Distance-Based Sampling of Software Configuration Spaces

Christian Kaltenecker[1], Alexander Grebhahn[2], Norbert Siegmund[3], Jianmei Guo[4], Sven Apel[5]

Abstract: Configurable software systems provide configuration options to adjust and optimize their functional and non-functional properties. However, to obtain accurate performance predictions, a *representative* sample set of configurations is required. Different sampling strategies have been proposed, which come with different advantages and disadvantages. In our experiments, we found that most sampling strategies do not achieve a good coverage of the configuration space with respect to covering relevant performance values. That is, they miss important configurations with distinct performance behavior. Based on this observation, we devise a new sampling strategy that is based on a distance metric and a probability distribution to spread the configurations of the sample set across the configuration space. To demonstrate the merits of distance-based sampling, we compare it to state-of-the-art sampling strategies on 10 real-world configurable software systems. Our results show that distance-based sampling leads to more accurate performance models for medium to large sample sets.

Keywords: Distance-Based Sampling; Configuration Sampling; Configurable Systems; Performance Modeling

1 Summary

Modern software systems can be configured by users to adapt them to specific devices, operating systems, and requirements. Configuration options often have a significant influence on non-functional properties, such as performance or energy consumption. Despite the benefits of configurability, identifying the performance-optimal configuration for a given setting is often a non-trivial task, due to the sheer size of configuration spaces [Xu15] and potential interactions among configuration options [KKB08, Ko18]. To identify the performance-optimal configuration of a configuration space, one can measure the performance of every valid configuration of the software system in a brute-force manner, which usually does not scale.

To avoid measuring all configurations, machine-learning techniques, such as multiple linear regression [Si12, Si15] and classification and regression trees [Gu13, Na17, Na18], have been used to learn a performance model based on a set of valid configurations,

[1] Saarland University, Germany
[2] adesso SE, Germany
[3] University of Weimar, Germany
[4] Alibaba Group, China
[5] Saarland University, Germany

doi:10.18420/SE2020_17

called the *sample set*. A performance model allows us to predict the performance of a configuration, and it can be used by an optimizer to determine the performance-optimal configuration [He15, Na18]. To create an accurate performance model, the sample set must be well-chosen, which is a non-trivial task especially when no domain knowledge is available. In essence, selecting a small, valid, and representative sample set is key to efficiency and accuracy of performance prediction, as performance measurements are usually costly in practice [Gu18].

Several sampling strategies have been proposed, which differ in their methods of selecting the sample set: (1) at random [GD06, Ch14], (2) by using an off-the-shelf constraint solver [He15], or (3) by aiming at a certain coverage criterion (e.g., selecting each configuration option, at least once) [Le08, JHF12, Ma13]. Naturally, all sampling strategies come with advantages and disadvantages. The main idea is often to cover the configuration space such that one obtains a *representative* sample set, which, ideally includes both influential configuration options and interactions among options relevant to performance, so that accurate performance models can be learned.

In general, a uniform coverage of the configuration space is desirable to obtain a representative sample set when no prior knowledge is available, since it tends to be unbiased when covering the configuration space. However, it is far from trivial to ensure unbiased uniformity if there are non-trivial constraints among configuration options. To achieve the goal in a light-weight way, we propose a new sampling strategy, called *distance-based sampling*, that addresses the shortcomings of existing strategies. The key idea behind distance-based sampling is to produce a sample set that covers the configuration space as uniformly as possible (or following another given probability distribution). To this end, distance-based sampling relies on a *distance metric* and assigns each configuration a *distance value*. It further relies on a *discrete probability distribution* to select configurations according to their distance values from the configuration space. It differs from other sampling strategies in that (1) it spreads the selected configurations across the configuration space according to a given probability distribution and is able to resemble the performance distribution of the whole population, (2) it does not require an analysis on the whole population, and (3) it uses a constraint solver for efficiency while avoiding locally-clustered sample sets [Ka19].

Our results demonstrate that distance-based sampling, when used in combination with a diversity optimization, leads to significantly lower error rates than state-of-the-art strategies, especially for larger sample sizes (t=2, t=3), and the predictions are more stable than solver-based sampling with respect to multiple runs using different random seeds. Our results demonstrate that, based on a distance metric and a probability distribution, we can effectively sample diverse configurations across the configuration space and without the need for a whole-population analysis, which makes random sampling unfeasible for highly configurable software systems. This work provides a new view on sampling based on probability distributions and paves the way for further research in this area. For instance, using other metrics or distributions could lead more accurate predictions or improve the prediction robustness [Ka19].

