

# Synergizing Theory and Practice of Automated Algorithm Design for Optimization

Diederick Vermetten<sup>\*1</sup>, Martin S. Krejca<sup>†2</sup>, Marius Lindauer<sup>†3</sup>,  
Manuel López-Ibáñez<sup>†4</sup>, and Katherine M. Malan<sup>†5</sup>

1 Leiden University, NL. [d.l.vermetten@liacs.leidenuniv.nl](mailto:d.l.vermetten@liacs.leidenuniv.nl)

2 LIX, Ecole Polytechnique, IP Paris, Palaiseau, FR.  
[martin.krejca@polytechnique.edu](mailto:martin.krejca@polytechnique.edu)

3 Institute of AI, Leibniz University Hannover, DE.  
[m.lindauer@ai.uni-hannover.de](mailto:m.lindauer@ai.uni-hannover.de)

4 Alliance Manchester Business School, University of Manchester, UK.  
[manuel.lopez-ibanez@manchester.ac.uk](mailto:manuel.lopez-ibanez@manchester.ac.uk)

5 Department of Decision Sciences, University of South Africa – Pretoria, ZA.  
[malankm@unisa.ac.za](mailto:malankm@unisa.ac.za)

---

## Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 23332, which focused on automated algorithm design (AAD) for optimization. AAD aims to propose good algorithms and/or parameters thereof for optimization problems in an automated fashion, instead of forcing this decision on the user. As such, AAD is applicable in a variety of domains. The seminar brought together a diverse, international set of researchers from AAD and closely related fields. Especially, we invited people from both the empirical and the theoretical domain. A main goal of the seminar was to enable vivid discussions between these two groups in order to synergize the knowledge from either domain, thus advancing the area of AAD as a whole, and to reduce the gap between theory and practice. Over the course of the seminar, a good mix of breakout sessions and talks took place, which were very well received and which we detail in this report. Efforts to synergize theory and practice bore some fruit, and other important aspects of AAD were highlighted and discussed. Overall, the seminar was a huge success.

**Seminar** August 13–18, 2023 – <https://www.dagstuhl.de/23332>

**2012 ACM Subject Classification** Theory of computation → Mathematical optimization; Computing methodologies → Machine learning; Theory of computation → Design and analysis of algorithms

**Keywords and phrases** automated algorithm design, hyper-parameter tuning, parameter control, heuristic optimization, black-box optimization

**Digital Object Identifier** 10.4230/DagRep.13.8.46

---

\* Editorial Assistant / Collector

† Editor / Organizer



Except where otherwise noted, content of this report is licensed under a Creative Commons BY 4.0 International license

Synergizing Theory and Practice of Automated Algorithm Design for Optimization, *Dagstuhl Reports*, Vol. 13, Issue 8, pp. 46–70

Editors: Martin S. Krejca, Marius Lindauer, Manuel López-Ibáñez, and Katherine M. Malan



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Executive Summary

*Martin S. Krejca (LIX, Ecole Polytechnique, IP Paris, FR, martin.krejca@polytechnique.edu)*

*Marius Lindauer (Institute of AI, Leibniz University Hannover, DE, m.lindauer@ai.uni-hannover.de)*

*Manuel López-Ibáñez (Alliance Manchester Business School, University of Manchester, UK, manuel.lopez-ibanez@manchester.ac.uk)*

*Katherine M. Malan (Department of Decision Sciences, University of South Africa – Pretoria, ZA, malankm@unisa.ac.za)*

**License** © Creative Commons BY 4.0 International license

© Martin S. Krejca, Marius Lindauer, Manuel López-Ibáñez, and Katherine M. Malan

Automated algorithm design (AAD) is a prevalent and highly important domain, as design-dependent algorithms are used in a plethora of very different areas. Determining which algorithms and/or their parameters to choose for each given problem is a task that is infeasible to solve in this magnitude solely by humans. Computers can greatly assist in this process in a multitude of ways. Consequently, many different branches of AAD exist. This Dagstuhl Seminar brought together a diverse set of researchers from the AAD domain as well as closely related domains, from all over the world. The aim of the seminar was to provide a common platform to exchange ideas about all of the different established AAD techniques and theories, how to advance the field of AAD, and how to join forces. Most prominently, the seminar aimed at a strong exchange between researchers who work empirically and those who work theoretically. We believe that this seminar was a great success, reaching all of the intended goals.

Due to the diverse backgrounds of the participants, the seminar started with elaborate introduction rounds followed by two overview talks – one detailing the theoretical perspective of AAD, given by Carola Doerr, the other one the empirical perspective, given by Katherine M. Malan. From then on, the seminar consisted of a mix of breakout sessions and short, inspiring talks as participants felt the need to share their experiences or ideas based on the discussions. Immediately from day one, interesting breakout sessions emerged, some of which had multiple iterations, due to the interest of the participants and due to the scope of the subject. In the morning of each day, we made sure to provide a summary of all breakout sessions of the previous day in order to update everyone and to discuss what other topics could be further explored.

Topics that came up on day one were discussions about the terminology and about benchmarks, which had multiple sessions throughout the week. Both of these topics are very important when aiming to unite the community and especially when trying to bridge the gap between theory and practice. The sessions about terminology showed that different communities use different terms for similar concepts and, perhaps more worrying, terminology is not used consistently across communities, making communicating results between different areas harder. Potential solutions include to provide a freely accessible overview of the different terms and define them well, or to use whatever terminology feels appropriate for each article but specify the terms very well in the article. The sessions about benchmarks highlighted that more diverse sets of benchmark functions are desired and that it is not straightforward to transfer the existing benchmarks into a setting that is well suited for theoretical investigations. The latter point sparked other interesting discussions, summarized in this report.

An inspiring talk was given by Kevin Leyton-Brown, who proposed that optimization should be guided based on users' utilities rather than on expected runtime of the algorithms. This led to an interesting breakout group in which various useful utility functions were discussed which could be used for realistic AAD scenarios. Another very interesting breakout group was inspired by Benjamin Doerr, who was looking for the easiest possible AAD scenario to study theoretically. In a fruitful exchange, a scenario was constructed that seemed interesting and approachable from a theoretical point and also useful from a practical point of view. The discussion emerging from this group delved into an open question that was discussed throughout the seminar, namely, what are the characteristics that make AAD scenarios a special case of the more general field of black-box optimization.

Besides these success stories, further talks led to potential collaborations and new ideas. One example was the talk by Johannes Lengler, in which he proposed how existing theoretical results for randomized optimization heuristics could be potentially interpreted as results for AAD. The ensuing discussion was about to what extent such generalizations make sense, with no clear consensus being reached so far. Nonetheless, the talk and the following in-depth discussion showed that there is significant work to be done on this topic.

The seminar also gave members who are not part of the core AAD community the opportunity to present their work and to integrate into the community. This was met with very positive feedback. Overall, we thank all of the participants for the great discussions, valuable talks, and the support, all of which made this seminar a great success. In addition, we thank the Dagstuhl staff, who supported us incredibly well and helped us run this seminar as smoothly as it did.

