Attributed Variability Models: Outside the Comfort Zone

Norbert Siegmund
Bauhaus-University Weimar, Germany

Stefan Sobernig
WU Vienna, Austria

Sven Apel
University of Passau, Germany

ABSTRACT

Variability models are often enriched with attributes, such as performance, that encode the influence of features on the respective attribute. In spite of their importance, there are only few *attributed* variability models available that have attribute values obtained from empirical, real-world observations and that cover interactions between features. But, what does it mean for research and practice when staying in the comfort zone of developing algorithms and tools in a setting where artificial attribute values are used and where interactions are neglected? This is the central question that we want to answer here. To leave the comfort zone, we use a combination of kernel density estimation and a genetic algorithm to rescale a given (real-world) attribute-value profile to a given variability model. To demonstrate the influence and relevance of realistic attribute values and interactions, we present a replication of a widely recognized, third-party study, into which we introduce realistic attribute values and interactions. We found statistically significant differences between the original study and the replication. We infer lessons learned to conduct experiments that involve attributed variability models. We also provide the accompanying tool Thor for generating attribute values including interactions. Our solution is shown to be agnostic about the given input distribution and to scale to large variability models.

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**;

KEYWORDS

Variability modelling, attributed variability models, Thor

ACM Reference format:
Norbert Siegmund, Stefan Sobernig, and Sven Apel. 2017. Attributed Variability Models: Outside the Comfort Zone. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 11 pages.
https://doi.org/10.1145/3106237.3106251

ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany

ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3106251

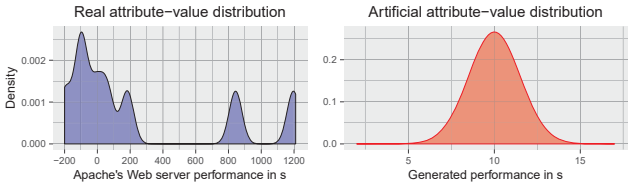


Figure 1: Exemplary density plots of real-world (left) and generated (right) attribute-value distributions. The *density* captures the probability of a set of features or interactions taking on attribute values over a specific range.

1 INTRODUCTION

The rise of variability modeling has been a success story both in industry [9] and research [13]. A *variability model* describes all valid configurations of a configurable system by specifying variation points in terms of features and constraints among them. Often, features influence quality attributes of a system, such as performance. To incorporate this influence in engineering tasks, such as finding an optimal configuration, developers and domain experts model and reason about *attributes* of features yielding an *attributed variability model* [8].

Attributed variability models form the conceptual basis of various research areas, such as multi-objective configuration optimization [19], runtime software adaptation [25], and service-oriented architecture [39]. In spite of their importance and general applicability, there are only few variability models publicly available that contain realistic attribute values [24], as we will discuss. Obtaining realistic values for performance and other *quantitative* quality attributes is expensive, as this requires measuring actual system variants (i.e., *configurations*) and determining the effects of selecting individual features on the attributes. Even worse, effects on quality attributes caused by interactions between features are largely ignored in prior work, although it has been shown that feature interactions are key to practicality and feasibility [32].

What does it mean for research and practice when staying in the comfort zone of developing algorithms and tools in a setting where artificial attribute values are used and where interactions are neglected? This is the central question that we want to answer here. Clearly, simply generating attribute values following standard distributions hinders researchers to explore corner cases or even to vary the input for their techniques systematically. For example, on the left of Figure 1, we show an attribute-value (performance) distribution obtained from a real-world system (Apache's Web server). Compare this distribution with a normal distribution (Figure 1, right), frequently assumed in the literature. This artificial distribution gives rise to a configuration space in which the optimal configurations can be easily found via simple hill climbing. In the same vein, ignoring feature interactions yields problem instances in which non-linearity does not occur – the resulting search space

becomes homogeneous and steady. This renders a search task into a convenient one, if not trivial. As a research community, we need to ask ourselves: Have we enjoyed the comfort zone and kept ignoring attribute-value distributions and interactions as important factors in our experiments for years?

Our aim is not only to make the community aware of this problem, but also to provide a solution to step outside the comfort zone. To mitigate the lack of realistic attributed variability models and to enable systematic testing of algorithms and tools in this area, we propose an approach and accompanying tool, called Thor, to generate realistic attributed variability models based on a given distribution of attribute values for features, interactions, and configurations. Based on insights from feature-interaction detection [3], Thor is able to generate different distribution patterns of interactions, both in terms of their number and degree. To make a leap toward a realistic setting, we have developed a novel technique that combines kernel density estimation [35] and the genetic algorithm NSGA-II [15]. Our technique allows to apply a given (empirically determined) attribute-value distribution profile to a given variability model. That is, we *rescale* a given distribution to another variability model such that we yield a similar distribution, guaranteed by kernel density estimation, as we will explain. So, for the first time, one can systematically explore how realistic distributions and interactions affect the quality, optimality, error-proneness, and performance of their respective algorithm or tool.

To summarize, we have two goals. *First*, we want to raise the community's awareness of the consequences of using unrealistic settings to develop, tune, and test their algorithms and tools on attributed variability models. *Second*, we want to offer a comprehensive solution to overcome the lack of realistic attributed variability models for quantitative quality attributes. To reach these goals, we make the following contributions:

- We propose an approach based on kernel density estimation and a genetic algorithm to apply (empirically determined) attribute-value distributions to a given variability model, and we incorporate interactions in a realistic manner. We demonstrate that our approach is agnostic about the given value distribution.
- We provide the open-source tool Thor and evaluate the effectiveness of kernel density estimation as well as Thor's scalability with respect to a different number of configurations.
- We report on a replication of a study on multi-objective optimization of attributed variability models by Sayyad et al. [27], demonstrating that the inclusion of interactions and the choice of the attribute-value distribution indeed affect the outcome of such a study. The study has been the basis of several follow-up studies, rendering it a reference point in this area. The differences between the original study and our replication offer new insights into the effects of attribute-value distributions and interactions and affects also very recent papers that reuse the same experimental setting.

Background material, a replication package, all measurement data, the open-source tool Thor, and the corpus of real-world distributions are available at a supplementary Web site: https://github.com/se-passau/thor-avm.

2 PRELIMINARIES AND PROBLEM STATEMENT

In this section, we give background information on attribute-value distributions as probability distributions and report on a literature study we conducted to assess the current state of using attributed variability models.

2.1 Probability Distributions

An attribute-value distribution is the ordered set of continuous probabilities of features or interactions taking certain quality-attribute values—hence, a probability distribution of quality-attribute values. That is, it captures how features distribute over different intervals of attribute values. For example, a distribution indicates whether the attribute can be represented by typical attribute values only (e.g., summary statistics, such as median and variance). An attribute-value distribution is commonly reported using boxplot statistics and visualized, for example, using a kernel-density plot over the continuous attribute-value range (as in Figure 1).

The shape of a distribution can highlight majority and minority groups of features and interactions as well as their relative influence on a given quality attribute. The capacity to highlight the relative influence of a minority is critical, because conventional summary statistics represent majority groups only. When analyzing attributed variability models, however, minorities (e.g., single features) can contribute heavily to characterizing a system's quality profile. For example, in a content management system, many features do not contribute to the performance of the system, but the choice of the database feature affects most of the system's performance. In this vein, Siegmund and others demonstrated that performance models derived from small sets of individual features capture important system-wide performance influences [31].

2.2 State of the Art

We conducted a literature study to obtain an overview of (a) the extent to which generated and real values for attributed variability models are used in the literature, (b) the assumptions and choices made by researchers that use generated attribute values, and (c) the awareness of the relevance of feature interactions. We present our methodology (e.g., the selection criteria), a description of the analyzed papers, and threats to validity on our supplementary Web site. In what follows, we summarize the procedure and the key findings: Based on an iterative paper selection, we obtained a set of 2346 papers, which we analyzed for references to attributed variability models. More specifically, we were interested in all papers, in which attributed variability models have been used to accomplish a certain task (e.g., optimization). This analysis revealed 69 papers, which we analyzed further. We found that 52 of the 69 papers use generated attribute values. Only 8 papers rely on real-world values, either obtained by actual measurements or extracted from other sources (e.g., the Web). From the 52 papers, 21 papers used a random value generator following a uniform distribution, 10 papers use a normal distribution, and 13 papers assign values in an ad-hoc manner without a certain target distribution. 17 papers do not specify the way how they generated the data. Remarkably, from the 69 papers we analyzed, not a single paper considered interactions. Only 8 papers discuss the absence of interactions as a threat to validity.

