AutoFolio: An Automatically Configured Algorithm Selector (Extended Abstract) *

Marius Lindauer and **Frank Hutter**
University of Freiburg
{lindauer,fh}@cs.uni-freiburg.de

Holger H. Hoos
Leiden University
University of British Columbia
hh@liacs.nl

Torsten Schaub
University of Potsdam
INRIA Rennes
torsten@cs.uni-potsdam.de

Abstract

Algorithm selection (AS) techniques – which involve choosing from a set of algorithms the one expected to solve a given problem instance most efficiently – have substantially improved the state of the art in solving many prominent AI problems, such as SAT, CSP, ASP, MAXSAT and QBF. Although several AS procedures have been introduced, not too surprisingly, none of them dominates all others across all AS scenarios. Furthermore, these procedures have parameters whose optimal values vary across AS scenarios. In this extended abstract of our 2015 JAIR article of the same title, we summarize AUTOFOLIO, which uses an algorithm configuration procedure to automatically select an AS approach and optimize its parameters for a given AS scenario. AUTOFOLIO allows researchers and practitioners across a broad range of applications to exploit the combined power of many different AS methods and to automatically construct high-performance algorithm selectors. We demonstrate that AUTOFOLIO was able to produce new state-of-the-art algorithm selectors for 7 well-studied AS scenarios and matches state-of-the-art performance statistically on all other scenarios. Compared to the best single algorithm for each AS scenario, AUTOFOLIO achieved average speedup factors between 1.3 and 15.4.

1 Introduction

Over the last decade, tremendous progress in Boolean constraint solving technology has brought benefits to several areas within AI. In all these areas, multiple algorithms with complementary solving strategies exist, and none dominates all others on all kinds of problem instances. This fact can be exploited by algorithm selection (AS) [Rice, 1976] methods, which use characteristics of individual problem instances (so-called *instance features*) to choose a promising algorithm for each instance. Algorithm selectors have been shown empirically to improve the state of the art for solving heterogeneous instance sets and, as a result, have won many prizes at competitions.

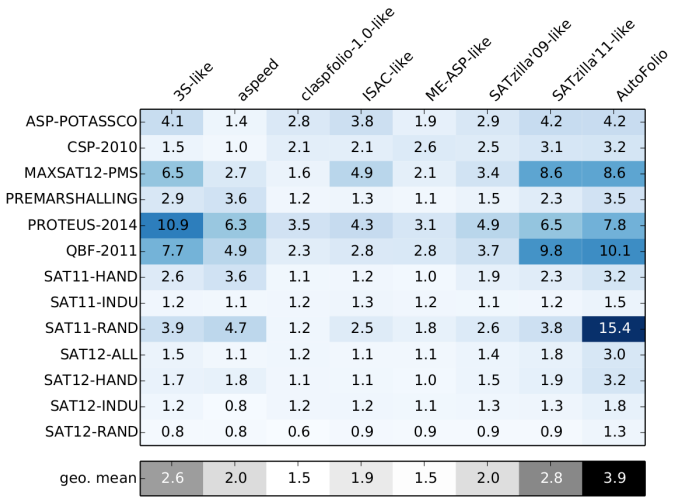

Figure 1: Factors by which the selection approach re-implemented in CLASPFOLIO 2 outperformed the single best algorithm in the 13 ASLIB scenarios w.r.t. penalized average running time (PAR10, which counts each timeout as 10 times the given running time cutoff). These results are for 10-fold cross-validation, ignoring test instances that were not solved by any solver. The last row shows the geometric mean over all 13 scenarios.

Figure 1 illustrates the performance benefits these existing selection strategies (as realized in CLASPFOLIO 2 [Hoos *et al.*, 2014]) yield across the wide range of AS benchmarks in the Algorithm Selection Library [Bischl *et al.*, 2016]. We observe that each approach has strengths and weaknesses on different scenarios. The SATZILLA'11-like approach [Xu *et al.*, 2011] performs best overall, but only achieves better performance than the other approaches considered on 8 out of the 13 scenarios, with 3S [Kadioglu *et al.*, 2011], ASPEED [Hoos *et al.*, 2015] or ISAC [Kadioglu *et al.*, 2010] yielding better performance in the remaining cases.