## 2 Table of Contents

### Executive Summary

*Martin S. Krejca, Marius Lindauer, Manuel López-Ibáñez, and Katherine M. Malan* 47

### Overview of Talks

Reflections on Synergizing Theory and Practice of Automated Algorithm Design for Optimization <i>Carola Doerr</i> . . . . .	51
Runtime Analysis for the NSGA-II <i>Benjamin Doerr</i> . . . . .	51
A graphical overview of applied research in automated algorithm design <i>Katherine Malan</i> . . . . .	52
Modular Optimization Algorithms: Testing AAD for Continuous Optimization <i>Diederick Vermetten</i> . . . . .	53
A OneMax for Automated Algorithm Design? <i>Johannes Lengler</i> . . . . .	53
Success Story: Power of Hyper-Heuristics and Transfer Learning in AAD <i>Nelishia Pillay</i> . . . . .	53
Bimodal Parameter Landscapes in EDAs, Phase Transitions, and Stability <i>Carsten Witt</i> . . . . .	54
A few open questions related to automated algorithm selection <i>Mario Andrés Muñoz</i> . . . . .	54
Searching Problem Spaces: Automated Problem Design? <i>Marcus Gallagher</i> . . . . .	55
Automated algorithm configuration/selection and constraint programming <i>Nguyen Dang</i> . . . . .	55
A Comparative study of NCO and heuristics on TSP <i>Shengcai Liu</i> . . . . .	56
A Utilitarian Foundation for Algorithm Configuration <i>Kevin Leyton-Brown</i> . . . . .	56
From Synthetic to Real-World to Application Benchmarks <i>Marcel Wever</i> . . . . .	57
On the Probabilistic Model-Based Evolutionary Algorithm for Mixed-Variables <i>Shinichi Shirakawa</i> . . . . .	57
Landscape Features in AAS/C <i>Niki van Stein</i> . . . . .	58
Automated Algorithm Design from the Bottom Up <i>Lars Kothoff</i> . . . . .	58
UNSAT Solver Synthesis via Monte Carlo Forest Search <i>Chris Cameron</i> . . . . .	58


**Working Groups**

Working Group 1: Terminologies of Automated Algorithm Design	
<i>Nelishia Pillay</i> . . . . .	59
Working Group 2: Automated Algorithm Design Benchmarks	
<i>Johannes Lengler</i> . . . . .	60
Working Group 3: New Methods for Automated Algorithm Configuration	
<i>Marius Lindauer</i> . . . . .	61
Working Group 4: Utility Functions	
<i>Kevin Leyton-Brown</i> . . . . .	63
Working Group 5: New Estimation-of-Distribution Algorithms	
<i>Manuel López-Ibáñez</i> . . . . .	63
Working Group 6: The OneMax Problem of AAD	
<i>Benjamin Doerr</i> . . . . .	65
Working Group 7: Explainability of AAD Methods	
<i>Pascal Kerschke</i> . . . . .	66
Working Group 8: Automated Machine Learning	
<i>Marius Lindauer and Bernd Bischl</i> . . . . .	67
<b>Participants</b> . . . . .	70

### 3 Overview of Talks

#### 3.1 Reflections on Synergizing Theory and Practice of Automated Algorithm Design for Optimization

Carola Doerr (Sorbonne Université, CNRS, LIP6 – Paris, FR, [carola.doerr@lip6.fr](mailto:carola.doerr@lip6.fr))

License  Creative Commons BY 4.0 International license  
© Carola Doerr

This seminar is aimed at bringing together theoreticians in (black-box) optimization with researchers developing and using automated algorithm design techniques. In order to create a joint understanding for the topics of this seminar, I will briefly present some reflections on what I consider useful to know to get to know each other. In particular, I will emphasize that some participants may have a strong interest in optimizing automated algorithm design techniques, while others may have a stronger interest in using automated algorithm design techniques to identify efficient optimization algorithms.

I will also present some ideas for possible breakout sessions, including using benchmarks with proven theoretical guarantees to understand behavior and performance limits of state-of-the-art automated algorithm design techniques, the selection of instances to use during the design (training) process, proxies (e.g., for performance measures) that can be used in the training process to make it more efficient, and automated algorithm configuration techniques that output an explicit instance feature – parameter map.

#### 3.2 Runtime Analysis for the NSGA-II

Benjamin Doerr (Ecole Polytechnique, IP Paris, FR, [lastname@lix.polytechnique.fr](mailto:lastname@lix.polytechnique.fr))

License  Creative Commons BY 4.0 International license  
© Benjamin Doerr

**Joint work of** Weijie Zheng, Yufei Liu, Benjamin Doerr

**Main reference** Weijie Zheng, Yufei Liu, Benjamin Doerr: “A First Mathematical Runtime Analysis of the Non-dominated Sorting Genetic Algorithm II (NSGA-II)”, in Proc. of the Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 – March 1, 2022, pp. 10408–10416, AAAI Press, 2022.

**URL** <http://dx.doi.org/10.1609/AAAI.V36I9.21283>

Recently, the mathematical runtime analysis of evolutionary algorithms made a huge step forward by conducting several analyses of the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [1], the most prominent multi-objective evolutionary algorithm intensively used in practice (the work [1] is cited more than 50,000 times). In this talk, I touch upon three results in this research direction.


- The first such runtime analysis [2] proved that the NSGA-II easily computes the Pareto front of the bi-objective OneMinMax and LOTZ benchmarks when the population size is at least four time the size of the Pareto front. In contrast, when the population size is only equal to the size of the Pareto front, regularly desirable solutions are lost and consequently, it takes at least exponential time to reach a population that does not miss a constant fraction of the Pareto front.
- In sharp contrast to these and several other positive results for bi-objective problems [3, 4, 5, 6, 7, 8, 9, 10], in three or more objectives the NSGA-II cannot even optimize the simple OneMinMax problem [11].
- On the positive side, as the first runtime analysis for this algorithm shows, the NSGA-III [12] can efficiently optimize the 3-objective OneMinMax problem [13].

## References

- 1 Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.
- 2 Weijie Zheng, Yufei Liu, and Benjamin Doerr. A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In *Proc. of AAAI '22*, pages 10408–10416, 2022.
- 3 Chao Bian and Chao Qian. Better running time of the non-dominated sorting genetic algorithm II (NSGA-II) by using stochastic tournament selection. In *Proc. of PPSN '22*, pages 428–441, 2022.
- 4 Sacha Cerf, Benjamin Doerr, Benjamin Hebras, Jakob Kahane, and Simon Wietheger. The first proven performance guarantees for the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) on a combinatorial optimization problem. In *Proc. of IJCAI '23*, pages 5522–5530, 2023.
- 5 Duc-Cuong Dang, Andre Opris, Bahare Salehi, and Dirk Sudholt. Analysing the robustness of NSGA-II under noise. In *Proc. of GECCO '23*, pages 642–651, 2023.
- 6 Duc-Cuong Dang, Andre Opris, Bahare Salehi, and Dirk Sudholt. A proof that using crossover can guarantee exponential speed-ups in evolutionary multi-objective optimisation. In *Proc. of AAAI '23*, pages 12390–12398, 2023.
- 7 Benjamin Doerr and Zhongdi Qu. A first runtime analysis of the NSGA-II on a multimodal problem. *IEEE Transactions on Evolutionary Computation*, 27(5), 2023.
- 8 Benjamin Doerr and Zhongdi Qu. From understanding the population dynamics of the NSGA-II to the first proven lower bounds. In *Proc. of AAAI '23*, pages 12408–12416, 2023.
- 9 Benjamin Doerr and Zhongdi Qu. Runtime analysis for the NSGA-II: provable speed-ups from crossover. In *Proc. of AAAI '23*, pages 12399–12407, 2023.
- 10 Weijie Zheng and Benjamin Doerr. Better approximation guarantees for the NSGA-II by using the current crowding distance. In *Proc. of GECCO '22*, pages 611–619, 2022.
- 11 Weijie Zheng and Benjamin Doerr. Runtime analysis for the NSGA-II: proving, quantifying, and explaining the inefficiency for three or more objectives. *CoRR*, abs/2211.13084, 2022.
- 12 Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. 18:577–601, 2014.
- 13 Simon Wietheger and Benjamin Doerr. A mathematical runtime analysis of the Non-dominated Sorting Genetic Algorithm III (NSGA-III). In *Proc. of IJCAI '23*, pages 5657–5665, 2023.