Our findings raise the question of whether research in this area is on solid grounds. Even some of the authors of the papers in question explicitly state this threat:

> "Finally, a threat is due to the artificial way the values of the attributes were assigned[..]" [19]

Furthermore, it seems that this threat to validity, explicitly mentioned though, is not carefully considered. In the work of Sayyad et al. [27], the authors state:

> "A potential threat to construct validity is the use of artificial attribute values as attributes of features[...]. Future work should attempt to collect real data for use with IBEA and other MEOAs to best optimize product configuration."

Overall, there are 6 papers in our corpus published within just two years in top-ranked software-engineering conferences that explicitly cite Sayyad's paper and use the *same* artificial data as attribute values. Unfortunately, the corresponding threat to validity is not even mentioned anymore. Likewise, the fact that interactions are practically ignored in research is astonishing as much as it is disturbing especially as interactions are common in real-world software systems [3].

3 GENERATING REALISTIC ATTRIBUTED VARIABILITY MODELS WITH THOR

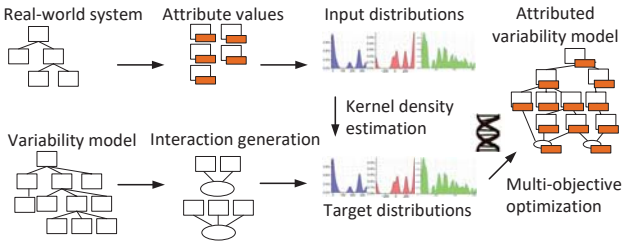


Figure 2: Generating attributed variability models.

In our approach, we consider *quantitative* quality attributes measured on a metric scale. The key idea is to generate corresponding attribute values for a given variability model based on attribute values determined before, either through (i) empirical observations of real-world applications (e.g., performance measurements), (ii) reverse engineered values (e.g., from Web sites [23]), or (iii) specifically generated attribute distributions (e.g., to test corner cases). Furthermore, users may include different sorts of interactions with their respective attribute values into the model. For the first time, this approach allows users to create their own and highly controlled test bed, in which they can purposefully vary the number and distribution of interactions as well as reuse profiles of attribute values that have been obtained from real applications. In what follows, we use the term *distribution profile* to describe a specific distribution of attribute values over the set of features, interactions, and configurations.

Figure 2 provides an overview of all steps involved in generating a realistic attributed variability model. Steps of the upper part aim at obtaining distribution profiles of a real system for features, interactions, and configurations. There are a few data sets already available [31, 33, 34], which have been incorporated into THOR, the tool accompanying the approach; the community can contribute further distribution profiles.

Here, we concentrate on the steps of the lower part in Figure 2: As input, we require a variability model (e.g., in SPLOT or DIMACS format), which is going to be augmented by realistic attribute values. In a next step, a user specifies whether we should generate interactions between features of the given variability model. This step addresses the problem that most variability models lack feature interactions (cf. Section 2.2). Then, the user specifies the distribution profiles to be applied to features, interactions, and configurations. These profiles act as separate objectives in the multi-objective optimization step, in which we generate attribute values using a genetic algorithm (see Section 3.3).

3.1 Objective Formalization

An attributed variability model is a triple $AVM = (\mathcal{F}, \mathcal{C}, \mathcal{M})$, where $\mathcal{F}$ is the set of features, $\mathcal{C}$ is the set of constraints, and $\mathcal{M}$ is a mapping $\mathcal{M} = \{ fc \mapsto r \mid fc \in \mathcal{P}(\mathcal{F}) \wedge r \in \mathbb{R}\}$ from a feature or a combination of features (i.e., an interaction, which is an element of the powerset $\mathcal{P}$) to their quality attribute, represented as a real number in $\mathbb{R}$.[1] Since we are interested in the distribution of attribute values and not necessarily in the specific mapping, we compute an output distribution profile $\mathcal{D}_O$ as a vector on real numbers with

$$\mathcal{D}_O = \langle r \mid r \in \mathcal{M}(\mathcal{P}(\mathcal{F}))\rangle$$

Let us first review the goal of our approach from a single objective point of view: The user specifies an input distribution profile $\mathcal{D}_I$ as a vector of real numbers. The objective is now to generate an output distribution profile $\mathcal{D}_O$ such that $\mathcal{D}_O$ is similar to $\mathcal{D}_I$ given a certain similarity metric (following the notation by Boyd and Vandenberghe [10]):

$$\underset{M}{\text{maximize}} \quad sim(\mathcal{D}_I, \mathcal{D}_O), \tag{1}$$

where *sim* is a similarity measure that can be instantiated with different functions. In essence, we aim at finding an optimal mapping from features and interactions to attribute values, such that their distribution is similar to the given one. Let us assume that a user specifies a distribution profile (i.e., the *input* distribution in Figure 3) using only 20 elements in the vector. The optimization process now aims at finding an *output* distribution (cf. Figure 3) for a desired variability model with 500 features and interactions, while maintaining the overall distribution profile of the input.

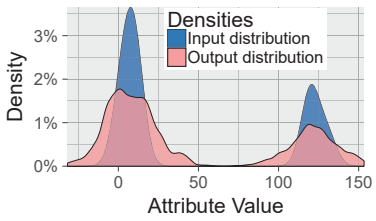


Figure 3: Comparison of an input and output distribution.

Next, we have to extend our optimization problem to three types of input distribution profiles. The background is that the attribute values of individual features of a system often have a different distribution profile (e.g., in the effect strength and in their number of values) than interactions among features. Hence, we consider $\mathcal{D}_I^{\mathcal{F}}$ as the input profile for features and $\mathcal{D}_I^{\mathcal{I}}$ as the input profile for interactions. Furthermore, there is the distribution profile of valid configurations (i.e., system variants) denoted by $\mathcal{D}_I^{\mathcal{V}}$. The inclusion

[1] The size of $\mathcal{P}$ is exponential in the size of $\mathcal{F}$. In practice though, there are way fewer interactions that have a relevant effect on an attribute value.

of the latter renders our approach fundamentally different from existing work. This is because most of the algorithms in the literature do not work directly on the attribute values of features, but rather on the attribute values of configurations [26]. For instance, finding the fastest configuration using a genetic algorithm evaluates the performance of configurations and not of individual features [27].

An attribute value of an individual configuration is computed by aggregating the attribute values of features and their interactions that appear in the configuration (i.e., all selected features and their corresponding interactions). A common aggregation function $\Pi : \mathcal{P}(\mathcal{F}) \mapsto \mathbb{R}$ computes the linear combination of all relevant attribute values of a given configuration *fc* as follows [31]:

$$\Pi(fc) = \sum_{i \in \mathcal{P}(fc)} \mathcal{M}(i)$$

Here, we sum the values of all elements in the powerset of configuration *fc*, which includes all features and all interactions. Furthermore, function Φ

$$\Phi(\mathcal{F}, C) = \{vc \mid vc \in \mathcal{P}(\mathcal{F}) \wedge vc \text{ is valid wrt. } C\}$$

computes the set of all valid configurations based on the features and constraints in the attributed variability model. Each configuration contains only the features that are selected. Function Φ is usually realized by a SAT solver [6]. To obtain the distribution profile $\mathcal{D}_O^{\mathcal{V}}$, we compute the attribute values of all valid configurations:

$$\mathcal{D}_O^{\mathcal{V}} = \langle r \mid r = \Pi(vc) \wedge \forall vc \in \Phi(\mathcal{F}, C) \rangle$$

An important challenge for computing $\mathcal{D}_O^{\mathcal{V}}$ to be similar to $\mathcal{D}_I^{\mathcal{V}}$ is that $\mathcal{D}_O^{\mathcal{V}}$ depends on the mapping $\mathcal{M}$, such that changing $\mathcal{D}_O^{\mathcal{F}}$ and/or $\mathcal{D}_O^{\mathcal{I}}$ influences also $\mathcal{D}_O^{\mathcal{V}}$. For example, changing the attribute value of a single feature in $\mathcal{M}$ changes the attribute value of every configuration, in which the respective feature is selected. Moreover, $\mathcal{D}_O^{\mathcal{V}}$ takes only valid configurations into account. Therefore, constraints can have a substantial influence on which combinations of features and which interactions can occur in a valid configuration, affecting $\mathcal{D}_O^{\mathcal{V}}$.