All these selection methods make use of machine learning techniques, whose performance is known to depend on hyper-parameter settings (e.g., in the case of an SVM, the kernel, kernel hyper-parameter and soft margin; cf. [Bergstra *et al.*,

*This paper is an invited extended abstract of our 2015 JAIR article [Lindauer et al., 2015].

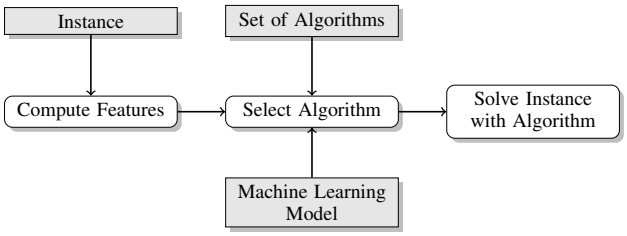

Figure 2: General outline of algorithm selection

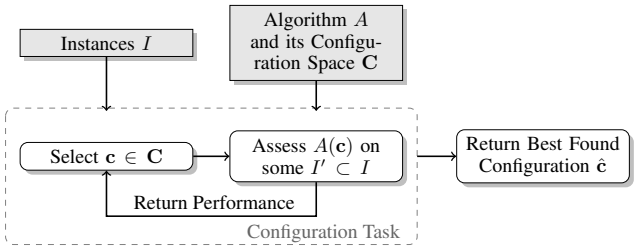

Figure 3: General outline of algorithm configuration

2011; Snoek *et al.*, 2012; Thornton *et al.*, 2013; Feurer *et al.*, 2015a]). We note that the hyper-parameters of the machine learning methods used in Figure 1 were fixed manually, based on limited experiments. Therefore, the performance of some of these AS systems could likely be improved by using more carefully chosen hyper-parameter settings.

Facing a new AS problem, we thus have to answer three salient questions: (i) which selection approach to use; (ii) how to effectively set the parameters of the selection approach (and its underlying machine learning method); and (iii) how to make best use of techniques augmenting pure AS, such as pre-solving schedules [Xu *et al.*, 2008; Kadioglu *et al.*, 2011]. Instead of the common, manual trial-and-error approach, we propose to automatically answer these questions by using automated algorithm configuration methods [Hutter *et al.*, 2009] to configure flexible AS frameworks. While the manual approach is error-prone, potentially biased and requires substantial human expert time and knowledge, the approach we introduce here is fully automatic, unbiased, and leverages the full power of a broad range of AS methods. It thus facilitates an easier and more effective use of algorithm selection and makes AS techniques accessible to a broader community.

Specifically, we present AUTOFOLIO, a general approach for automatically determining a strong AS method for a particular dataset. The last column of Figure 1 previews the results obtained with AUTOFOLIO and clearly shows significant improvements over CLASPFOLIO 2 on 10 of the 13 scenarios in ASLIB.

2 Preliminaries on AS and AC

Algorithm Selection. Figure 2 shows the general outline of algorithm selection [Rice, 1976; Smith-Miles, 2008; Kotthoff, 2014]. For a given problem instance, we first compute cheap instance features [Nudelman *et al.*, 2004]; these are numerical characteristics, including simple ones (such as the number of variables or clauses in a SAT instance) and more complex ones (such as statistics gathered from short probing runs of an actual SAT solver on the given instance). Based on these features, a machine learning model is used to select an appropriate algorithm from a set of algorithms to solve the given instance [Huberman *et al.*, 1997; Gomes and Selman, 2001].

Algorithm Configuration. Figure 3 shows a general outline for algorithm configuration methods [Hutter *et al.*, 2009]. Given a parameterized algorithm A with possible parameter settings $\mathbf{C}$, a set of training problem instances I, and a performance metric $m : \mathbf{C} \times I \to \mathbb{R}$, the objective in the algorithm configuration problem is to find a parameter configuration $\mathbf{c} \in \mathbf{C}$ that optimize m across the instances in I. In the following, we assume w.o.l.g. that performance metric m is to be minimized, as is the case for running time or classification error. The configuration procedure (or short *configurator*) iteratively evaluates the performance of parameter configurations $\mathbf{c} \in \mathbf{C}$ (by running A with them on one or more instances in I) and uses the result to decide about the next configurations to evaluate. After a given budget for the configuration process has been exhausted, the configurator returns the best known parameter configuration it found until then.

