Exploitation of Default Parameter Values in Automated Algorithm Configuration

Marie Anastacio[1] , **Chuan Luo**[1,2] , **Holger Hoos**[1,3]
[1]LIACS, Leiden University, Leiden, The Netherlands
[2]Microsoft Research, Beijing, China
[3]University of British Columbia (UBC), Vancouver, Canada
m.i.a.anastacio@liacs.leidenuniv.nl

Abstract

Over the last few years, automated algorithm configuration methods have achieved impressive success for a broad range of prominent AI problems. This increasingly encourages algorithm developers to expose parameters to the automated optimisation process, leading to substantially larger configuration spaces that are more challenging to search. In most cases, when exposing parameters, algorithm designers choose default values based on their intuition, understanding of the underlying behaviour involved and at least limited experimentation.

In this work, we first demonstrate that state-of-the-art automated algorithm configuration procedures moderately exploit information provided by default parameter values. Then, based on insights given by previous work on parameter response landscapes, we introduce two simple techniques to utilise this information by focusing the search around the default parameter values. By reducing the range of possible values, and thus the size of the search space, we improve the efficiency of prominent automatic configuration procedures for a broad range of widely used configuration scenarios. Specifically, our reduction techniques increases the efficacy of SMAC for 15 out of the 20 configuration scenarios we studied, that of GGA++ for 7 out of the 9 scenarios, and that of irace for a scenario on which previously only minor improvements could be achieved.

1 Introduction

The performance of state-of-the-art algorithms for many AI problems critically depends on the settings of certain parameters [Hutter *et al.*, 2009]. Traditionally, algorithm developers have manually tuned these parameters to find configurations that perform well across a range of benchmarks. However, this manual configuration process tends to be tedious and inefficient. For the past decade, automated algorithm configuration has emerged as an effective alternative to this manual approach. Prominent examples of general-purpose automatic algorithm configurators include irace [López-Ibáñez *et al.*, 2016], ParamILS [Hutter *et al.*, 2009], GGA/GGA++ [Ansótegui *et al.*, 2009; 2015] and SMAC [Hutter *et al.*, 2011]. These fully automated configurators are based on advanced machine learning and optimisation methods and have been demonstrated to be strikingly effective across a broad range of challenging problems of high significance to AI, including propositional satisfiability (SAT) (see, *e.g.*, [Hutter *et al.*, 2017]), mixed integer programming (MIP) (see, *e.g.*, [Hutter *et al.*, 2010]), automated planning (see, *e.g.*, [Fawcett *et al.*, 2011]) and supervised machine learning (ML) (see, *e.g.*, [Thornton *et al.*, 2013; Kotthoff *et al.*, 2017]).

The availability of effective automated algorithm configurators encourages algorithm designers to expose design choices as parameters. This has lead to a design paradigm known as programming by optimisation (PbO) [Hoos, 2012]. PbO is based on the idea that by avoiding premature choices in the design of algorithms, we can let automated methods, such as general-purpose algorithm configurators, make design choices specifically adapted to a use case and thus achieve better performance. However, the size of the combinatorial configuration spaces in this context grows exponentially with the number of exposed parameters, which can pose considerable challenges for state-of-the-art configurators.

Meanwhile, the choice of default parameter settings still tends to involve substantial human intuition, experience and at least limited manual experimentation. We conjecture that these default settings contain valuable information that can be exploited in the automatic configuration process.

In this work, we explore the degree to which this information is leveraged by state-of-the-art algorithm configurators. Our empirical results indicate that the degree to which default parameter settings matter highly depends on the configurator in question. While SMAC does not benefit much, irace tends to critically depend on good defaults. To take better advantage of the default settings, we propose two simple techniques that leverage the default values of continuous parameters to reduce the size of a given configuration space. Using these generic reduction techniques, we obtain improved results for two of the three prominent general-purpose configurators we study.

The remainder of this paper is structured as follows. Section 2 introduces the algorithm configuration problem, three state-of-the-art, general-purpose procedures for solving it,

and the two research questions we investigate in the following. Section 3 describes the configurators, algorithms and benchmarks used in our study, as well as our experimental setup. In Section 4, we compare the performance of the configurators with and without giving them access to the default parameter values provided in our benchmark scenarios. In Section 5, we outline our two parameter reduction techniques. In Section 6, we show how the performance of SMAC, one of the most widely used algorithm configurators, is affected by these reduction techniques, and investigate how these observation generalise to other configurators. Finally, Section 7 draws general conclusions and insights from our results.