In Figure 4, we give an example of how we compute $\mathcal{D}_O^{\mathcal{V}}$ based on a genetic encoding using (a) a configuration matrix ($m \times r$, where m is the number of configurations and r is the sum of the number of features and interactions) to model feature selections and the resulting presence of interactions and (b) a value matrix to model the mappings from features and interactions to their values ($r \times 1$). Each row in the configuration matrix on the left represents a genetically encoded valid configuration. The whole set of configurations is computed using function Φ. The presence of interactions depends on whether their corresponding features are also selected in the respective configuration. In the center of the figure, we depict the value matrix, where features and interactions are mapped to their corresponding attribute values. We use a global ordering of features and interactions, such that we can use the dot product to compute the attribute value of each configuration, as shown with the matrix ($m \times 1$) on the right of the figure. We further show the relationship between the mapping function $\mathcal{M}$ and the distribution profiles $\mathcal{D}_O^{\mathcal{F}}$ and $\mathcal{D}_O^{\mathcal{I}}$ as well as how we obtain $\mathcal{D}_O^{\mathcal{V}}$.

Combining the three input profiles results in the following multi-objective optimization problem:

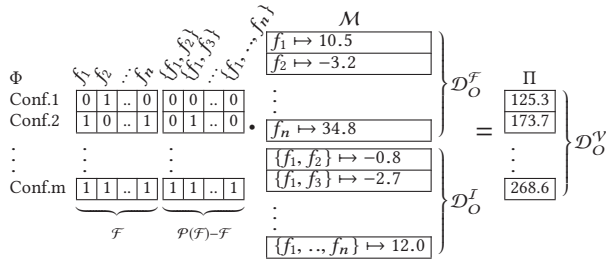


Figure 4: The process of computing the output distribution profile $\mathcal{D}_O^{\mathcal{V}}$ based on a genetic encoding of features and interactions combined into a configuration matrix. The configuration matrix is combined using the dot product with the matrix containing values of features and interactions, determined by function $\mathcal{M}$.

$$\underset{\mathcal{M}}{\text{maximize}} \quad w_1 \cdot sim(D_I^{\mathcal{F}}, D_O^{\mathcal{F}}) + w_2 \cdot sim(D_I^{\mathcal{I}}, D_O^{\mathcal{I}}) + w_3 \cdot sim(D_I^{\mathcal{V}}, D_O^{\mathcal{V}}) \tag{2}$$

where $w_1 + w_2 + w_3 = 1$ and $0 \leq w_1, w_2, w_3 \leq 1$. The remaining task is to *efficiently* learn function $\mathcal{M}$.

3.2 Including Interactions

As said previously, most variability models do not come with interactions. So, we have to include new interactions into the given model. However, we cannot include interactions blindly, because then interactions will often have no effect. For example, generating an interaction between mutually exclusive features will not affect the attribute value of a configuration, because both features will never be present together in any valid configuration. Furthermore, interactions arise not randomly in practice, but follow certain patterns. Hence, in Thor, a user can specify (1) the number of interactions and (2) the ratio of the degree of the interactions to be included.[2] For example, we can specify that 80 % of all interactions are of degree two (i.e., between two features) and 20 % of the interactions are of degree three (i.e., among three features).

Our algorithm to inject an interaction is as follows: First, we select two features (for pair-wise interactions) at random. Second, we query a SAT solver to verify: Are there valid configurations with both features selected and where each feature is selected, but not the other one? Third, if so, we add an interaction with the chosen features to the model; we repeat otherwise. For example, if we find that the features f_1, f_2, and f_5 can be simultaneously selected in a configuration and the selection of any of these features does not imply the selection of another feature in this set, an interaction will be generated such that there is a mapping $\mathcal{M} : \{f_1, f_2, f_5\} \rightarrow \mathbb{R}$ that needs to be learned in the optimization process.

3.3 Generating Attribute Values

The remaining task is to find an optimal mapping from features and interactions to real numbers such that we solve the multi-objective optimization problem defined in Equation 2. We start from a random set of values and use a genetic algorithm for adjusting the values to optimize for all three objectives (i.e., the three terms in Equation 2).

[2]The degree of an interaction denotes the number of participating features in the interaction.

As a result, we compare the Pareto front of all solutions with respect to the distribution profiles. The Pareto front can be used to inspect the solutions to find the output distribution that fits best a user's requirements. For example, some users might prefer a precise match between $\mathcal{D}_I^{\mathcal{F}}$ and $\mathcal{D}_O^{\mathcal{F}}$, whereas others might tend to a mapping closer to the similarity of the distribution profiles of configurations.

Kernel Density Estimation. Starting from randomly initialized values seems to be inefficient and can lead to suboptimal solutions due to local optima. Hence, we propose an optimization to the process described before based on *kernel density estimation* [35]. Kernel density estimation is a non-parametric method in statistics for estimating the probability density function $P(x)$ of a random variable x: $P(x) = \int_{-\infty}^{\infty} K(x)\,dx$, where $K(x)$ is the kernel function to be determined. So, when we apply this function to a given input distribution profile $\mathcal{D}_I$, we obtain a probability density function that describes an underlying function from which the values in $\mathcal{D}_I$ originate. Now, we can draw new samples from this function while maintaining the overall distribution (as we did in Figure 3). This method solves two problems: (1) we can scale a given distribution profile to a much larger distribution profile $|\mathcal{D}_I| \ll |\mathcal{D}_O|$ without having the same values; (2) we provide a good starting assignment for the mapping function to speed up the optimization process and mitigate the thread of local optima. To additionally improve similarity for the initial assignment of values, we use the two-sample Kolmogorov–Smirnov test [36] to measure the extent ("goodness") of the fit between the input distribution and the intermediate output distribution generated by kernel density estimation. Of course, also the other metrics are applicable here.

Optimization Process. Starting from an initial set of solutions (e.g., randomly assigned numbers or drawn from kernel density estimation), the genetic algorithm changes the numbers in the mapping function in an iterative process to maximize the objective function (i.e., to improve the goodness-of-fit scores). Changes to the numbers are made by standard genetic operators, such as mutation and crossover, as provided by NSGA-II.

To make the computation of $\mathcal{D}_O^{\mathcal{V}}$ feasible, we select and measure only a sample set of configurations. The *sample set* is collected using a user-defined sampling technique in combination with a SAT solver. The rows in the matrices on the left in Figure 4 represent the sample set. Based on this set, we compute the presence of interactions by checking for each interaction and each configuration whether the features causing the interaction are all present in the respective configuration (see the right part of the matrix on the left of Figure 4). We compute the matrix on the left only once before we start the genetic algorithm. All other vectors and matrix operations have to be populated and applied in each evolutionary step, as the generated attribute values change after each iteration. That is, in each iteration, we compute the dot product to obtain the attribute value for each configuration.

4 EVALUATION: REPLICATION

To demonstrate how THOR facilitates the systematic investigation of the relevance of interactions and the choice of attribute values, we replicate a popular study on attributed variability models [27, 28]. We selected this study because we had identified it as the most fundamental one. Our literature study revealed the two corresponding papers as the most cited ones. In addition, we found critical details of Sayyad's study adopted by several follow-up studies (e.g., the attribute data).