## 3.3 A graphical overview of applied research in automated algorithm design

Katherine Malan (University of South Africa – Pretoria, ZA, [malankm@unisa.ac.za](mailto:malankm@unisa.ac.za))

License  Creative Commons BY 4.0 International license  
© Katherine Malan

With the aim of building a common understanding among the participants of seminar 23332, I have attempted to unpack the components of automated algorithm design (AAD) to visually illustrate the goals of different scenarios in AAD. After describing the different components (problem space, algorithm space, problem feature space, algorithm feature space, and performance space) and their specialized types, I describe some recent studies in AAD visually overlayed onto the framework. Studies from the three scenarios of automated

algorithm configuration, composition and selection are covered. The purpose of the talk is to generate discussion with the hope that this leads to a common understanding of some of the aspects of AAD.

### 3.4 Modular Optimization Algorithms: Testing AAD for Continuous Optimization

*Diederick Vermetten (Leiden University – NL, d.l.vermetten@liacs.leidenuniv.nl)*

License © Creative Commons BY 4.0 International license  
© Diederick Vermetten

I present a use case of AAD in the domain of continuous optimization where we found that modularizing popular algorithms and applying configuration techniques can yield clear improvements in anytime performance. We do however have problems when testing generalizability, as our test suites have not been designed for this purpose. We thus propose a new function generator: MA-BBOB, which we hope can provide a playground for testing the generalizability of AAD methods in optimization.

### 3.5 A OneMax for Automated Algorithm Design?

*Johannes Lengler (ETH Zürich, CH, johannes.lengler@inf.ethz.ch)*

License © Creative Commons BY 4.0 International license  
© Johannes Lengler

In this talk I present one candidate for a “OneMax” function in automated algorithm composition, i.e., a simple benchmark that we would expect an automated algorithm composer to solve.

The suggestion is that the potential algorithm consists of  $n$  components. For each of them there are two components, type 0 and type 1. When the composer wants to compare two or more algorithms, it draws a random permutation of size  $n$ , and gives credit  $2^i$  to the  $i$ -th component of the algorithm if this is of type 1, and credit 0 if it is of type 0. The total fitness performance of an algorithm is then the sum of the credits of all its components.

I discussed connections of this benchmark to the class of monotone functions (like HotTopic functions) and of dynamic BinVal functions that have been used as benchmarks in the context of randomized optimization heuristics.

### 3.6 Success Story: Power of Hyper-Heuristics and Transfer Learning in AAD

*Nelishia Pillay (University of Pretoria, ZA, nelishia.pillay@up.ac.za)*

License © Creative Commons BY 4.0 International license  
© Nelishia Pillay


Automated design (AutoDes) encompasses automated configuration, automated composition, automated generation and automated selection. We focus on automated configuration (AutoCon), automated composition (AutoCom) and automated generation (AutoGen) for



finding scalable solutions to complex problems in agriculture, health, education and industry. We investigate hyper-heuristics and transfer learning in these three AutoDes areas. We measure success in terms of the human competitiveness, i.e., do the solutions outperform human solutions, as well as in terms of providing improvements over existing techniques used for solving these problems. Hyper-heuristics work in an alternate space that maps to the design space and as such allows for areas that could not be reached by searching the design space directly to be explored. Transfer learning involves transferring knowledge between design spaces. We have found that using hyper-heuristics and transfer learning in AutoCom, AutoCon and AutoGen provide human competitive results with reduced computational cost.

### 3.7 Bimodal Parameter Landscapes in EDAs, Phase Transitions, and Stability

*Carsten Witt (Technical University of Denmark, Kgs. Lyngby, DK, cawi@dtu.dk)*

License  Creative Commons BY 4.0 International license  
© Carsten Witt

We consider the simple estimation-of-distribution algorithm cGA on the OneMax benchmark function. Previous research by Lengler, Sudholt and Witt (Algorithmica 2021) shows that its expected runtime depends in a non-trivial way on its update parameter  $K$ , resulting in a bimodal landscape and two widely different settings minimizing the expected runtime. Moreover, we discuss phase transitions in algorithm behavior related to the choice of  $K$ , with chaotic behavior of marginal probabilities and high genetic drift below the phase transition, and very stable behavior above it.

### 3.8 A few open questions related to automated algorithm selection

*Mario Andrés Muñoz (OPTIMA, The University of Melbourne, AU, munoz.m@unimelb.edu.au)*

License  Creative Commons BY 4.0 International license  
© Mario Andrés Muñoz

From over ten years of work on algorithm selection for continuous black-box optimization, I have more open questions than answers. In this talk, I discussed some of my most recent projects and, what they are to me, important challenges worth tackling. First, it is the understanding of the relationships between real-world problems and benchmarks, by being able to place them in a common instance space. Related, is the need to have more informative descriptions of problems that are interpretable to humans and not only machine learning algorithms. Second, we need to describe algorithm behaviors, and strategies to find suitable algorithms for hard problems, where selection seems insufficient. All of this must be supported by an infrastructure that facilitates sharing of results, data and software.

### 3.9 Searching Problem Spaces: Automated Problem Design?

Marcus Gallagher (*The University of Queensland – Brisbane, AU, marcusg@uq.edu.au*)

License  Creative Commons BY 4.0 International license  
© Marcus Gallagher


Problem instances are a key component of automated algorithm design in optimization, particularly when we are concerned with benchmarking and comparing the performance of different techniques. Problem instances may be produced via manual design, randomized generators, feature space or derived from real-world domains. Alternatively, we can produce problem instances via some kind of search over some appropriate problem space. The key challenges lie in how to search this space effectively. Some existing work in this space includes:

- Using racing algorithms across problem instances.
- Creating problem transformations (e.g. sphere to Rastrigin function), implicitly defining a path through the problem space.
- Search the (hyper)parameter space of a problem instance generator.
- Search the (hyper)parameter space of a surrogate model.
- Perturbation of a dataset (when the objective function is data driven).
- Use of some type of generative model.

