Combining Sequential Model-Based Algorithm Configuration with Default-Guided Probabilistic Sampling

Marie Anastacio
m.i.a.anastacio@liacs.leidenuniv.nl
Universiteit Leiden
Leiden, The Netherlands

Holger H. Hoos
hh@liacs.nl
Universiteit Leiden
Leiden, The Netherlands
University of British Columbia
Vancouver, Canada

ABSTRACT

General-purpose automated algorithm configuration procedures have enabled impressive improvements in the state of the art in solving challenging problems from AI, operations research and other areas. The most successful configurators combine multiple techniques to search vast combinatorial spaces of parameter settings for a given algorithm as efficiently as possible. Specifically, two of the most prominent general-purpose algorithm configurators, SMAC and irace, can be seen as combinations of Bayesian optimisation and racing, and of racing and an estimation of distribution algorithm, respectively. Here, we investigate an approach that combines all three of these techniques into one single configurator, while exploiting prior knowledge contained in expert-chosen default parameter values. We demonstrate significant performance improvements over irace and SMAC on a broad range of running time optimisation scenarios from AClib.

CCS CONCEPTS

• **Computing methodologies → Optimization algorithms**; **Machine learning**; • **General and reference → Empirical studies**;

KEYWORDS

Parameter tuning and algorithm configuration, Combinatorial optimization, Artificial intelligence, Machine learning

ACM Reference Format:
Marie Anastacio and Holger H. Hoos. 2020. Combining Sequential Model-Based Algorithm Configuration with Default-Guided Probabilistic Sampling. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3389999

1 INTRODUCTION

The performance of state-of-the-art algorithms critically depends on parameter settings [7], and manual configuration of these parameters tends to be tedious and inefficient. For the past decade, automated methods for configuring those algorithms have been broadly established as an effective alternative to a manual approach.

The algorithm configuration problem (see, *e.g.*, [4]) can be defined as follows. Given: a target algorithm A with parameters $p_1, ..., p_k$; a domain D_j of possible values and a default value $d_j \in D_j$ 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 $c \in C$ of target algorithm A on I; find $c^* \in C$ that optimises the performance of A on instance set I, according to m.

Prominent examples of general-purpose automated algorithm configurators include irace [10], ParamILS [7], GGA/GGA++ [2, 3] and SMAC [6]. State-of-the-art automated configurators are strikingly effective (see, *e.g.*, [5, 8]), leveraging combinations of sophisticated methods, such as racing, estimation of distribution algorithms, Bayesian optimisation and model-free stochastic local search. In particular, SMAC and irace can be seen as combinations of Bayesian optimisation and racing, and of racing and an estimation of distribution algorithm, respectively. We propose a first approach combining these three techniques into a single general-purpose automated algorithm configuration procedure that shows significant performance improvements over irace and SMAC, the two state-of-the-art configurators it is based on.

2 OUR APPROACH

The search space explored by automated algorithm configurators grows exponentially with the number of parameters. Our recent study [1] shows that reducing its size to focus the search on the given default values can lead to significant performance improvements for SMAC, despite excluding regions that could be relevant in specific cases. Our new configurator, SMPS, exploits this insight by including probabilistic sampling in the context of sequential model-based optimisation; it combines the sequential model-based approach of SMAC with probabilistic sampling inspired by irace.

SMPS uses the same sequential model-based optimisation approach (also known as Bayesian optimisation) as SMAC. It creates a random forest model to predict the performance of the given target algorithm for arbitrary parameter configuration, based on performance data collected from specific target algorithm runs. In each iteration, it selects a candidate configuration and runs it on some problem instances from the given training set. The random forest model is then updated with the observed performance. This process continues until a given time budget (usually specified in terms of wall-clock time) has been exhausted.

Algorithm 1 outlines our new configuration method. The step of interest for us is the way SMAC selects new configurations. It does so via two different mechanisms. On the one hand, it performs local search from many randomly generated configurations on its

Algorithm 1 SMPS

C: configuration space. c_d: the default configuration.
C_{rand}, C_{prom} and C_{new}: sets of configurations.
$\mathcal{R}$: target algorithm runs performed. $\mathcal{M}$: performance model.

```
 1: $inc \leftarrow c_d$
 2: $\mathcal{R} \leftarrow run\,(c_d)$
 3: while Budget not exhausted do
 4:     $\mathcal{M} \leftarrow update\,(\mathcal{M}, \mathcal{R})$
 5:     $C_{prom} \leftarrow sample\_configurations\,(C)$
 6:     $C_{prom} \leftarrow local\_search\,(\mathcal{M}, C)$
 7:     $C_{rand} \leftarrow sample\_configurations\,(C)$
 8:     $C_{new} \leftarrow interleave\,(C_{rand}, C_{prom})$
 9:     $inc \leftarrow intensify\,(C_{new}, inc)$
10: end while
11: Return $inc$
```

random forest model (line 6). On the other hand, it selects new configurations uniformly at random from the entire (usually vast) configuration space (line 7). Those two sets are then interleaved and raced against the current incumbent (line 9). For continuous and integer parameters, we change the sampling distribution in those two cases. The range is normalized to $[0, 1]$, then sampled using truncated normal distributions with means given by the default parameter values and variance set to a fixed value of 0.05, based on preliminary experiments. For categorical parameters, we still sample uniformly at random.

