Universiteit Leiden
The Netherlands

Sampling strategies in automated algorithm configuration

Anastacio, M.I.A.

Summary

When confronted with complex real-world problems, the first step is often to express them mathematically to make them accessible to a multitude of tools that can potentially be used for solving them. Among those mathematical representations comes one dreaded family: the non-deterministic polynomial-time hard (NP-hard) problems, commonly considered unsolvable in polynomial time. NP-hard problems have two important characteristics: creating the correct solution from scratch is typically intractable, but verifying if a solution is correct does not require as much effort. Thus came heuristic methods: through approximations and trial and error, they can often solve NP-hard problems, even large ones, within an acceptable timeframe.

Researchers gathered a large variety of methods to solve increasingly complex problems. Once the proper tool has been selected, which in itself is already the subject of an entire area of research, the next step is to decide how best to use it. Heuristic algorithms come with various parameters corresponding to switches and knobs one can play with until the fastest path to a solution is found. They allow, for example, to activate pre-processing steps, to select among several metrics, to decide the direction a search process should branch first, or to define the probability of restarting from a random position. All these can be tailored to the characteristics of the problem instances at hand, and setting them correctly can make the difference between solving a problem in minutes, hours, or weeks. To optimise the performance of such algorithms by setting the values of their parameters is an optimisation problem known as the automated algorithm configuration problem. Over the last twenty years, researchers have developed heuristic algorithms (referred to as configurators) to configure their heuristic algorithms (referred to as solvers). Optimisation methods based on various paradigms — such as evolutionary algorithms, racing, and Bayesian optimisation — have provided the bases for configuring the inner workings of algorithms from various fields of application. Those configurators, more specifically their strategies to sample

parameter values and problem instances, are the main topic of study in the present thesis. In particular, we are interested in their application to reduce the running time of solvers for NP-hard problems.

We start by assessing the state of the art, evaluating 7 different approaches on 20 datasets and configuration scenarios from the literature. We find configurators perform differently depending on the tasks, but also that recent ones do not always significantly outperform older approaches. This shows that there has been limited development in the state of the art in the last few years with running time optimisation. Indeed, many recent methods are tailored toward the specific problem of hyperparameter optimisation for machine learning models, which focus on solution quality. We also listed the characteristics of the configuration scenarios and found no straightforward link between them and the performance of the different configurators. Deciding which configurator would be best on which scenario is a difficult problem by itself.

Then, we focus our attention and the sampling distribution of new parameter values, around the default value. This value is typically given with solvers for NP-hard problems of interest in our work. We find that those values typically defined by the developer based on their own understanding of their algorithms nor on experiments on their own set of benchmarks, provide a strong prior in the sampling of new values. We show that using this prior, we can improve the performance of the state-of-the-art configurators based on Bayesian optimisation, SMAC.

After sampling parameter values, the configurator would sample the problem instances on which to test those parameter values. We developed several instance selection approaches based on the literature to decide which instances the solver should try to solve first, and to determine quickly if a set of parameter values leads to better performance. We evaluate them in several cases: comparing two algorithms, two parametrised version of one algorithm, and based on different amounts and types of prior information to represent several steps of the configuration process in our experiments. Then, we integrated them into SMAC and evaluated this new version of SMAC with instance selection. We reached better performances than ever reached before. However, it is difficult to decide which method should be used when, and these methods thus need more research before being applicable in practice.

Overall, in this thesis, we evaluated several methods to sample both new values of parameters and new instances on which to test them based on prior information either from the developer or from the data gathered so far. We showed that better leveraging this information allows the configurators to be more efficient.