3 Configuration of Algorithm Selectors

We now present our AUTOFOLIO approach of using algorithm configurators to automatically customize flexible AS frameworks to specific AS scenarios. To apply algorithm configuration in this context, we need to specify a parameterized selector and its configuration space, as well as the performance metric by which we judge its performance.

3.1 Formal Problem Statement

An AS scenario includes algorithms A, problem instances I, performance and feature data $\mathbf{D}$, and a performance metric $m : A \times I \to \mathbb{R}$ to be minimized, with the data split into disjoint sets $\mathbf{D}_{train}$ and $\mathbf{D}_{test}$. Let $S(\mathbf{D}_{train}) : I \to A$ denote the selector learned by the AS system S when trained on the data $\mathbf{D}_{train}$. Then, the performance of S, $P(S)$ is the average performance of the algorithms it selects on the instances in the test data set $\mathbf{D}_{test}$:

$$P(S) = \frac{1}{|\mathbf{D}_{test}|} \sum_{i \in \mathbf{D}_{test}} m(S(\mathbf{D}_{train}), i). \qquad (1)$$

Likewise, we can evaluate the performance of an AS system S_c parameterized by a configuration c as $P(S_c)$. In order to obtain an unbiased estimate of the configured selector's performance, we need to hold back a test set of instances that is not touched during the configuration process. In order to still be able to optimize parameters without access to that test set, the standard solution in machine learning is to partition the training set further, into k cross-validation folds. Overall, we use the following approach for each selection scenario: (i) we split the full set of instances into a training and a test set

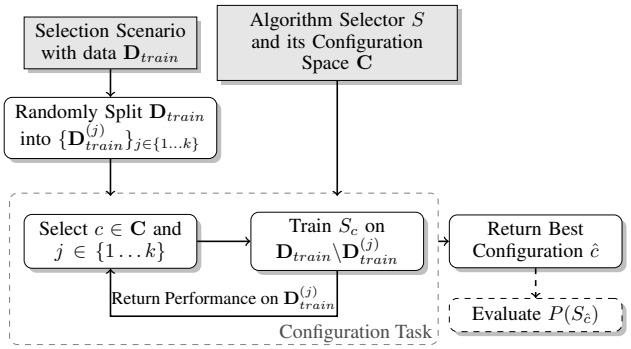

Figure 4: General outline of AUTOFOLIO

and (ii) we partition the training data further into k folds (in our experiments, we use $k = 10$), which are used as follows.

Let $\mathbf{D}^{(1)}_{train}, \dots, \mathbf{D}^{(k)}_{train}$ be a random partition of the training set $\mathbf{D}_{train}$. The cross-validation performance $CV(S_c)$ of S_c on the training set, which is optimized during configuration, is then:

$$CV(S_c) = \frac{1}{k}\sum_{j=1}^{k}\frac{1}{|\mathbf{D}^{(j)}_{train}|}\sum_{i\in\mathbf{D}^{(j)}_{train}} m(S_c(\mathbf{D}_{train}\backslash\mathbf{D}^{(j)}_{train}), i)$$

We minimize $CV(S_c)$ with an algorithm configuration procedure to yield

$$\hat{c} \in \arg\min_{c\in\mathbf{C}} CV(S_c).$$

We then evaluate the generalization performance of $\hat{c}$ by training a selector $S_{\hat{c}}$ with it on the entire training data and evaluating $P(S_{\hat{c}})$ on $\mathbf{D}_{test}$, as defined in Equation 1.

Following Thornton *et al.* (2013), we use each of the k folds $\mathbf{D}^{(j)}_{train}$ as one *meta-instance* within the configuration process. We note that many configurators, such as *FocusedILS* [Hutter *et al.*, 2009], *irace* [López-Ibáñez *et al.*, 2016] and *SMAC* [Hutter *et al.*, 2011], can discard configurations when they perform poorly on a subset of meta-instances and therefore do not have to evaluate all k cross-validation folds for every configuration. This saves time and lets us evaluate more configurations within the same configuration budget. Based on these considerations, Figure 4 outlines the process to configure an algorithm selector with AUTOFOLIO.