In a series of experiments, Sayyad et al. [27, 28] contrasted the performance (solution quality and speed) of two evolutionary meta-heuristics (NSGA-II, IBEA) on finding valid configurations of attributed variability models under multiple objectives. The objectives were: validity of a solution, maximal configuration size, as well as attribute values of features, such as state of reuse, defect counts, and costs. Sayyad et al. performed their experiments on seven different, reverse-engineered variability models taken from the LVAT repository[3] in DIMACS format. Each variability model was associated with artificial attribute values having the same properties (distributional profiles). The key findings were that the tested meta-heuristics are capable of finding valid and optimal configurations in practical time boxes (30 minutes) for configuration and optimization spaces otherwise unsuitable for exact approaches. In particular, IBEA was found to out-perform NSGA-II regarding solution quality for the computation tasks at hand. Solution quality was measured in terms of Hypervolume (HV) and valid solutions in Pareto-optimal solution sets (PCORRECT).[4]

While considering a diverse set variability models, the original study by Sayyad et al. [27] does not include robustness checks of the meta-heuristics in the face of varying attribute-value distributions and the presence of interactions. With the kind support of Sayyad et al., we performed replications for the seven variability models from the baseline experiment to assess the robustness regarding systematically varying attribute-value distributions and for the inclusion of interactions.

In our replications, we re-run the experiments with *identical* protocol and operationalization conditions of Sayyad's baseline experiment, while altering only two experimental conditions: the distributional shape of attribute values and the presence of feature interactions. All the other details of protocol, operationalization, and the other experimental objects remain unchanged, including: variability models, algorithm implementation, number of objective functions, parameter settings, outcome measures, timeboxes. Design-wise, this is a form of operational or differentiated replication [17]. This implies that, for the scope of such a replication, other threats to the baseline experiment are not controlled for (e.g., threats to internal or construct validity). We designed and report this replication according to the guidelines on robustness of computational tests [4, 5] with emphasis on evolutionary meta-heuristics [37]. A statistical companion and the replication package are available at the supplementary Web site.

In Section 4.1, we report on the replication for one LVAT model (Toybox) in all detail. Toybox is a command-line utility that combines a subset the GNU shell commands into a single executable. We selected the Toybox model as a showcase because it is illustrative for robustness effects found across the replications. Then, in Section 4.2, we summarize the replication results for the six remaining LVAT models, which were obtained from running the identical replication protocol as for Toybox.

[3] https://code.google.com/archive/p/linux-variability-analysis-tools/
[4] Hypervolume (HV): a ratio capturing the coverage of a known Pareto front with respect to the objective space in a 2+-optimization problem [11]; PCORRECT: the ratio of valid configurations contained by the Pareto-optimal solution set [27].

4.1 Toybox Replication

To establish a baseline, we successfully confirmed the original results for Toybox's variability model (using 50 rather than 10 independent runs): At comparable levels of HV (IBEA: median 0.22/MAD[5] 0.0004; NSGA-II: 0.21/0.011), a search process using IBEA results in more than twice as many valid configurations in the Pareto-optimal solution sets than NSGA-II (IBEA 25.2% valid, on average; NSGA-II: 10.83% valid, on average).

Setup. We are interested in whether the winning IBEA performs equally well when varying the attribute values for one objective (COST) as compared to the confirmatory replication. All other four objectives remained unchanged. We selected the attribute COST, because it has a continuous data scale. This allows us to test a range of different distributions without introducing additional assumptions. Planned variations of COST included (a) imposing an empirical data distribution derived from a real-world data set (as opposed to the original normal distribution) and (b) the inclusion of interactions between features affecting COST. As data set, we use measurements from x264, a video encoding library. We included the same number of interactions as features are in the Toybox variability model (544; hence, 100 %, denoted as FI100); other numbers have shown similar results. The interaction degrees follow artificial and empirical (x264) interaction data [31].

We tested the following null hypotheses on the *two* outcome variables available for Toybox from the original study (HV, PCORRECT):

- **H1^0:** There is *no* difference between the mean outcome obtained by IBEA when optimizing for the original, normally distributed ("artificial") attribute COST and the x264-based ("empirical") attribute COST.
- **H2^0:** There is *no* difference between the mean outcome obtained by IBEA when optimizing for the original COST attribute without interactions (F) and a COST attribute adjusted for interactions (FI100).
- **H3^0:** There is *no* difference between different combinations of artificial/empirical distribution shapes and of COST computation with/without interactions.

We tested these hypotheses using a two-way analysis of variance (ANOVA) procedure based on a 2×2 data layout: DIST (normal, x264) × FINT (F, FI100). In preparation, we performed the standard checks for an ANOVA (i.e., normality of residuals and the homogeneity of variances in each data cell). The nulls are discredited at a significance level smaller or equal than 0.05.

The computational test for the extended replication included 50 runs per factor combination (i.e., 200 independent runs each limited to 30 minutes runtime maximum). We performed the runs on an HPC cluster, offering 17 nodes running Ubuntu 14.04 with 64 GB RAM and 20 cores each (Intel Xeon E-5 2690v2 CPU, 10 physical cores). Each run was allocated identical computational resources to guarantee comparability. The job script is available at the supplementary Web site.

Results. We tested the three hypotheses for each outcome variable, HV and PCORRECT, respectively; see Table 1. For both, we found significant, substantial, and non-trivial variability in the light of different attribute-value distributions (DIST: normal, x264) and of the inclusion/exclusion of interactions (FINT: F, FI100).[6]

Figure 5 shows a line graph to examine the main effects of DIST and FINT on HV and PC, respectively, as well as their interaction effects. The mean outcomes (HV, PC) for each level of one factor (DIST: normal, x264) are plotted over all levels of FINT (F, FI100). An *interaction line* connects pairs of mean outcomes (HV, PC) from the angle of the experimental factor on the x-axis (i.e., FINT; see A–B and C–D in Fig. 5). The interaction lines and their relative position allow us to visually identify the type and the strength of the (possible) interaction effects. For example, two parallel lines signal that there is no interaction effect at all and two intersecting lines denote the presence of some interaction between the two factors.

Hypervolume (HV). There is a significant, but small ordinal interaction ($p<0.001$, $\eta^2 < 0.01$) between FINT and DIST (H3): The effect of distributions (normal, x264) is different at the two FINT levels (F and FI100), that is, it appears slightly stronger for COST computations neglecting interactions (F). However, the order of magnitude of the DIST effect is unchanged (hence, *ordinal*) across the two different FINT levels (F and FI100). Despite this interaction effect, there is a noteworthy—significant and large—effect of FINT ($p<0.001$, $\eta^2 = 0.859$) on HV (H2): 85.9 % of the variance in HV is associated (covaries) with FINT (excluding or including interactions). HV obtained by IBEA increases by 1.16 scores (confidence interval, CI: 95 %: 1.12 to 1.18; Tukey HSD) when including feature interactions (FI100); or decreases by the same amount when excluding them (see the steep slopes of the interaction lines A–B and C–D in Fig. 5, on the left). Testing different value distributions (DIST: normal, x264) results in a significant, medium-level effect of DIST ($p<0.001$, $\eta^2=0.134$; H1). This effect is visualized by the gap between the mid-points of the interaction lines in Fig. 5 (on the left).

PCORRECT (PC). We found a significant and large disordinal ("cross-over") interaction between DIST and FINT ($p<0.001$, $\eta^2 = 0.428$): 42.8 % of the variance in PCORRECT can be linked to the levels of FINT and DIST interacting with each other (H3). The order of magnitude of the DIST effect changes (hence, *disordinal*) across the two different levels of cost (with and without interactions). In Figure 5 (on the right), this disordinal interaction is depicted as a cross-over between the two interaction lines A–B and C–D. This interaction effect between FINT and DIST gives rise to a significant and immediately tangible difference: Including interactions (FI100) for the real-world distribution (x264) yields additionally 9.6 % (CI: 95 %: 9.3 to 9.9 %) of valid configurations found by IBEA when compared to neglecting interactions for the same distribution (see the different slopes of the x264 interaction line C–D and the normal interaction line A–B in Fig. 5, on the right). When compared to the original setup (no interactions, normal; A vs. C), this difference amounts to an averaged decrease of 4.7 % (CI: 95%: 4.4 to 5). Hence, depending on the combination of distribution and interactions, IBEA yields considerably more or less valid configurations in its solution set. Given their disordinal interaction, the results and interpretations of the effects of DIST must always be qualified in terms of the effect of FINT; and vice versa. As a consequence, we

[5] Median absolute deviation; a robustness measure.