2 Automatic algorithm configuration

The algorithm configuration problem can be defined as follows. Given:

- a target algorithm A with parameters $p_1; p_2; :::; p_k$, a domain D_j of possible values and a default value d_j P D_j for each parameter p_j;
- a configuration space C, containing all valid combinations of parameter values of A;
- a set of problem instances I;
- a performance metric m that measures the performance of a configuration c P C of target algorithm A on I;

find c^* P C that optimises the performance of A on instance set I, according to metric m.

There are different types of parameters. Categorical parameters have an unordered, finite set of values; they are often used to select between several heuristic components or mechanisms.Boolean parameters typically activate or deactivate optional algorithm components. Numerical, integer- or real-valued parameters are often used to quantitatively control performance-relevant aspects of a given target algorithm, such as a learning rate. Some parameters may conditionally depend on others, in that they are only active for certain values of other (often categorical) parameters.

In recent years, much work has been done on automatic algorithm configuration, resulting in several general-purpose automatic algorithm configurators, based on different approaches. The configurator irace [Birattari *et al.*, 2010; López-Ibáñez *et al.*, 2016] leverages racing methods based on a statistical test in conjunction with model-based sampling of configurations. GGA [Ansótegui *et al.*, 2009] and GGA++ [Ansótegui *et al.*, 2015] are based on gender-based evolutionary algorithm. SMAC [Hutter *et al.*, 2011] is based on a sequential model-based (aka Bayesian) optimisation approach and utilises a random forest model and an expected improvement criterion to produce promising configurations.

Default parameter values. Algorithm developers usually provide a default parameter configuration that has been chosen to perform reasonably well across a broad range of problem instances. A recent study of the dependence of algorithm performance on the setting of numerical parameters has found that these configuration landscapes tend to be surprisingly benign [Pushak and Hoos, 2018]. This suggests that even limited manual tuning may produce useful information about promising parameter values. At the same time, it has been shown that by starting SMAC from a performance model learned on a another set of benchmark instances, considerable speedups can be obtained compared to SMAC runs without this kind of prior knowledge [Lindauer and Hutter, 2018]. This suggest that knowledge regarding high-quality configurations can be transferred between sets of problem instances. Together, these observations indicate that default parameter configurations may contain valuable information that can be exploited for automatic algorithm configuration. In light of this, we consider the following research question :

(RQ1) How much impact does the default configuration have on current configurators? To obtain insight into the degree to which default parameter values are exploited by state-of-the-art algorithm configuration procedures, we compare performance of two prominent, yet very different, configurators, SMAC and irace, with or without using the default setting to initialize the configuration process.

Search space reduction. Recently, it has been demonstrated that automatic configuration with irace can be sped up by focusing the search process on specific areas of a given configuration space, by means of applying transformations to one real-valued parameter of a simulated annealing algorithm [Franzin *et al.*, 2018]. This work also produced evidence that poorly chosen transformation can degrade the automatic configuration process. As mentioned above, we have reason to believe that promising regions of values for individual parameters can often be found around expert-determined default values, and we aim to investigate this hypothesis. For scenarios with large configuration spaces in which high-performance configurations are difficult to find, we conjecture that searching configurations inside a reduced search space, obtained by restricting the ranges of certain parameters, can be more effective than searching the full configuration space. This leads us to our second research question:

(RQ2) Can we leverage the information contained in the default by reducing the range of the parameters? Specifically, we introduce two techniques that limit the search to a neighbourhood of the default values and evaluate their impact on the performance of several state-of-the-art configurators, across a broad range of configuration scenarios.

3 Experimental setup

To answer our research questions, we selected 20 configuration scenarios from the general algorithm configuration library AClib [Hutter *et al.*, 2014], covering four prominent AI problems: SAT, MIP, automated planning and automated machine learning. The algorithms and benchmarks included in AClib are well known and widely used in the automatic algorithm configuration literature. This section describes the target algorithms, benchmark instance sets and algorithm configurators, as well as our experimental protocol and execution environment.

3.1 Target algorithms

In our experiments, we used three solvers for propositional satisfiability (SAT), one automated planning system, one