3.2 Configuration Space of Selectors

Most existing algorithm selectors implement one specific AS approach, using one specific machine learning technique. However, most selection approaches, at least implicitly, admit more flexibility, and in particular could be used with a range of machine learning techniques. Based on this observation, we consider a hierarchically structured configuration space with a top-level parameter that determines the overall AS approach. For most selection approaches, we can then choose between different regression techniques. Each of these machine learning techniques can be configured by its own (hyper-)parameters. Additionally, further techniques can be used for preprocessing the training data and for presolving schedules, which can be configured independently from the selection approach, and are therefore also handled by top-level parameters.

We implemented these choices in the CLASPFOLIO 2 system. Figure 5 illustrates the complete configuration space thus obtained. Our current version, which we use for the concrete implementation of our AUTOFOLIO approach, covers six different AS approaches:

(hierarchical) regression (inspired by SATZILLA'09; [Xu *et al.*, 2008]) learns a regression model for each algorithm; for a new instance, it then selects the algorithm with best predicted performance;

multiclass classification (inspired by LLAMA; [Kotthoff, 2013]) learns a classification model that directly selects an algorithm based on the features of a new instance;

pairwise classification (inspired by SATZILLA'11; [Xu *et al.*, 2011]) learns a (cost-sensitive) classification model for all pairs of algorithms; for a new instance, it evaluates all models and selects the algorithm with the most votes;

clustering (inspired by ISAC; [Kadioglu *et al.*, 2010]) determines subsets of similar training instances in the feature space and the best algorithm on these subsets; for a new instance, it determines the nearest cluster center and selects the associated algorithm;

k-NN (inspired by 3S; [Kadioglu *et al.*, 2011], and MASP; [Maratea *et al.*, 2014]) determines a set of similar training instances in the feature space for a given new instance and selects the algorithm with the best performance on this instance set;

SNNAP (inspired by [Collautti *et al.*, 2013]) predicts the performance of each algorithm with regression models and uses this information for a k-NN approach in the predicted performance space.

For each of these approaches, CLASPFOLIO 2 covers at least three different machine learning techniques (where appropriate). For preprocessing strategies, it supports several feature and performance preprocessing strategies.

The current version of AUTOFOLIO employs the algorithm configurator *SMAC* [Hutter *et al.*, 2011] to search for the best AS approach in this combined space. We chose *SMAC*, because it performed best across the algorithm configuration problems we studied so far.

4 Empirical Performance Analysis

We studied the performance of AUTOFOLIO across a wide variety of hard combinatorial problems from the algorithm selection library (ASlib [Bischl *et al.*, 2016]).[1] Figure 1 shows the performance of AUTOFOLIO (last column) compared to its components with fixed parameter settings. AUTOFOLIO found well-performing configurations of CLASPFOLIO 2 and performed better than all other approaches on 11 out of the

[1] www.aslib.net

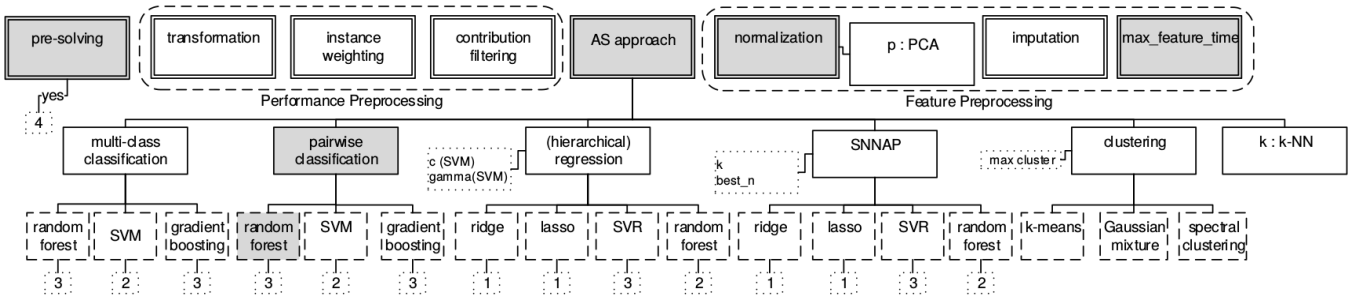

Figure 5: Configuration space of CLASPFOLIO 2. Parameters in double boxes are top-level parameters; single boxes represent AS approaches based on classes of machine learning techniques, dashed boxes machine learning techniques, and dotted boxes the number of low-level parameters. Parameter boxes used in the default configuration are filled in grey.