Bibliography

[Ch14] Chakraborty, Supratik; Fremont, Daniel J; Meel, Kuldeep S; Seshia, Sanjit A; Vardi, Moshe Y: Distribution-Aware Sampling and Weighted Model Counting for SAT. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI). AAAI Press, pp. 1722–1730, 2014.

[GD06] Gogate, Vibhav; Dechter, Rina: A New Algorithm for Sampling CSP Solutions Uniformly at Random. In: Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP). Springer, pp. 711–715, 2006.

[Gu13] Guo, J.; Czarnecki, K.; Apel, S.; Siegmund, N.; Wasowski, A.: Variability-Aware Performance Prediction: A Statistical Learning Approach. In: Proceedings of the International Conference on Automated Software Engineering (ASE). IEEE, pp. 301–311, 2013.

[Gu18] Guo, Jianmei; Yang, Dingyu; Siegmund, Norbert; Apel, Sven; Sarkar, Atrisha; Valov, Pavel; Czarnecki, Krzysztof; Wasowski, Andrzej; Yu, Huiqun: Data-Efficient Performance Learning for Configurable Systems. Empirical Software Engineering, 23(3):1826–1867, 2018.

[He15] Henard, Christopher; Papadakis, Mike; Harman, Mark; Le Traon, Yves: Combining Multi-Objective Search and Constraint Solving for Configuring Large Software Product Lines. In: Proceedings of the International Conference on Software Engineering (ICSE). IEEE, pp. 517–528, 2015.

[JHF12] Johansen, Martin Fagereng; Haugen, Øystein; Fleurey, Franck: An Algorithm for Generating T-Wise Covering Arrays from Large Feature Models. In: Proceedings of the International Software Product Line Conference (SPLC). ACM, pp. 46–55, 2012.

[Ka19] Kaltenecker, Christian; Grebhahn, Alexander; Siegmund, Norbert; Guo, Jianmei; Apel, Sven: Distance-Based Sampling of Software Configuration Spaces. In: Proceedings of the International Conference on Software Engineering (ICSE). pp. 1084–1094, 2019.

[KKB08] Kim, Chang Hwan Peter; Kästner, Christian; Batory, Don S: On the Modularity of Feature Interactions. In: Proceedings of the International Conference on Generative Programming and Component Engineering (GPCE). ACM, pp. 23–34, 2008.

[Ko18] Kolesnikov, Sergiy; Siegmund, Norbert; Kästner, Christian; Grebhahn, Alexander; Apel, Sven: Tradeoffs in Modeling Performance of Highly-Configurable Software Systems. Software and Systems Modeling, 2018. Online first: `http://rdcu.be/GzLq`.

[Le08] Lei, Yu; Kacker, Raghu; Kuhn, D Richard; Okun, Vadim; Lawrence, James: IPOG/IPOG-D: Efficient Test Generation for Multi-Way Combinatorial Testing. Software Testing, Verification and Reliability, 18(3):125–148, 2008.

[Ma13] Marijan, Dusica; Gotlieb, Arnaud; Sen, Sagar; Hervieu, Aymeric: Practical Pairwise Testing for Software Product Lines. In: Proceedings of the International Software Product Line Conference (SPLC). ACM, pp. 227–235, 2013.

[Na17] Nair, Vivek; Menzies, Tim; Siegmund, Norbert; Apel, Sven: Using Bad Learners to Find Good Configurations. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 257–267, 2017.

[Na18] Nair, Vivek; Yu, Zhe; Menzies, Tim; Siegmund, Norbert; Apel, Sven: Finding Faster Configurations using FLASH. IEEE Transactions on Software Engineering, 2018. Online first: `https://arxiv.org/abs/1801.02175/`.

[Si12] Siegmund, Norbert; Kolesnikov, Sergiy S; Kästner, Christian; Apel, Sven; Batory, Don S; Rosenmüller, Marko; Saake, Gunter: Predicting Performance via Automated Feature-Interaction Detection. In: Proceedings of the International Conference on Software Engineering (ICSE). IEEE, pp. 167–177, 2012.

[Si15] Siegmund, Norbert; Grebhahn, Alexander; Apel, Sven; Kästner, Christian: Performance-Influence Models for Highly Configurable Systems. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 284–294, 2015.

[Xu15] Xu, Tianyin; Jin, Long; Fan, Xuepeng; Zhou, Yuanyuan; Pasupathy, Shankar; Talwadker, Rukma: Hey, You Have Given Me Too Many Knobs!: Understanding and Dealing with Over-Designed Configuration in System Software. In: Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 307–319, 2015.