There seems to be much room for further research in this direction. Seminar attendees began a collaborative document to collate references in this area.

### 3.10 Automated algorithm configuration/selection and constraint programming

Nguyen Dang (*University of St Andrews, UK, nttd@st-andrews.ac.uk*)

License  Creative Commons BY 4.0 International license  
© Nguyen Dang

Constraint programming (CP) can offer an accessible means to non-expert users to solve their combinatorial problems. More concretely, constraint modeling and solving pipelines such as MiniZinc and Essence provide an expressive modeling language to describe a problem and the relevant data (problem instance) and a toolchain to translate/reformulate the problem (and instance) into the input understandable by a generic (CP/SAT/SMT/MIP) solver. In this talk, I will talk about two applications we have done in our CP group in St Andrews which link automated algorithm configuration/selection (AC/AS) and CP. The first one is an application of AC/AS on automated generation and selection of streamliner constraints, which was shown to significantly speed up the solving of constraint/SAT solvers. The second one is about combining AC/AS and CP to create an automated instance generation system for benchmarking purposes.

### 3.11 A Comparative study of NCO and heuristics on TSP

*Shengcai Liu (Agency for Science, Technology and Research, Singapore, SG, liu\_shengcai@cfar.a-star.edu.sg)*

**License** © Creative Commons BY 4.0 International license

© Shengcai Liu

**Joint work of** Shengcai Liu, Yu Zhang, Ke Tang, Xin Yao

**Main reference** Shengcai Liu, Yu Zhang, Ke Tang, Xin Yao: “How Good is Neural Combinatorial Optimization? A Systematic Evaluation on the Traveling Salesman Problem”, IEEE Computational Intelligence Magazine, Vol. 18(3), pp. 14–28, 2023.

**URL** <http://dx.doi.org/10.1109/MCI.2023.3277768>

Neural Combinatorial Optimization (NCO) is a recent research area in deep learning that uses deep reinforcement learning to train DNNs (policies) to solve combinatorial optimization problems (e.g., TSPs, VRPs). However, the experiments presented in existing works are generally non-conclusive due to several reasons: 1) The state-of-the-art heuristic solvers are usually missing, e.g., EAX for TSPs, HGS for VRPs; 2) For heuristic solvers, their default parameter values are used which neglects the fact of parameter tuning; 3) Unfair comparison of parallel NCO solvers vs. sequential heuristic solvers; 4) The benchmark instances are quite limited in terms of both types and sizes.

In this talk, I introduce our recent comparative study of NCO solvers and heuristic solvers on the TSPs. The study is the first that 1) considers five different problem types, 2) involves problem instances with up to 10000 nodes, 3) includes tuned traditional solvers in the comparison, and 4) investigates five different performance aspects including the energy consumption of the solvers. Four main conclusions are drawn from the experiments: 1) For all the TSP problem sizes and types considered in the experiments, heuristic solvers still significantly outperform NCO solvers in nearly all the five performance aspects; 2) A potential benefit of NCO solvers would be their superior time and energy efficiency for small-size problem instances when sufficient training instances are available; 3) current NCO approaches are not suitable for handling large-size problem instances, structural problem instances, and mixed problem instances; 4) When the training instances have different problem characteristics (problem types and sizes) from those of the testing instances, both NCO solvers and tuned traditional solvers exhibit performance degradation, and NCO solvers suffered from far more severe performance degradation

### 3.12 A Utilitarian Foundation for Algorithm Configuration

*Kevin Leyton-Brown (University of British Columbia – Vancouver, CA, kevinlb@cs.ubc.ca)*

**License** © Creative Commons BY 4.0 International license

© Kevin Leyton-Brown

**Joint work of** Devon R. Graham, Kevin Leyton-Brown, Tim Roughgarden

**Main reference** Devon R. Graham, Kevin Leyton-Brown, Tim Roughgarden: “Formalizing Preferences Over Runtime Distributions”, in Proc. of the 40th International Conference on Machine Learning, Proceedings of Machine Learning Research, Vol. 202, pp. 11659–11682, PMLR, 2023.

**URL** <https://proceedings.mlr.press/v202/graham23a.html>

When trying to solve a computational problem, we are often faced with a choice between algorithms that are guaranteed to return the right answer but differ in their runtime distributions (e.g., SAT solvers, sorting algorithms). This work aims to lay theoretical foundations for such choices by formalizing preferences over runtime distributions. It might seem that we should simply prefer the algorithm that minimizes expected runtime. However, such preferences would be driven by exactly how slow our algorithm is on bad inputs, whereas

in practice we are typically willing to cut off occasional, sufficiently long runs before they finish. We propose a principled alternative, taking a utility-theoretic approach to characterize the scoring functions that describe preferences over algorithms. These functions depend on the way our value for solving our problem decreases with time and on the distribution from which captimes are drawn. We describe examples of realistic utility functions and show how to leverage a maximum-entropy approach for modeling underspecified captime distributions. Finally, we show how to efficiently estimate an algorithm's expected utility from runtime samples.

### 3.13 From Synthetic to Real-World to Application Benchmarks

*Marcel Wever (LMU Munich, DE, marcel.wever@ifi.lmu.de)*

License © Creative Commons BY 4.0 International license  
© Marcel Wever

Algorithm selectors and algorithm configurators are most commonly benchmarked on algorithms for computationally hard problems, e.g., SAT, TSP, or MIP. While such benchmarks facilitate unleashing the power of algorithm selection and configuration systems, results underlie limited interpretability. More specifically, if an algorithm configurator fails to yield a substantially better-performing configuration, it is unclear whether it is nonexistent or hard to find. From the viewpoint of a practitioner, however, it is not clear to what extent algorithm selection or configuration can be applied to a broader scope of software systems. In this presentation, I invite the community to create theoretically inspired, synthetic benchmarks that exhibit certain characteristics inherent to algorithm configuration problems on the one hand, and more applied benchmarks on the other hand.

### 3.14 On the Probabilistic Model-Based Evolutionary Algorithm for Mixed-Variables


*Shinichi Shirakawa (Yokohama National University, JP, shirakawa-shinichi-bg@ynu.ac.jp)*

License © Creative Commons BY 4.0 International license  
© Shinichi Shirakawa

In automated machine learning and automatic algorithm design, mixed types of design variables, including continuous, integer, and categorical variables, should be optimized. I briefly introduce the extension of CMA-ES, called CMA-ES with margin, to handle integer variables. CMA-ES with margin prevents stagnation of the search distribution to a certain integer value in mixed-integer problems by introducing lower-bounded marginal probabilities of integer variables. I also describe information geometric optimization (IGO) with categorical distribution, which is a probabilistic model-based evolutionary algorithm for categorical domains. CMA-ES with margin and IGO with categorical distribution update the parameters of Gaussian and categorical distributions, respectively, to search for better solutions. Then, I mention a possible extension of these algorithms for handling mixed-variable problems consisting of continuous, integer, and categorical variables.