[6] Beware that the term *interaction* has two meanings in this section: "feature interaction" as an experimental factor in its own right (FINT: F, FI100) vs. "interaction effect" between two experimental factors (DIST+FINT) in terms of the ANOVA.

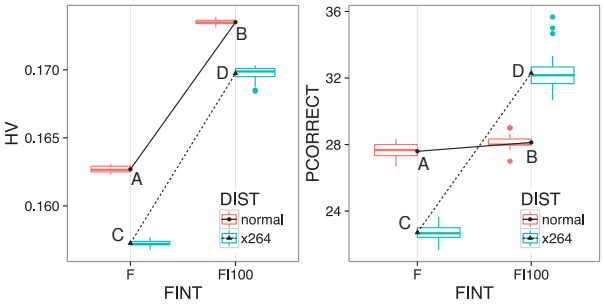


Figure 5: Nested box and interaction plots for the outcome variables, with the lines A–B and C–D representing the "interaction lines"; on the left (Hypervolume, HV): significant ordinal interaction effect (DIST+FINT), plus significant main effects (DIST, FINT); on the right (PCORRECT): significant disordinal interaction effect (DIST+FINT)

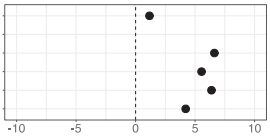
do not interpret the effects of FINT and DIST (although significant) in isolation from each other (H1, H2).

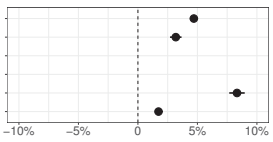
Table 1: Variance tables; SS: sum of squares; Df: degrees of freedom; MSS: Mean sum of squares; +: interaction between two factors

	Src	SS	Df	MSS	F	p	η^2
HV	DIST (H1)	1.1e−3	1	1.1e−3	12714	<0.001	0.134
	FINT (H2)	6.8e−3	1	6.8e−3	81544	<0.001	0.859
	DIST+FINT (H3)	3.5e−5	1	3.5e−5	423	<0.001	0.005
		1.6e−5	196	8.3e−8			
PC	DIST (H1)	6.5	1	6.48	15	<0.001	0.003
	FINT (H2)	1280.0	1	1280.0	3098	<0.001	0.535
	DIST+FINT (H3)	1025.0	1	1025.0	2479	<0.001	0.428
		81.0	196	0.41			

Next, we summarize the replications of the computational experiments for *all* other LVAT models used by Sayyad et al. [27].

Table 2: Effects and effect sizes (mean differences) on Hypervolume (HV) and the ratio of correct configurations (PC) from the replications of Sayyad et al. for six out of seven LVAT models; significance level ≤0.05; CI95%

Model	HV		Mean Differences on FINT (F–FI100)
	effect(s)[a]	η^2	
Toybox	FINT (DIST)	0.859 (0.134)	
axTLS[b]	FINT+DIST	0.401	
eCos	FINT (DIST)	0.966 (0.034)	
FreeBSD	FINT (DIST)	0.993 (0.005)	
Fiasco	FINT (DIST)	0.997 (0.002)	
uClinux	FINT (DIST)	0.914 (0.075)	

Model	PC		Mean Differences on FINT+DIST[c]
	effect(s)[a]	η^2	
Toybox	FINT+DIST	0.443	
axTLS	FINT+DIST	0.401	
eCos[b]	FINT, DIST	0.452, 0.542	
FreeBSD[b]	FINT, DIST	0.350, 0.551	
Fiasco	FINT+DIST	0.181	
uClinux	FINT+DIST	0.794	

[a]FINT, DIST: two significant *and* substantial main effects; FINT (DIST): two significant main effects, but DIST is not substantial; FINT+DIST: a significant and substantial interaction effect, no main effects
[b]axTLS (for HV) as well as eCos and FreeBSD (for PC) represent important exceptions from the otherwise observed effect patterns that are discussed in the text.
[c]Mean differences between F/normal and FI100/x264

4.2 Further Replications

We conducted the differentiated replication as detailed for Toybox in Section 4.1 also for the other six LVAT models: axTLS, eCos, FreeBSD, Fiasco, uClinux, and Linux. As reported for the baseline experiment [27], Linux does not yield results for IBEA (unless using a seed configuration), therefore, it is excluded from the following report. Table 2 summarizes the findings for the remaining six models (incl. Toybox). Overall, we found significant, substantial, and non-trivial effects of different attribute-value distributions (DIST: x264 vs. normal) and of the presence of interactions (FINT: F vs. FI100) on solution-quality indicators (HV, PC) when applying the evolutionary meta-heuristic IBEA on the six optimization problems (see columns "effect(s)" in Table 2):

FINT, DIST: There are both main effects (significant and substantial).

FINT (DIST): FINT shows a significant and substantial effect, DIST is significant but *not* substantial.

FINT+DIST: There is a significant and substantial interaction effect between FINT and DIST (no main effects).

Hypervolume. First, in five replications, adding or removing interactions from the computation (FINT) is capable of increasing or decreasing the coverage of the objective space by the solution set (HV), independently from the underlying distribution (DIST). The upper compartment of Table 2 summarizes the typical differences ranging between 1.16 (min, Toybox) to 6.7 HV scores (max, eCos). The same replications also indicate a significant and, when compared to interactions, very small effect of the underlying attribute-value distribution (normal, x264). A noteworthy exception is the axTLS replication, which counters the above findings. There is an interaction FINT+DIST with HV being reduced substantially for x264 attribute-value distribution with interactions, while it increases for the normal distribution with interactions (by a smaller fraction though). IBEA for axTLS with an x264 distribution and with interactions (FI100/x264) obtains 11.43 HV scores less than axTLS with normal distribution and without interactions (F/normal).

PCORRECT. Second, in the original study, the use of IBEA was reported superior because it results in more valid system configurations (PC) being found in a timebox of 30 minutes than using NSGA-II. While our replications confirm this general picture, we found that PC is sensitive to the presence/absence of interactions (FINT) as well as the distribution shape (DIST) for all six models (see Table 2; lower compartment). In three replications (Toybox, axTLS, and uClinux), interactions (FINT) and distribution shape (DIST) interfere to an extreme (cross-over): An IBEA search on a realistic attribute-value distribution (x264) yields significantly less configurations than one based on an artificial one *without* interactions (F/normal), and more valid configurations for the realistic distribution *with* interactions (FI100/x264). The latter effect amounts to mean differences on PC between 1.73% (min, uClinux) and 8.33% (max, Fiasco; see Table 2). For eCos and FreeBSD, however, interactions (FINT) and distribution (DIST) take effect independently from each other. The sizes of the effects on PC are comparatively smaller than the one interaction-effect size for the other four models: up to approximately 1% increase or decrease (FreeBSD) in valid configurations found.

Summary. In our experiments, we learnt that the solution quality (HV, PC) in the Sayyad et al. experiment is sensitive to the presence of feature interactions and realistic attribute-value distributions. Varying both factors (FINT, DIST) at a time as the experimental condition is also capable of revealing exceptional variations (axTLS, eCos, and FreeBSD). Note that changes to the setting will likely add to the observed effects on HV and PC (since we touched only one out of five objectives). Such changes include realistic distributions for the remaining attributes, varying the number of feature interactions,[7] or dropping the original value for FINT (5.0,15.0) in favor of realistic values (due to negative boundaries and heterogeneous intervals for attribute values; see Section 2).

