0.1 Performance-Influence Model

The amount of possible configurations of modern software systems and the complex constrains between them can be overwhelming. Making it difficult to find an optimal configuration, that performs as desired. Performance influence models are meant to ease understanding, debugging and optimization of configurable software systems [?].

A performance influence model consists of several terms that describe the performance of a configuration based on the values of configuration options [?]. Performance, in this context, can be measurable quality attributes such as execution time, memory size or energy consumption. The model describes the influence of several independent variables X, our configuration, on a dependent variable y, our measurable quality attribute. While there are several approaches to predict performance, in general, they all work similarly. They sample a subset of configurations - this is done because it is infeasible to measure performance of all configurations if the configuration space is too big - and learn a model with the sampled configurations.

?] introduces a variability-aware approach to predict a configuration's performance based on random sampling. They use a Classification-And-Regression-Tree [?] to recursively partition the configuration space into smaller segments until they can fit a simple local prediction model into each segment.

?] describes how to create human understandable models based on previous work [?]. They combined binary sampling strategies such as option-wise, negative option-wise and pair-wise, with numerical sampling strategies, such as the Placket-Burman-Design. They then used stepwise linear regression to learn the influence model.

0.1.1 Sampling

The selection of a subset of configurations plays an integral role in almost all methods to predict the performance of a software system. If a configuration option is not present in the sampled subset of configuration a model can not learn the influence of this option. The random selection, random sampling, as used by most machine learning applications proves difficult with configurations. Mainly due to the constraints on the configuration options. Near uniform random sampling in the presence of constraints, although possible, is infeasible [?]. This resulted in developing dedicated sampling strategies for configuration spaces.

Distance based sampling

?] describe a way to randomly sample a configuration space based on a distance metric and a portability distribution, called distance-based sampling. For this, they rely on a distance metric, like the Mannhatten-Distance, to assign each configuration a distance value. By selecting a distance value through a discrete probability distribution and then picking a configuration with the corresponding distance value, they achieve a spread over the configurations resembling the given probability distribution. This allows for uniform random like sampling if the chosen probability distribution is the uniform distribution.

Binary decision diagram-sampling

Although **?**] do not create a model to predict the performance of a system, they implement a way of random sampling configuration spaces through a binary decision diagrams (BDD)[**?**]. They transform a given feature model into BDD, this makes it easy to count the number of valid configuration and thus easy to randomly sample from them. While a BDD allows for random sampling, a major drawback is the creation of it, which may exceed time or memory constrains for some use cases.