### 3.15 Landscape Features in AAS/C


*Niki van Stein (Leiden University, NL, [n.v.stein@liacs.leidenuniv.nl](mailto:n.v.stein@liacs.leidenuniv.nl))*

License  Creative Commons BY 4.0 International license  
© Niki van Stein

We propose DoE2Vec, a variational autoencoder (VAE)-based methodology to learn optimization landscape characteristics for downstream meta-learning tasks, e.g., automated selection of optimization algorithms. Principally, using large training data sets generated with a random function generator, DoE2Vec self-learns an informative latent representation for any design of experiments (DoE). Unlike the classical exploratory landscape analysis (ELA) method, our approach does not require any feature engineering and is easily applicable for high dimensional search spaces. For validation, we inspect the quality of latent reconstructions and analyze the latent representations using different experiments. The latent representations not only show promising potentials in identifying similar (cheap-to-evaluate) surrogate functions, but also can significantly boost performances when being used complementary to the classical ELA features in classification tasks.

### 3.16 Automated Algorithm Design from the Bottom Up

*Lars Kothoff (University of Wyoming – Laramie, US, [larsko@uwyo.edu](mailto:larsko@uwyo.edu))*

License  Creative Commons BY 4.0 International license  
© Lars Kothoff

Recent advances in algorithm selection allow to automatically characterize algorithms based on their source code. This offers new directions for automated algorithm design, enabling researchers to quantify existing and generated code in novel ways and compare and contrast different sources. Most existing approaches for automated algorithm design are top-down, i.e. algorithms are designed or modified based on high-level descriptions. In this talk, I will give an overview of the new methods for characterizing source code and sketch future research directions that could leverage this, highlighting the results of a preliminary analysis of existing solvers for hard AI problems.

### 3.17 UNSAT Solver Synthesis via Monte Carlo Forest Search

*Chris Cameron (University of British Columbia – Vancouver, CA, [cchris13@cs.ubc.ca](mailto:cchris13@cs.ubc.ca) )*

License  Creative Commons BY 4.0 International license  
© Chris Cameron

We introduce Monte Carlo Forest Search (MCFS), a class of reinforcement learning (RL) algorithms for learning policies in tree MDPs, for which policy execution involves traversing an exponential-sized tree. Examples of such problems include proving unsatisfiability of a SAT formula; counting the number of solutions of a satisfiable SAT formula; and finding the optimal solution to a mixed-integer program. MCFS algorithms can be seen as extensions of Monte Carlo Tree Search (MCTS) to cases where, rather than finding a good path (solution) within a tree, the problem is to find a small tree within a forest of candidate trees. We instantiate and evaluate our ideas in an algorithm that we dub Knuth Synthesis, an

MCFS algorithm that learns DPLL branching policies for solving the Boolean satisfiability (SAT) problem, with the objective of achieving good average-case performance on a given distribution of unsatisfiable problem instances. Knuth Synthesis leverages two key ideas to avoid the prohibitive costs of policy evaluations in an exponentially-sized tree. First, we estimate tree size by randomly sampling paths and measuring their lengths, drawing on an unbiased approximation due to Knuth (1975). Second, we query a strong solver at a user-defined depth rather than learning a policy across the whole tree, to focus our policy search on early decisions that offer the greatest potential for reducing tree size.

## 4 Working Groups

### 4.1 Working Group 1: Terminologies of Automated Algorithm Design

*Nelishia Pillay (University of Pretoria, ZA, nelishia.pillay@up.ac.za)*

License © Creative Commons BY 4.0 International license  
© Nelishia Pillay

This focus of this working group was to find a standardization of terminology for AAD. In the literature there is no consistency in terms defining AAD concepts.

#### 4.1.1 Discussed Problems

The first session of the working group highlighted the following challenges that need to be addressed regarding the terminology used for AAD:

- Standardize terminology for AAD – different researchers use different terms for the same concepts, thus a standardization is needed.
- Remove confusion regarding AAD terms in the literature – due to different terms being used for the same concept this leads to confusion.
- The need of an extensible framework/standardization as the field develops further – AAD is a rapidly developing field and as such there is a need for an extensible standardization.
- Principle components of design need to be identified – a starting point of the standardization would be to identify the main/principle components of designs.

#### 4.1.2 Possible Approaches

Criteria Considered for Categorization

- Underlying design problem – what is the design problem being solved, e.g. determining hyperparameters, creating a new operator.
- Outcome – What is the design that is produced? Does it have to be reusable?
- High-level methodology – What optimization technique is used to generate the design?

Proposed Categorization

- Automated configuration – Involves determining the parameters and hyper-parameters, e.g. learning rate, population size, operator probabilities. Related terms: HPO, parameter tuning, parameter control.
- Automated Selection – Involves selecting algorithms. Same underlying problem in automated configuration.
- Automated Composition – Involves combining existing processes/algorithms as black box components.
- Automated Generation – Involves creating new algorithms, e.g. new move operators, new genetic operators.

**Textbox 2: Automatic Algorithm Configuration**

In the machine learning community, **automatic algorithm configuration** is a method for determining a well-performing parameter configuration  $\theta^* \in \Theta$  of a given algorithm  $A \in \mathcal{P}$  across a given set of instances  $i \in \Pi = \{i_1, \dots, i_l\}$ . Given a cost metric  $c: \mathcal{P} \times \Pi \rightarrow \mathbb{R}$  that computes a cost  $c(A(\theta), i)$  for a given algorithm and its configuration on an instance, a distribution  $\mathcal{C}(A(\theta), \Pi)$  of such cost values over instances drawn from  $\Pi$ , and a function  $m(\mathcal{C}(A(\theta), \Pi))$  that computes a statistical population parameter (e.g., mean, median, ...) of  $\mathcal{C}(A(\theta), \Pi)$ , the **algorithm configuration problem** is to find the optimal parameter configuration  $\theta^*$  for algorithm  $A$  for the objective  $m$ , over the instance set  $\Pi$ :

$$\theta^* = \arg \max_{\theta \in \Theta} m(\mathcal{C}(A(\theta), \Pi))$$

■ **Figure 1** Proposed Mathematical Formulation.

Proposed Principle Components:

- Problem input – What is the input to the design program, e.g. values for parameters, potential components of an operator?
- Search space – What space is being explored to solve the problem, e.g. parameter space, program space, heuristic space.
- Methods – The optimization approaches used for search.
- Objective – What is the purpose of solving the problem, e.g. reduced workload, better design leading to better results.
- Outcome – The design produced.

Areas for further discussion:

- Is automated selection and automated configuration the same problem?
- Is the search space a principle component?

### 4.1.3 Conclusions

The discussions held provided a starting point to develop a standardization for AAD. There are some areas for further discussion where consensus was not reached. The aim is for this standardization proposal generated from the discussions to be taken further by interested members of the community for further discussion and possible formalization.

## 4.2 Working Group 2: Automated Algorithm Design Benchmarks

Johannes Lengler (ETH Zurich, CH, [johannes.lengler@inf.ethz.ch](mailto:johannes.lengler@inf.ethz.ch))

License  Creative Commons BY 4.0 International license  
© Johannes Lengler

This working group focused on how benchmark suites for Automated Algorithm Design should look like.

