. In *RE*, pages 141–150, 2012.

[S41] L. Yu, F. Duan, Y. Lei, R. Kacker, and D. R. Kuhn. Combinatorial test generation for software product lines using minimum invalid tuples. In *HASE*, pages 65–72. IEEE Computer Society, 2014.

[S42] W. Zhang, H. Zhao, and H. Mei. Binary-search based verification of feature models. In *Top Productivity through Software Reuse*, volume 6727 of *LNCS*, pages 4–19. 2011.