4.3 Threats to Validity

Running a differentiated replication gives rise to threats to construct validity: Considering interactions required us to adjust the original true Pareto fronts for the HV computation to include lower and upper boundaries based on the interactions' attribute values. Also note that we had to incorporate corrections to address unreported issues in the original study. These include a contradiction regarding attribute-value distributions: Sayyad et al. stated a dependency between two attributes (objectives) used in their experiment (USED_BEFORE, DEFECTS), which effectively led to one set of test data not having the claimed normal distribution. Therefore, we performed the replication with and without this dependency, and report any variations in the statistical companion. However, the big picture does not change. A threat to external validity is that our robustness experiment is limited to one systematic variation (FI100, x264, one value range). At this point, however, the objective was to demonstrate the mere existence of significant and substantial variations due to the choice of the distribution and the presence of interactions. In future work, we shall investigate possible variation patterns on solution-quality indicators by systematically varying attribute-value distribution characteristics.

5 EVALUATION: VALIDITY & SCALABILITY

Although we have already successfully applied Thor to conduct the replication in Section 4, in what follows, we shed more light on the validity and scalability of Thor.

5.1 Validity Experiments

The validity of Thor naturally depends on whether the generated attributed variability models are realistic. By using kernel density estimation, we *rescale* a given distribution to another variability model. That is, the degree of realism depends on whether the input distribution is realistic. What remains to be shown is whether our main innovation—applying kernel density estimation as a prestep of the genetic algorithm—actually improves the similarity of the output distributions with respect to the input distributions. To this end, we conducted an experiment, in which we generate an attributed variability model for the same input distribution and the same variability model, once using kernel density estimation (KDE+GA) and once using only the genetic algorithm (GA). This way, we want to answer the following research question:

RQ1: Does kernel density estimation (KDE+GA) improve the similarity of the output distributions to the input distributions compared to pure genetic optimization (GA)?

Setup. To rate the similarity between the output and input distribution, we need ground-truth attributed variability models. We were able to get hold of five models with feature attributes, interaction attributes, and attribute values of all valid configurations [18, 31, 32], which we use for this purpose. Our aim here is to show that our technique is agnostic with respect to the given distribution profile. As similarity measures, we use three tests: the Anderson-Darling test, the Pearson's correlation coefficient, and the Euclidean distance. The rationale of using three different statistics is that a single one is usually not sufficient for comparison, due to implications of the central limit theorem: Ideally, we can check whether the output distribution and the input distribution are drawn from the same probability distribution, which is why we use the p-value of the Anderson-Darling test as a first measure. However, with a growing size of the data set, every change becomes significant and so the p-value tends to become zero. In this case, we can combine two other statistics. First, we compute Pearson's correlation coefficient to assess the linear dependence (correlation) between the input and output distribution. If the value is 1, both distributions are linearly correlated. However, this measure alone can become misleading when the output distribution has been shifted towards smaller or larger values. Hence, we compute, in addition, the distance of both distributions $dist(\mathcal{D}_I, \mathcal{D}_O) = abs(\sum \mathcal{D}_I - \sum \mathcal{D}_O)/|\mathcal{D}_O|$[8] to capture such shifts. We consider an output distribution similar to a given input when it yields a high p-value or a high correlation plus a low distance.

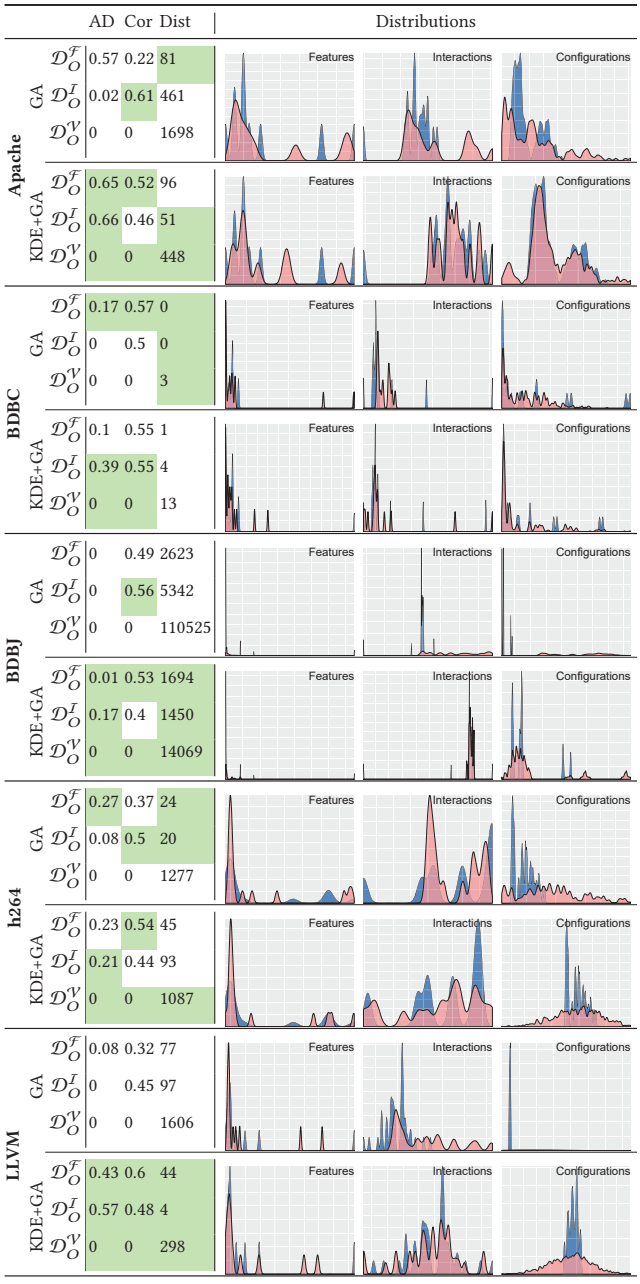
Since we have non-determinism in the genetic algorithm and the kernel density estimation, we execute the whole optimization process 30 times. Also, we use all 50 solutions provided by the genetic algorithm. This means that our metrics are averaged over 30+50 output distributions. The plotted output distributions (red& bright in Table 3) represents the best solution with respect to the p-values. We provide also a comparison with all other distributions at our supplementary Web site.

Results. In Table 3, we show the results of our validity experiment. For each model, we compare the three statistics when using KDE+GA against GA. Furthermore, to illustrate how the actual distributions compare, we plot the input distributions (blue&dark) and output distributions (red&bright). We highlighted the cells in green, in which the respective statistics suggest closer similarity. Table 3 clearly shows that, for most systems, KDE+GA provides more similar output distributions. For Apache, h264, and LLVM, the high p-values indicate that $\mathcal{D}_O^{\mathcal{F}}$ and $\mathcal{D}_O^{\mathcal{I}}$ are drawn from the same probability distribution as $\mathcal{D}_I^{\mathcal{F}}$ and $\mathcal{D}_I^{\mathcal{I}}$, respectively. For Apache, we also see that, although the p-value is close to zero for $\mathcal{D}_O^{\mathcal{V}}$, we have a nearly perfect match when looking at the plotted distributions of the configurations. For the remaining models, the p-value is usually too low such that we need to look at correlation and distance. For BDBC, we see that all metrics perform very similar for both approaches, which can also be seen in the plotted distributions. For BDBJ, we observe again a clear trend that KDE+GA approximates

[7] Prior work shows that quadratic numbers of interactions can be found when compared to the feature count [21].

[8] We sum up all elements of both distributions and compute the absolute difference, and we divide by the number of values in the distribution to obtain the mean distance.

Table 3: Comparing output distributions with input distributions using kernel density estimation (KDE) and genetic algorithm (GA). AD=Anderson-Darling test; Cor=Pearson's correlation coefficient; Dist=Euclidean distance



the true distributions substantially better than GA. This is especially apparent in the plotted distributions.

Discussion. The results paint a clear picture: Kernel density estimation improves the similarity of the output distributions substantially (RQ1). It seems that relying only on genetic optimization quickly saturates in a local optimum, such that, in many cases, $\mathcal{D}_O^{\mathcal{F}}$ and $\mathcal{D}_O^{\mathcal{I}}$ are still uniformly distributed. We conclude that, to avoid this general trap in meta-heuristic optimizations, initializing the optimization process using a seed distribution is crucial. Moreover, we also observe that statistical measures are not entirely suitable as they may depend on the size of the distribution and other factors. This is why we suggest a visual comparison at the end of the optimization process to select a proper candidate out of the Pareto front generated by THOR.