### 4.2.1 Discussed Problems

- What are landscape features? Do theoretical benchmarks capture them?
- Useful benchmarks for algorithm configuration
- Benchmarks on AAD

- AAD across combinatorial problem domains
- How to bridge theory and practice? What about theoretical empirics?
- Benchmarking on real-world applications
- Representativeness of benchmark data
- Theory-inspired benchmarks for understanding AAD
- Improving experimental methods for AAS etc.
- How to select instances for the transfer from training to testing
- Understanding problem spaces and problem properties better

#### 4.2.2 Conclusions

- Benchmark sets in optimization and in AAD often tend to be small.
- Benchmarking is probably more instrumental in automated algorithm design than in “manual” algorithm design however we could not point to in which way benchmarking needs to be distinctively different.
- It is not clear whether we want to increase the size and diversity of the benchmark suites. In principle this is desirable (ML), but we do want our benchmark set to still reflect the real world. Data augmentation was successful in ML.
- Perhaps optimization heuristics even have a strength in being able to work with little training data (flexibility, adaptability).
- This touches on the question at which level the automation should take place. For one problem class? For everything? That decides the scope of the benchmark set.
- It is controversial how useful or arbitrary a feature space representation of benchmark problems is.
- Invariance properties may be a relevant and desirable aspect for features.

### 4.3 Working Group 3: New Methods for Automated Algorithm Configuration

*Marius Lindauer (Leibniz University Hannover, DE, m.lindauer@ai.uni-hannover.de)*

License © Creative Commons BY 4.0 International license  
© Marius Lindauer

This working group focused on current and new methods for automated algorithm configuration (AAC).

#### 4.3.1 Open and Understudied Problems

As a first discussion point, we discussed what is missing or not well-studied in AAC. This includes:

1. Multi-objective AAC guided by a surrogate (e.g., Bayesian optimization) s.t. users can optimize for complementary objectives and obtain an approximated Pareto front at the end [7];
2. AC methods with feature-dependent results (a.k.a per-instance algorithm configuration or instance-specific algorithm configuration) where the results are (I) interpretable (e.g., by providing symbolic formulas) or (II) derived from a grammar to be more expressive compared to common bounded configuration spaces;
3. Pre-trained surrogate models that are inspired the current trend of foundation models;



4. Text-based interfaces for AAC based on large language models (LLMs) to provide an easy user interface where domain experts can easily interact with [28];
5. Allowing more complicated constraints in the configuration space – currently this is dominated by rejection sampling that cannot deal with highly constrained spaces;
6. Interactive methods where one can aggregate several preferences across multiple interactions – the major difficulty lies in the aggregation;
7. Statistical guarantees s.t. users can trust that the results are close to optimal w.r.t. the evaluated configurations (e.g., the configuration returned is not worse than any other seen with 95% certainty);
8. How to deal with low signal-noise ratios where there is a lot (maybe even heteroscedastic) noise on algorithm evaluations?
9. How can theoretically-driven racing schemes (e.g. [30]) and user-defined utility-functions [13] be combined with Bayesian-optimization guided AAC?
10. How can we configure our AAC methods in view of how expensive it is to run AAC methods? Simply applying AAC to AAC will be too expensive in most cases or diminishing returns have to be taken into account [21];
11. Maybe surprisingly, the commonly used AAC packages (e.g., irace [22] and SMAC [18, 20]) do not implement all state-of-the-art ideas from other AI subfields (e.g., foundation models, constraint solving, LLMs and theoretical sound approaches).

#### 4.3.2 Promising Next Steps

At the end of the discussions, several promising directions were identified:

1. What are actually the most important components of an AAC method? A systematic study across different AAC methods and application domains is still missing. This could be done based on AClib [19]. This will also require updating AClib with new scenarios, algorithms and instances.
2. How can we make results from AAC methods more interpretable to (I) increase the users' trust into these methods and to (II) enable theory-driven results based on empirical results? The common approaches include either using more interpretable models (e.g., linear models or symbolic models) or model-agnostic post-hoc explanations.
3. How can we increase the robustness of AAC methods where robustness can refer to noisy algorithm evaluations, heterogeneous instance sets or stochasticity of the AAC method?

#### 4.3.3 Dedicated Discussion on Dynamic Algorithm Configuration

In contrast to the traditional approach of wrapping AAC around existing algorithms, an alternative is to tightly integrate AAC into the algorithms. This enables dynamic configuration (DAC) [1] while the algorithm is running and can, e.g., dynamically adapt to new challenges in the optimization landscape.

There are two major lines of work in this direction:

1. Configuring the parameters of the algorithm dynamically. This follows the idea of the traditional AAC methods that takes the algorithm as it is and only controls its parameters [27, 5]. With DACBench [10], there is a benchmark library that allows to benchmark such approaches on various AI subfields efficiently. In addition, there is a benchmark that is on the intersection of empirical experiments and theory [6].
2. Algorithm components can also be completely replaced by AI components and thus increase efficiency [26, 17, 8]. This can come with reduced theoretical guarantees since the learned components are typically based on neural networks.

Open problems for dynamic configurations include:

1. How to scale DAC to large configuration spaces (i.e. many algorithm parameters)? Most current approaches deal with less than five parameters and some even struggle to perform well on more than two parameters.
2. How to decide on which instances one should learn in a dynamic AAC strategy? An efficient learning strategy is required since dynamic configuration needs to perform well on new instances. Approaches from curriculum and self-paced learning could be promising [9].
3. When should DAC be applied? How to efficiently determine whether dynamic configuration is beneficial beyond static configuration?

#### 4.3.4 Conclusions

Although there is active research on AAC methods for more than a decade and there is substantial progress (e.g., tremendous speedups, efficient scaling to more algorithm parameters (at least for static AAC) and dealing with different kinds of instances), there are still many open challenges that are not yet addressed fully and open up many further research directions.

## 4.4 Working Group 4: Utility Functions

*Kevin Leyton-Brown (University of British Columbia, Canada, kevinlb@cs.ubc.ca)*

**License** © Creative Commons BY 4.0 International license  
© Kevin Leyton-Brown

This working group focused on how to apply a utilitarian perspective to the problem of algorithm configuration. We discussed the benefits of different (random vs stratified) sampling strategies to estimate utility value from samples. We considered the impact of a utilitarian approach on algorithm configuration methods that offer theoretical guarantees. We also discussed candidate utility functions that could be suggested to practitioners. In the end, we recommend a simple approach:

1. Is there a minimum amount of time  $t_0$  below which all algorithm runs should be considered equivalent?
2. Is there a maximum amount of time  $\bar{\kappa}$  above which the end user would never be willing to wait?
3. In between  $t_0$  and  $\bar{\kappa}$ , how does our preference for time change? We foresee three common cases:
  - a.  $t_0 = \bar{\kappa}$ : we have a step function
  - b. linearly (e.g., we face a fixed cutoff and pay per hour of compute)
  - c. geometrically (utility falls by a constant fraction per unit time)

## 4.5 Working Group 5: New Estimation-of-Distribution Algorithms

*Manuel López-Ibáñez (University of Manchester, UK, manuel.lopez-ibanez@manchester.ac.uk)*

**License** © Creative Commons BY 4.0 International license  
© Manuel López-Ibáñez

This working group focused on estimation-of-distribution algorithms (EDAs) and how they could be used for AAD.