3 EXPERIMENTAL SETUP AND RESULTS

We compare SMPS to SMAC and irace on 16 configuration scenarios for running time optimisation from AClib [9], covering SAT, AI planning and MIP. Because algorithm configurators are randomised, it is common practice to perform multiple independent runs and report the best configuration (evaluated on the training instances) as the final result. To estimate the probability distribution of the quality of the result produced by this standard protocol, we performed 24 independent runs per configurator on each scenario. Then we repeatedly sampled 8 runs uniformly at random and identified the best of these according to performance on the training set. We compared the medians of these empirical distributions, using a one-sided Mann-Whitney U-test ($\alpha = 0.05$) to assess statistical significance of observed performance differences.

The median PAR10 values obtained are shown in table 1. Note that the missing results for CPLEX on RCW2 and REG200 are due to the fact that less than 10 out of the 24 irace runs accepted to start as it considered the configuration budget too low. Overall, these results indicate clearly that SMPS represents a significant improvement over both baselines, and hence an advance in the state of the art in automated algorithm configuration.

The empirical cumulative distribution functions over the individual configurator runs (without applying the standard protocol) is shown in fig. 1 for one scenario on which SMPS achieves better results than SMAC (fig. 1a) and one in which irace performs better than the two other configurators (fig. 1b).

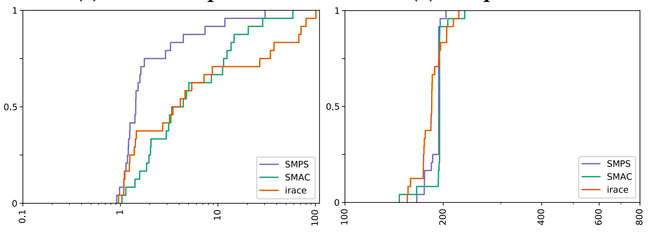
4 FUTURE WORK

We plan to extend our approach with a mechanism that adapts the median and variance of the sampling distributions, such that we handle cases in which defaults are poorly chosen.

Table 1: Results for SMAC, SMPS and irace; median PAR10 (in CPU sec); best results are underlined, while boldface indicates results that are statistically tied to the best, according to a one-sided Mann-Whitney test ($\alpha = 0.05$).

Solver	Benchmark	Default	SMAC	irace	SMPS
Clasp	CF	193.87	193.00	**174.00**	174.87
	LABS	745.74	837.93	847.10	**745.49**
	UNSAT	0.885	0.362	**0.359**	0.360
Lingeling	CF	327.00	**261.06**	328.21	**249.33**
	LABS	873.67	866.99	959.353	**857.12**
	UNSAT	2.41	1.59	2.36	**1.50**
SpToRiss	CF	472.78	**226.51**	236.61	**218.60**
	LABS	911.25	857.67	**846.40**	855.34
	UNSAT	22.26	1.56	1.56	**1.46**
LPG	Depots	34.68	1.14	1.08	**0.98**
	Satellite	22.40	**5.32**	8.04	6.62
	Zenotravel	29.64	**2.61**	2.93	2.85
CPLEX	CLS	4.06	3.36	3.91	**2.95**
	COR-LAT	24.81	19.86	**10.04**	16.35
	RCW2	82.51	**71.98**	–	81.94
	REG200	13.08	5.56	–	**5.05**

Figure 1: Cumulative distribution functions for PAR10 scores (in CPU seconds) over independent configurator runs on the testing set.



(a) LPG on Depots **(b) Clasp on CF**

REFERENCES

[1] M. Anastacio, C. Luo, and H. Hoos. 2019. Exploitation of Default Parameter Values in Automated Algorithm Configuration. In *Workshop Data Science meets Optimisation (DSO), IJCAI 2019.*

[2] C. Ansótegui, Y. Malitsky, H. Samulowitz, M. Sellmann, and K. Tierney. 2015. Model-Based Genetic Algorithms for Algorithm Configuration. In *Proc. IJCAI 2015.* 733–739.

[3] C. Ansótegui, M. Sellmann, and K. Tierney. 2009. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In *Proc. CP 2009.* 142–157.

[4] H. H. Hoos. 2012. Automated Algorithm Configuration and Parameter Tuning. In *Autonomous Search*, Y. Hamadi, E. Monfroy, and F. Saubion (Eds.). Springer, 37–71.

[5] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2010. Automated Configuration of Mixed Integer Programming Solvers. In *Proc. CPAIOR 2010.* 186–202.

[6] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *Proc. LION 5.* 507–523.

[7] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. 2009. ParamILS: An Automatic Algorithm Configuration Framework. *J. Artif. Intell. Res.* 36 (2009), 267–306.

[8] F. Hutter, M. Lindauer, A. Balint, S. Bayless, H. H. Hoos, and K. Leyton-Brown. 2017. The Configurable SAT Solver Challenge (CSSC). *Artif. Intell.* 243 (2017), 1–25.

[9] F. Hutter, M. López-Ibáñez, C. Fawcett, M. T. Lindauer, H. H. Hoos, K. Leyton-Brown, and T. Stützle. 2014. AClib: A Benchmark Library for Algorithm Configuration. In *Proc. LION 8 (LNCS)*, Vol. 8426. 36–40.

[10] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, and M. Birattari. 2016. The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives* 3 (2016), 43–58.