5.2 Scaling Experiments

Beside validity, we want to assess the scaling capabilities of THOR to larger-sized variability models. In particular, we want to answer the following research question:

RQ2: How does THOR's performance scale with an increasing number of configurations?

The rationale of RQ2 is that computing the matrix of configurations (cf. Figure 4) has its limits, as we can potentially generate $2^{|\mathcal{F}|}$ valid configurations. Hence, we want to assess how different sizes of the matrix affect optimization time.

Setup. To answer RQ2, and for the same reasons as in our replication (see Section 4), we use Toybox's variability model. This variability model is realistic and feasible to observe how THOR scales with respect to an increasing number of configurations. For this purpose, we generate 109 interactions (20 %) divided into 80 % of degree 2, 10 % of degree 3, and 10 % of degree 4. We repeated the run of the genetic algorithm 10 times, to account for measurement bias and random effects. We aborted the genetic algorithm after 5000 iterations and evaluate 50 populations per iteration. As input distributions, we use the distributions of binary size values (in KB) of BDBC (other distribution profiles yielded similar results).

We used an Intel Core i7–4790 @ 3.60 GHz with 16GB RAM running Windows 10 Pro, version 1607, build 14393.5 for all experiments. We provide all data at the supplementary Web site.

Results. Figure 6 shows the performance factors of the individual steps when generating attribute values (RQ2). We can see that the tasks to be executed only once have only a marginal effect on the overall execution time. Executing Φ includes the sampling process (i.e., taking only a subset of all valid configurations) and the SAT check. Matrix creation computes the binary matrix (left in Figure 4) based on the set of determined configurations and the set of generated interactions. Computing the dot product and the fitness values have to be performed at each iteration of the genetic algorithm, 50 times (one for each candidate). The largest portion of computation time (80 %) requires the computation of the dot product, in which the matrix of the selected features and interactions are multiplied with the generated attribute values. For illustration, this operation requires approximately 1500s for a matrix with 125,000 rows and 653 columns. The remaining 20 % of computation time are mainly consumed by calculating and ranking the fitness of all solutions.

Discussion. To answer RQ2, the results of our performance experiments show that THOR scales linearly with the number of sampled configurations, which are used to compute $\mathcal{D}_O^{\mathcal{V}}$. All remaining tasks have only a constant influence on execution time. Son, the linear dependence enables THOR to handle large-scale variability models.

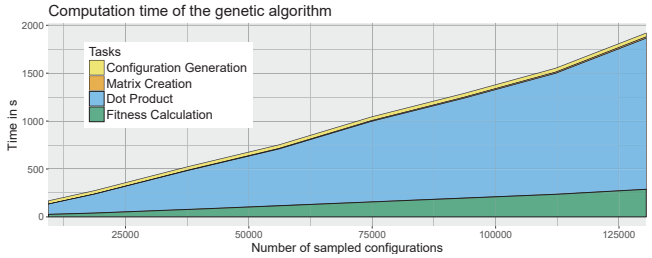


Figure 6: Processing time fractions with an increasing sample set, before and during executing the genetic algorithm.

5.3 Threats to Validity

Internal Validity. There are threats resulting from the setting choices on the genetic optimization (GA) and the computational test environment. As for the GA, we adopted the built-in NSGA-II operators plus parameter defaults as provided by the library JMetalCSharp, because the focus was on a comparison based on a standard setup. Systematically studying the effects of alternative GAs, domain-specific operators [12], and parameter tuning is needed to eliminate this threat, entirely. The reported execution times may have been dependent on the machine load. We mitigated this threat by aggregating over 10 independent runs. An environmental threat arises from possible bugs in our implementation. So, we tested for multiple goodness-of-fit metrics and compared the results manually in multiple steps of the GA. Hence, we are convinced that our metrics are correctly implemented. Furthermore, we use existing open-source implementations for some metrics and the GA (JMetalCSharp), which further mitigates this threat.

External Validity. We have used a number of different variability models and distribution profiles for the validity and the scaling experiment. What we did not report here, due to space limitations, is that we performed additional experiments with models taken from the SPLOT repository as well as models generated using BeTTy. These models have different sizes and constraints so that we covered a large spectrum of variability models and the big picture is the same. Further information is available at the supplementary Web site.

6 RELATED WORK

Variability-Model Generators. A popular variability-model generator is provided by SPLOT [22]. SPLOT allows users to specify the number of features, the ratio of alternative and optional features, and the number of cross-tree constraints. SPLOT contains also an online repository of variability models. Most of the papers in our literature study used models from the SPLOT repository. The SPLOT models can be used as an input for Thor.

The closest approach to ours is BeTTy, a Java library for generating variability models [29]. It also supports the assignment of constants as well as the random generation of attribute values following normal and uniform distributions. However, BeTTy (and other generators, as well) does not support the generation of interactions and it is not possible to supply realistic distribution profiles.

Variability-Model Synthesis. Synthesizing realistic variability models has recently gained momentum. There are different approaches that aim at reverse engineering variability models based on textual descriptions and feature dependencies [30], documentation of individual products and their relationships [14], requirements specification [1, 38], and genetic programming [20]. Recent approaches aim at improving the hierarchy within the model, for instance, based on an ontology [7]. All approaches do not consider attribute values. Only recently, Nasr and others have demonstrated that also technical descriptions, such as attribute values, can be extracted from product descriptions [23]. We see this work as an important source to obtain further realistic input distribution profiles.

Distribution Profiles. There are various domains, in which the distribution of data is important and can affect the outcome and behavior of an analysis. The testing community, for instance, is constantly developing novel approaches to find bugs in programs. But, it is always difficult to state to which degree an approach or a test suite can find unknown bugs and—similar to real-world attributed variability models—there are often too few real faults to be used in experiments [2]. Mutation testing is a way to assess the quality of test suites and detection algorithms [16]. The idea is to inject faults into a program such that a developer can evaluate whether the test suite can spot the faults. A related study explored whether faults generated by hand or from mutation operators are representatives of real faults [2]. The study found that one must carefully select the mutation operators and there is danger in using manually seeded faults. We draw a similar conclusion for attribute-value distributions of variability models.

7 CONCLUSIONS

Attributed variability models are used in various areas, but, due to the lack of realistic attribute values, the overwhelming majority of algorithms and tools operating on attributed variability models have been developed and trained on artificial attribute values, ignoring feature interactions. The overarching goal of our work is to make researchers and practitioners aware of this problem, and we strive for a more realistic and robust setting—leaving the comfort zone of artificial attributed variability models. We conducted a literature study with the main outcome that interactions are not considered by state-of-the-art experiments and that attribute values are mostly generated based on artificial distributions. As the first replication of this kind, we reproduced a popular experimental setting of Sayyad et al. [27] and found that, while the key results of the original study hold, feature interactions and varying attribute-value distributions lead to important deviations in solution quality.

As an actionable solution, we provide Thor, a tool for including realistic attribute values and feature interactions into a given variability model. We employ kernel density estimation for this purpose, to rescale an input distribution to the features, interactions, and configurations of a given variability model. Using a genetic algorithm, we adjust the attribute values such that they match the input distributions. In a series of experiments, we demonstrated that (a) using kernel density estimation is key for obtaining a good match with output distributions and that (b) our approach scales linearly with the number of configurations that are used for computing the distribution profile.

ACKNOWLEDGMENTS

Siegmund's and Apel's work are supported by the DFG under the contracts SI 2171/2, AP 206/6, and AP 206/7.

REFERENCES

[1] Vander Alves, Christa Schwanninger, Luciano Barbosa, Awais Rashid, Peter Sawyer, Paul Rayson, Christoph Pohl, and Andreas Rummler. 2008. An Exploratory Study of Information Retrieval Techniques in Domain Analysis. In *Proc. Int. Software Product Line Conference (SPLC)*. IEEE, 67–76.

[2] J. H. Andrews, L. C. Briand, and Y. Labiche. 2005. Is Mutation an Appropriate Tool for Testing Experiments?. In *Proc. Int. Conf. Software Engineering (ICSE)*. ACM, 402–411.