#### 4.5.1 Discussed Problems

- What are the challenges for EDAs in the AAD scenarios:
  1. The search space is mixed (continuous, integer, categorical).
  2. There exists a hierarchy between the parameters, in the form of known *a priori* conditions.
  3. There might be unknown stochastic dependencies between parameters.
  4. The problem can be noisy, with non-Gaussian noise.
  5. Since we often consider multiple instances, this is another area in which a different kind of noise gets introduced.
- How can the sampling be made more efficient? In sig-cGA, some variables are fixed when you are sure they should have some particular value (good theoretical results but unclear experimental results).
- How to create good synthetic functions?
- How to handle the conditional parameters? Can we just ignore the condition and sample values which have no effect (which results in a random walk of the respective values)?

#### 4.5.2 Possible Approaches

- Can we use the Bayesian optimization algorithm (BOA) with Bayesian networks (or P3 or GOMEA)? Can the dependency network be created, and could this handle mixed-integer variables?
  - The algorithms in questions should be capable to represent the dependency network well up to a degree.
  - EDAs are also capable to handle mixed-integer problems, but we came to no conclusion how well this would work for our purpose.
- Univariate EDAs require far less work to build a model. For certain contexts, e.g., where it is very hard to figure out actual dependencies, this might already be sufficient. We have some theoretical guarantees for how to choose the parameters of univariate EDAs, preventing them from converging prematurely to bad values.
- Density estimation trees or forests (e.g., the `mlpack` package): how to update the trees?
  - We did not come to a good conclusion.
- Use CMA-ES with handling of mixed-integer spaces (except categorical). There is a version of CMA-ES that can adapt the size of the covariance matrix.
- We can use techniques from deep reinforcement learning to learn the update function of CMA-ES. Is there a structural prior to use to speed up this training? Could we apply algorithm design, define a grammar, and apply genetic programming to this context?
  - We were indecisive about the next steps.

#### 4.5.3 Conclusions

This area is heavily underexplored so far. Some AAD approaches, like irace, exist that can be thought of as EDAs. However, they do not make use of theoretical guarantees. Partially because the theoretical results are too recent, partially because we have no guarantees for multivariate EDAs. Hence, employing complex EDAs should follow proper empirical tests.

## 4.6 Working Group 6: The OneMax Problem of AAD

*Benjamin Doerr (Ecole Polytechnique, IP Paris, FR, lastname@lix.polytechnique.fr)*

License © Creative Commons BY 4.0 International license  
© Benjamin Doerr

The aim of this working group was to bring together researchers in AAD and in the theory of evolutionary computation to discuss what could be an AAD problem that is simple enough to allow analyses via mathematical means, but for which a deeper understanding promises to lead to useful insights on AAD methods used in practice.

The hope is that such a first simple problem can trigger a success story similar to the theoretical analysis of evolutionary algorithm, which has started with toy problems like how the  $(1 + 1)$  evolutionary algorithm optimizes the OneMax problem [2, 12, 23], but which now, for example, is able to analyze complex algorithms intensively used in practice like the NSGA-II [4], the most prominent multi-objective evolutionary algorithm (50,000 citations).

Obviously, what is such a good first problem is highly non-obvious and the group discussed various directions. It was noted that some first works already exist [14, 15, 16], however, no participant of the breakout session had a deeper understanding of these works. This together with short preliminary readings of these works were interpreted in the way that most likely the problems regarded in these works do not yet perfectly satisfy the needs of the automated algorithm design community.

Finally, the following problem set-up was proposed. We try to optimize a pseudo-Boolean function  $f : \{0, 1\}^n \rightarrow \mathbb{R}$ . This function is defined via two functions  $g_0, g_1$  defined on  $n - 1$  bits via  $f(x) = g_0(x_2, \dots, x_n)$  if  $x_1 = 0$  and  $f(x) = g_1(x_2, \dots, x_n)$  if  $x_1 = 1$ . This models an automated algorithm design problem in which we have the choice between two algorithms  $A_0$  and  $A_1$ . This choice is described via the variable  $x_1$ . We further have  $n - 1$  binary algorithm parameters. For  $i = 0, 1$ , if we run algorithm  $A_i$  with parameters values  $x_2, \dots, x_n$  on our problem, the performance is  $f(i, x_2, \dots, x_n) = g_i(x_2, \dots, x_n)$ . In this model, we assume that we have access to a precise and noise-free performance of the algorithms on our problem instance. We do, a priori, not assume that  $g_0$  and  $g_1$  are in some sense related, though most likely the more interesting results will be obtained if there is some correlation between the influences of the parameters on the result. An example for such a setting could be that both functions are linear functions with coefficients that mostly have the same sign. Also, it was pointed out that this model captures the situation that the two algorithms have different numbers of parameters, namely by letting one of the two functions only depend on some of the variables.

A particular research question that could be studied in this model is the following. To try to find out which algorithm with which parameters to use for our problem, is it better to optimize the parameters of the two algorithms separately, that is, we optimize  $g_0$  and  $g_1$  separately and take the better of the two results, or is it preferable to just optimize  $f$ , that is, let the algorithm design process switch between the algorithms.

It was later noted that a special case of the function  $f$  was regarded in [29].

## 4.7 Working Group 7: Explainability of AAD Methods

*Pascal Kerschke (Technical University of Dresden, DE, [pascal.kerschke@tu-dresden.de](mailto:pascal.kerschke@tu-dresden.de))*

License  Creative Commons BY 4.0 International license  
© Pascal Kerschke

This working group focused on the explainability of AAD, starting from the question of whether / why we would want to have explainable AAD.

### 4.7.1 Discussed Problems

- Everyone interprets explainability in their own way, ranging from understanding complementarity of algorithms and explaining the autoML process itself to explaining experimental data such as identification of flaws or biased data/setup of the process.
- Overarching question: when / why care for explainability if we just want better-performing algorithms?
- There are different levels of explainability: designers vs. users. For example, users want explainability to gain trust in the AAD frameworks / tools. Explanations are also always contextualized in your own (domain) knowledge, so different users might need different explanations.
- Can explainability help identify what parts of a problem are relevant, or help to identify useful guidelines from the vast amount of data collected by AAD approaches? Explanations might help understand the complexity of a model, e.g. that some HPO problems have low effective dimensionality [3].
- What are the limits of explainability: can we sufficiently explain very complex models with high fidelity? Do explanation techniques have sufficient explanatory power?
- To what extent are features useful for explainability? If we use features, should we limit ourselves only to those we fully understand and build simple models based on those?
- We need to be careful of the Rashomon effect: many possible explanations might exist based on spurious interactions, high correlation of features,...
- Diagnostic tools for the existing interpretable/explainable ML methods and theoretical proofs are still missing. Many of the IML methods are still very new, and not yet fully trusted.

### 4.7.2 Possible Approaches

- One potential way to help explainability is to use functional ANOVA or Generalized Additive Models for decomposition of the model.
- There exists a wide range of techniques which might be useful for the explainability of AAD: fANOVA, Symbolic Formulas, (automatic) Ablations, Partial Dependency Plots, Item Response Theory, Instance Footprint plots, Local Parameter Importance, landscape features, feature selection, simple explainable models, regularized models in expensive settings...