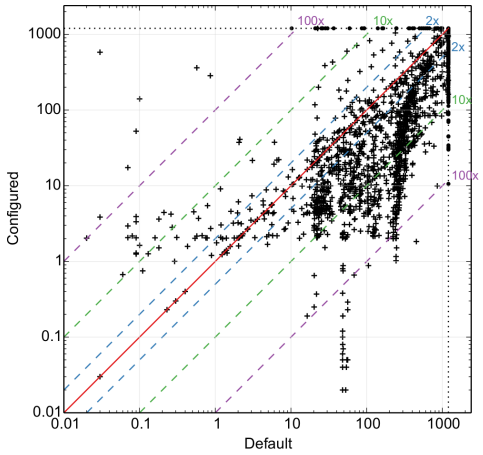

Figure 6: Scatter plots comparing the per-instance performance of default CLASPFOLIO 2 (SATZILLA'11-like) and AUTOFOLIO on SAT12-ALL. AUTOFOLIO improved performance on most instances and also reduced the number of timeouts.

13 ASlib scenarios. Figure 6 shows the performance difference between the default of CLASPFOLIO 2 and AUTOFOLIO on an per-instance basis on the ASlib scenario SAT12-ALL indicating much better performance of AUTOFOLIO.

Additionally, in our journal article [Lindauer *et al.*, 2015b], we investigated several aspects of AUTOFOLIO's behaviour in more detail:

1. We characterized the performance of AUTOFOLIO with different configuration spaces, showing advantages of a small space if only few performance evaluations are feasible in the configuration budget;
2. We demonstrated that AUTOFOLIO performs more robustly than other state-of-the-art selectors, establishing new state-of-the-art performance on 7 ASlib scenarios;
3. We applied parameter importance analysis [Hutter *et al.*, 2014; Fawcett and Hoos, 2016] to identify that the following choices were crucial to obtain strong performance for a given ASlib scenario: feature computation time, the set of instance features and the AS approach.

5 Conclusions and Future Work

In this extended abstract of our JAIR article [Lindauer *et al.*, 2015b], we summarized AUTOFOLIO—to the best of our knowledge, the first approach to automatically configure algorithm selectors. Using a concrete realization of this approach based on the highly parameterized AS framework CLASPFOLIO 2, we showed that by using state-of-the-art algorithm configurators, algorithm selectors can be customized to robustly achieve peak performance across a range of AS scenarios. The resulting approach performs significantly (and sometimes substantially) better than manually configured selectors and can be applied out-of-the-box to previously unseen AS scenarios.

In future work, we plan to investigate how the potential gains of larger configuration spaces (including feature and algorithm subset selection) can be used more effectively. To this end, we would like to (i) extend AUTOFOLIO's configuration space by implementing more algorithm selection approaches (e.g., CSHC [Malitsky *et al.*, 2013]); (ii) shrink large configuration spaces based on the analysis of parameter importance through fANOVA [Hutter *et al.*, 2014] and ablation [Fawcett and Hoos, 2016], allowing the configurator to focus on the most important parameters; and (iii) automatically select between pre-configured algorithm selectors, based on features of a given AS scenario, and further improve performance by warmstarting automatic configuration from the configurations thus selected [Feurer *et al.*, 2015b]. Another promising avenue for reducing the computational cost of our approach would be to pre-select algorithms, features, and problem instances based on the techniques proposed by Hoos *et al.* (2013). Finally, we plan to investigate to which extent AUTOFOLIO can configure AS systems for selecting parallel portfolios [Lindauer *et al.*, 2015a] to exploit the increasing availability of parallel computing resources.

Overall, our results for AUTOFOLIO strongly suggest that the automated configuration of AS systems can improve the performance and versatility of those systems across a broad range of application domains. Our AUTOFOLIO approach also facilitates future improvements, by making it easier to realize and assess the performance potential inherent in new design choices for the various components of an AS system. Our open-source implementation of AUTOFOLIO is available at `www.ml4aad.org/autofolio/`.

References

[Bergstra *et al.*, 2011] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Proc. of NIPS'11*, pages 2546–2554, 2011.

[Bischl *et al.*, 2016] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Frechétte, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren. ASlib: A benchmark library for algorithm selection. *AIJ*, pages 41–58, 2016.