[3] Sven Apel, Sergiy Kolesnikov, Norbert Siegmund, Christian Kästner, and Brady Garvin. 2013. Exploring Feature Interactions in the Wild: The New Feature-interaction Challenge. In *Proc. GPCE Workshop on Feature-Oriented Software Development (FOSD)*. ACM, 1–8.

[4] Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and William R. Stewart. 1995. Designing and Reporting on Computational Experiments with Heuristic Methods. *J. Heuristics* 1, 1 (1995), 9–32.

[5] Thomas Bartz-Beielstein and Mike Preuss. 2010. *The Future of Experimental Research*. Springer, 17–49.

[6] Don Batory. 2005. Feature Models, Grammars, and Propositional Formulas. In *Proc. Int. Software Product Line Conference (SPLC)*. Springer, 7–20.

[7] Guillaume Bécan, Mathieu Acher, Benoit Baudry, and Sana Ben Nasr. 2015. Breathing Ontological Knowledge Into Feature Model Synthesis: An Empirical Study. *Empirical Software Engineering* 21, 4 (2015), 1794–1841.

[8] David Benavides, Pablo Trinidad, and Antonio Ruiz-Cortes. 2005. Automated Reasoning on Feature Models. In *Proc. Conf. Advanced Information Systems Engineering (CAiSE)*. Springer, 491–503.

[9] Thorsten Berger, Ralf Rublack, Divya Nair, Joanne M. Atlee, Martin Becker, Krzysztof Czarnecki, and Andrzej Wasowski. 2013. A Survey of Variability Modeling in Industrial Practice. In *Proc. Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*. ACM, 1–8.

[10] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

[11] Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. 2006. *Evolutionary Algorithms for Solving Multi-Objective Problems* (2nd ed.). Springer.

[12] Thelma Elita Colanzi and Silvia Regina Vergilio. 2016. A Feature-Driven Crossover Operator for Product Line Architecture Design Optimization. *Journal of Systems and Software* 121 (2016), 126–143.

[13] Krzysztof Czarnecki, Paul Grünbacher, Rick Rabiser, Klaus Schmid, and Andrzej Wasowski. 2012. Cool Features and Tough Decisions: A Comparison of Variability Modeling Approaches. In *Proc. Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*. ACM, 173–182.

[14] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans. 2013. Feature Model Extraction from Large Collections of Informal Product Descriptions. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 290–300.

[15] Kanpur Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.

[16] Richard A. DeMillo, Richard J. Lipton, and Frederick G. Sayward. 1978. Hints on Test Data Selection: Help for the Practicing Programmer. *Computer* 11, 4 (1978), 34–41.

[17] Omar S. Gómez, Natalia Juristo, and Sira Vegas. 2014. Understanding Replication of Experiments in Software Engineering: A Classification. *Information and Software Technology* 56, 8 (2014), 1033–1048.

[18] Jianmei Guo, Krzysztof Czarnecki, Sven Apel, Norbert Siegmund, and Andrezj Wasowski. 2013. Variability-Aware Performance Prediction: A Statistical Learning Approach. In *Proc. Conf. Automated Software Engineering (ASE)*. IEEE, 301–311.

[19] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. 2015. Combining Multi-objective Search and Constraint Solving for Configuring Large Software Product Lines. In *Proc. Int. Conf. on Software Engineering (ICSE)*. IEEE, 517–528.

[20] Lukas Linsbauer, Roberto Erick Lopez-Herrejon, and Alexander Egyed. 2014. Feature Model Synthesis with Genetic Programming. In *Proc. Int. Symp. on Search-Based Software Engineering (SSBSE)*. Springer, 153–167.

[21] Jia Liu, Don Batory, and Christian Lengauer. 2006. Feature Oriented Refactoring of Legacy Applications. In *Proc. Int. Conf. Software Engineering (ICSE)*. ACM, 112–121.

[22] Marcilio Mendonca, Moises Branco, and Donald Cowan. 2009. SPLOT: Software Product Lines Online Tools. In *Companion Proc. Int. Conf. Object-Oriented Programming, Systems, Languages and Applications (OOPSLA)*. ACM, 761–762.

[23] Sana Ben Nasr, Guillaume Bécan, Mathieu Acher, João Bosco Ferreira Filho, Nicolas Sannier, Benoit Baudry, and Jean-Marc Davril. 2017. Automated Extraction of Product Comparison Matrices from Informal Product Descriptions. *Journal of Systems Software* 124, C (2017), 82–103.

[24] Lina Ochoa, Juliana Alves Pereira, Oscar González-Rojas, Harold Castro, and Gunter Saake. 2017. A Survey on Scalability and Performance Concerns in Extended Product Lines Configuration. In *Proc. Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*. ACM, 5–12.

[25] Gustavo G. Pascual, Roberto E. Lopez-Herrejon, Mónica Pinto, Lidia Fuentes, and Alexander Egyed. 2015. Applying Multiobjective Evolutionary Algorithms to Dynamic Software Product Lines for Reconfiguring Mobile Applications. *Journal of Systems Software* 103, C (2015), 392–411.

[26] Atri Sarkar, Jianmei Guo, Norbert Siegmund, Sven Apel, and Krzysztof Czarnecki. 2015. Cost-Efficient Sampling for Performance Prediction of Configurable Systems. In *Proc. Int. Conf. Automated Software Engineering (ASE)*. IEEE, 342–352.

[27] Abdel S. Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. 2013. Scalable Product Line Configuration: A Straw to Break the Camel's Back. In *Proc. Int. Conf. Automated Software Engineering (ASE)*. IEEE, 465–474.

[28] Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. 2013. On the Value of User Preferences in Search-based Software Engineering: A Case Study in Software Product Lines. In *Proc. Int. Conf. on Software Engineering (ICSE)*. IEEE, 492–501.

[29] Sergio Segura, José A Galindo, David Benavides, José A Parejo, and Antonio Ruiz-Cortés. 2012. BeTTy: Benchmarking and Testing on the Automated Analysis of Feature Models. In *Proc. Int. Workshop on Variability Modelling of Software-intensive Systems (VaMoS)*. ACM, 63–71.

[30] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. 2011. Reverse Engineering Feature Models. In *Proc. Int. Conf. Software Engineering (ICSE)*. ACM, 461–470.

[31] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. 2015. Performance-Influence Models for Highly Configurable Systems. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*. ACM, 284–294.

[32] Norbert Siegmund, Sergiy Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, and Gunter Saake. 2012. Predicting Performance via Automated Feature-Interaction Detection. In *Proc. Int. Conf. Software Engineering (ICSE)*. IEEE, 167–177.

[33] Norbert Siegmund, Marko Rosenmüller, Christian Kästner, Paolo Giarrusso, Sven Apel, and Sergiy Kolesnikov. 2013. Scalable Prediction of Non-functional Properties in Software Product Lines: Footprint and Memory Consumption. *Information and Software Technology* 55, 3 (2013), 491–507.

[34] Norbert Siegmund, Alexander von Rhein, and Sven Apel. 2013. Family-Based Performance Measurement. In *Proc. Int. Conf. Generative Programming and Component Engineering (GPCE)*. ACM, 95–104.

[35] Bernard W. Silverman. 1998. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall/CRC.

[36] N. Smirnov. 1948. Table for Estimating the Goodness of Fit of Empirical Distributions. *The Annals of Mathematical Statistics* 19, 2 (1948), 279–281.

[37] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. 2014. Replication and Comparison of Computational Experiments in Applied Evolutionary Computing: Common Pitfalls and Guidelines to Avoid Them. *Applied Soft Computing* 19 (2014), 161–170.

[38] Nathan Weston, Ruzanna Chitchyan, and Awais Rashid. 2009. A Framework for Constructing Semantically Composable Feature Models from Natural Language Requirements. In *Proc. Int. Software Product Line Conference (SPLC)*. Carnegie Mellon University, 211–220.

[39] Erik Wittern and Christian Zirpins. 2014. Service Feature Modeling: Modeling and Participatory Ranking of Service Design Alternatives. *Software and Systems Modeling* 15, 2 (2014), 1–26.