Performance-Influence Models for Highly Configurable Systems

Norbert Siegmund[1] Alexander Grebhahn[2] Sven Apel[3] Christian Kästner[4]

1 Introduction

The original paper has been published in the proceedings of ESEC/FSE 2015 [SGAK15]. End-users, developers, and administrators are often overwhelmed with the possibilities to *configure* a software system. In most systems today, including databases, Web servers, video encoders, and compilers, hundreds of configuration options can be combined, each potentially with distinct functionality and different effects on quality attributes. The sheer size of the configuration space and complex constraints between configuration options make it difficult to find a configuration that performs as desired, with the consequence that many users stick to default configurations or only try changing an option here or there. This way, the significant optimization potential already present in many of our modern software systems remains untapped. Even domain experts and the developers themselves often do not (fully) understand the performance influences of all configuration options and their combined influence when options interact.

Our goal is to build performance-influence models (and models of other measurable quality attributes, such as energy consumption) that describe how configuration options and their interactions influence the performance of a system (e.g., throughput or execution time of a benchmark). A distinctive feature of our approach is that we consider both binary and numeric options and that we do not solely target prediction accuracy. Performance-influence models are meant to ease *understanding, debugging, and optimization* of highly configurable software systems. For example, a user may identify the best performing configuration from the model and a developer may compare an inferred performance-influence model with her own mental model to check whether the system behaves as expected.

2 Approach

Our approach is to infer a performance-influence model for a given configurable system in a black-box manner, from a series of *measurements* of a set of *sample* configurations using

[1] University of Passau, Germany
[2] University of Passau, Germany
[3] University of Passau, Germany
[4] Carnegie Mellon University, USA

machine learning. That is, we benchmark a given system multiple times in different configurations and learn the influence of individual configuration options and their interactions from the differences between the measurements. Conceptually, a performance-influence model is simply a function from a configuration $c \in \mathscr{C}$ to a performance measure $\Pi : \mathscr{C} \to \mathbb{R}$, where performance can be any measurable property that produces interval-scaled data. All performance-influence models are of the following form:

$$\Pi(c) = \beta_0 + \sum_{i \in \mathscr{O}} \phi_i(c(i)) + \sum_{i..j \in \mathscr{O}} \Phi_{i..j}(c(i)..c(j)) \tag{1}$$

where β_0 represents a minimum, constant base performance shared by all configurations, as determined during learning; $\sum_{i \in \mathscr{O}} \phi_i(c(i))$ represents the sum of the influences of all individual options; $\sum_{i..j \in \mathscr{O}} \Phi_{i..j}(c(i)..c(j))$ is the sum of the influences of all interactions among all options. This structure allows us to easily see the influence of an individual option or an interaction between options from the model.

Learning. We use *stepwise linear regression* to learn the function of a performance-influence model from a sample set of measured configurations. To reduce the dimensionality problem of handling a very large number of options and interactions, we use *feature subset selection* to incrementally learn the model. The key challenge of using linear regression is to identify the relevant terms to be used as independent variables; a term represents the (possibly non-linear) influence of one or multiple configuration options. Conceptually, any combination of options may cause a distinct performance interaction [SKK+12], which would render any learning approach useless, as there is no common pattern. In practice, however, performance behavior is usually more tractable in that only few interactions contribute substantially to the overall performance. In our previous work, we found that relevant interactions do not emerge randomly between configuration options, but form a hierarchy [SvRA13]. Thus, we perform our learning hierarchically and incrementally: Starting with an empty model, our algorithm selects one term in each iteration until improvements of model accuracy become marginal or a threshold for expected accuracy is reached. The term to be added stems from a number of candidate terms. The initial candidates are only the individual option influences, which are then extended by candidates representing interactions between options that have been found already to contribute to performance, and additional functions (e.g., logarithmic or quadratic) representing the influence of numeric options.

Sampling. We divide the configuration space along binary and numeric configuration options and apply structured sampling heuristics to them. For binary sampling, we use heuristics developed in previous work that aim at selecting configurations such that we can learn the influences of individual options and of pair-wise interactions. For sampling numeric options, we use a number of experimental designs, including fractional factorial designs and optimal designs. We found that the Plackett-Burman design provides a sweet spot between measurement effort and accuracy of the learned model. The separately selected configurations for binary and numeric options are combined using the cross product.

Experiments. Our approach is able to build reasonably accurate performance models of configuration spaces of real-world systems, including compilers, multi-grid solvers, and video encoders. In a series of experiments with configurable systems with up to 10^{31}

configurations, we found that few measurements are sufficient to build fairly accurate models (19 % prediction error, on average). The performance-influence models learned by our approach can explain the performance variation between configurations with a few dozen terms describing the influence of individual options and another dozen terms describing interactions. Finally, while accuracy is important, simple models are important, too. Views on a performance-influence model can be used to isolate influences of individual options and their interactions.

References

[SGAK15] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. Performance-Influence Models for Highly Configurable Systems. In *Proc. ESEC/FSE*, pages 284–294. ACM, 2015.

[SKK+12] Norbert Siegmund, Sergiy Kolesnikov, Christian Kästner, Sven Apel, Don Batory, Marko Rosenmüller, and Gunter Saake. Predicting Performance via Automated Feature-Interaction Detection. In *Proc. ICSE*, pages 167–177. IEEE, 2012.

[SvRA13] Norbert Siegmund, Alexander von Rhein, and Sven Apel. Family-Based Performance Measurement. In *Proc. GPCE*, pages 95–104. ACM, 2013.