[Collautti *et al.*, 2013] M. Collautti, Y. Malitsky, D. Mehta, and B. O'Sullivan. SNNAP: Solver-based nearest neighbor for algorithm portfolios. In *Proc. of ECML/PKDD'13*, pages 435–450, 2013.

[Fawcett and Hoos, 2016] C. Fawcett and H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22(4):431–458, 2016.

[Feurer *et al.*, 2015a] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *Proc. of NIPS'15*, 2015.

[Feurer *et al.*, 2015b] M. Feurer, T. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Proc. of AAAI'15*, pages 1128–1135, 2015.

[Gomes and Selman, 2001] C. Gomes and B. Selman. Algorithm portfolios. *AIJ*, 126(1-2):43–62, 2001.

[Hoos *et al.*, 2013] H. Hoos, B. Kaufmann, T. Schaub, and M. Schneider. Robust benchmark set selection for boolean constraint solvers. In *Proc. of LION'13*, pages 138–152, 2013.

[Hoos *et al.*, 2014] H. Hoos, M. Lindauer, and T. Schaub. claspfolio 2: Advances in algorithm selection for answer set programming. *TPLP*, 14:569–585, 2014.

[Hoos *et al.*, 2015] H. Hoos, R. Kaminski, M. Lindauer, and T. Schaub. aspeed: Solver scheduling via answer set programming. *TPLP*, 15:117–142, 2015.

[Huberman *et al.*, 1997] B. Huberman, R. Lukose, and T. Hogg. An economic approach to hard computational problems. *Science*, 275:51–54, 1997.

[Hutter *et al.*, 2009] F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. ParamILS: An automatic algorithm configuration framework. *JAIR*, 36:267–306, 2009.

[Hutter *et al.*, 2011] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION'11*, pages 507–523, 2011.

[Hutter *et al.*, 2014] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proc. of ICML'14*, pages 754–762, 2014.

[Kadioglu *et al.*, 2010] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC - instance-specific algorithm configuration. In *Proc. of ECAI'10*, pages 751–756, 2010.

[Kadioglu *et al.*, 2011] S. Kadioglu, Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann. Algorithm selection and scheduling. In *Proc. of CP'11*, pages 454–469, 2011.

[Kotthoff, 2013] L. Kotthoff. LLAMA: leveraging learning to automatically manage algorithms. *CoRR*, abs/1306.1031, 2013.

[Kotthoff, 2014] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, pages 48–60, 2014.

[Lindauer *et al.*, 2015a] M. Lindauer, H. Hoos, and F. Hutter. From sequential algorithm selection to parallel portfolio selection. In *Proc. of LION'15*, pages 1–16, 2015.

[Lindauer *et al.*, 2015b] M. Lindauer, H. Hoos, F. Hutter, and T. Schaub. Autofolio: An automatically configured algorithm selector. *JAIR*, 53:745–778, August 2015.

[López-Ibáñez *et al.*, 2016] M. López-Ibáñez, J. Dubois-Lacoste, L. Perez Caceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[Malitsky *et al.*, 2013] Y. Malitsky, A. Sabharwal, H. Samulowitz, and M. Sellmann. Algorithm portfolios based on cost-sensitive hierarchical clustering. In F. Rossi, editor, *Proc. of IJCAI'13*, pages 608–614, 2013.

[Maratea *et al.*, 2014] M. Maratea, L. Pulina, and F. Ricca. A multi-engine approach to answer-set programming. *TPLP*, 14:841–868, 2014.

[Nudelman *et al.*, 2004] E. Nudelman, K. Leyton-Brown, A. Devkar, Y. Shoham, and H. Hoos. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Proc. of CP'04*, pages 438–452, 2004.

[Rice, 1976] J. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[Smith-Miles, 2008] K. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM*, 41(1), 2008.

[Snoek *et al.*, 2012] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proc. of NIPS'12*, pages 2960–2968, 2012.

[Thornton *et al.*, 2013] C. Thornton, F. Hutter, H. Hoos, and K. Leyton-Brown. Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In *Proc. of KDD'13*, pages 847–855, 2013.

[Xu *et al.*, 2008] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *JAIR*, 32:565–606, 2008.

[Xu *et al.*, 2011] L. Xu, F. Hutter, H. Hoos, and K. Leyton-Brown. Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming. In *RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion at the International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.