### 4.7.3 Conclusions

Explainability in general is gaining more attention in many communities, and there are ways in which AAD might also benefit from more explainable techniques. It is however still unclear when to focus on explainability over just improving the performance of the final algorithm. The quick growth of explainability techniques gives rise to a lot of opportunities, but we should not lose sight of the overall picture and reason about what we want to explain, why and to whom.

## 4.8 Working Group 8: Automated Machine Learning

*Marius Lindauer (Leibniz University Hannover, DE, m.lindauer@ai.uni-hannover.de)*

*Bernd Bischl (LMU, DE, bernd.bischl@stat.uni-muenchen.de)*

License © Creative Commons BY 4.0 International license  
© Marius Lindauer and Bernd Bischl

This working group focused on current challenges and promising future directions for automated machine learning (AutoML).

### 4.8.1 Discussed Problems

1. Although the efficiency of AutoML has improved by orders of magnitude in recent years, AutoML might still be too inefficient for very expensive models (e.g., LLMS).
2. AutoML is not always robust; e.g., a simple default pipeline sometimes (rarely) outperforms a complex AutoML system. How can users trust AutoML systems that the invested compute (and waiting) time is worth it?
3. AutoML still does not handle all kinds of pitfalls of ML. For example, (I) class imbalance is not properly dealt with in all tools, (II) wrong or inefficient validation schemes were used or (III) ensembling is not beneficial.
4. The user interfaces of many AutoML tools are not user-friendly. It is an open question of how verbose the output of an AutoML tool should be and what kind of warnings are required (e.g., class imbalance and fairness). Generally, designing good user interfaces without specifying the target user group is hard, but AutoML aims to be an approach for everyone.
5. AutoGluon [11] is very fast and delivers strong performance (in particular on tabular data for supervised tasks). Is AutoML for this subtask already solved or can we outperform it by using drastically different approaches?
6. Large language models (LLMS) pose a major challenge since they are expensive and combine several learning paradigms. Current AutoML approaches are hardly applicable [28].
7. The hidden objectives of practitioners are not always explicitly expressed. For example, some users care only about predictive performance and others want to have rather small and interpretable models. Not all AutoML tools can address different user preferences; e.g., AutoGluon returns neither small nor interpretable models.
8. How can AutoML help to understand the data in a better way (e.g., in data-centric approaches)?
9. How can we design an interactive AutoML system that (I) allows users to learn from the AutoML system about the AutoML process and the data, and (II) to specify their preferences, (III) benefits from users' input to improve the overall performance of the system (given the users' objectives)? This might be important s.t. practitioners do not get the impression that these systems can get out of control, but they are still in charge and thus can trust the system.
10. Surprisingly, there seems to be a large group of practitioners rather using optimizers for AutoML problems but fewer the full-fledged AutoML systems. These optimizers offer much more flexibility but require more expertise and effort to set them up. What is missing in full AutoML systems s.t. more practitioners want to use them? Learning search spaces on the fly could be one promising direction [25, 24].
11. Large language models (such as ChatGPT) could offer new ways to interact with AutoML systems (or partially replace them) s.t. users will need even less knowledge about how to code ML systems [31].

### 4.8.2 Conclusions

Although AutoML is much more mature than it was a few years ago, there are still major challenges in applying AutoML in practice without expertise in ML and AutoML. One of the most crucial next steps will be to bridge the gap between the expectations, expertise, and workflows of ML practitioners and AutoML tools.

### References

- 1 Steven Adriaensen, André Biedenkapp, Gresa Shala, Noor H. Awad, Theresa Eimer, Marius Lindauer, and Frank Hutter. Automated dynamic algorithm configuration. *Journal of Artificial Intelligence Research*, 75:1633–1699, 2022.
- 2 Thomas Bäck. Optimal mutation rates in genetic search. In *Proc. of ICGA '93*, pages 2–8, 1993.
- 3 James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.
- 4 Weijie Zheng, Yufei Liu, and Benjamin Doerr. A first mathematical runtime analysis of the Non-Dominated Sorting Genetic Algorithm II (NSGA-II). In *Proc. of AAAI '22*, pages 10408–10416, 2022.
- 5 André Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, and Marius Lindauer. Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *Proc of ECAI '20*, pages 427–434, 2020.
- 6 André Biedenkapp, Nguyen Dang, Martin S. Krejca, Frank Hutter, and Carola Doerr. Theory-inspired parameter control benchmarks for dynamic algorithm configuration. In *Proc. of GECCO '22*, pages 766–775, 2022.
- 7 Aymeric Blot, Holger H. Hoos, Laetitia Jourdan, Marie-Éléonore Kessaci-Marmion, and Heike Trautmann. Mo-paramils: A multi-objective automatic algorithm configuration framework. In *Proc. of LION '10*, pages 32–47, 2016.
- 8 Chris Cameron, Jason S. Hartford, Taylor Lundy, Tuan Truong, Alan Milligan, Rex Chen, and Kevin Leyton-Brown. Monte carlo forest search: UNSAT solver synthesis via reinforcement learning. *CoRR*, abs/2211.12581, 2022.
- 9 Theresa Eimer, André Biedenkapp, Frank Hutter, and Marius Lindauer. Self-paced context evaluation for contextual reinforcement learning. In *Proc. of ICML '21*, pages 2948–2958, 2021.
- 10 Theresa Eimer, André Biedenkapp, Maximilian Reimer, Steven Adriaensen, Frank Hutter, and Marius Lindauer. DACBench: A benchmark library for dynamic algorithm configuration. In *Proc. of IJCAI '21*, pages 1668–1674, 2021.
- 11 Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *CoRR*, abs/2003.06505, 2020.
- 12 Josselin Garnier, Leila Kallel, and Marc Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7:173–203, 1999.
- 13 Devon R. Graham, Kevin Leyton-Brown, and Tim Roughgarden. Formalizing preferences over runtime distributions. In *Proc. of ICML '23*, pages 11659–11682, 2023.
- 14 George T. Hall, Pietro S. Oliveto, and Dirk Sudholt. On the impact of the cutoff time on the performance of algorithm configurators. In *Proc. of GECCO '19*, pages 907–915. 2019.
- 15 George T. Hall, Pietro S. Oliveto, and Dirk Sudholt. Analysis of the performance of algorithm configurators for search heuristics with global mutation operators. In *Proc. of GECCO '20*, pages 823–831. 2020.
- 16 George T. Hall, Pietro S. Oliveto, and Dirk Sudholt. Fast perturbative algorithm configurators. In *Proc. of PPSN '20*, pages 19–32. 2020.



- 17 André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. In *Proc. of ECAI '20*, pages 443–450, 2020.
- 18 Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION '11*, pages 507–523, 2011.
- 19 Frank Hutter, Manuel López-Ibáñez, Chris Fawcett, Marius Lindauer, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. AClib: A benchmark library for algorithm configuration. In *Proc. of LION '14*, pages 36–40, 2014.
- 20 Marius Lindauer, Katharina Eggersperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23:54:1–54:9, 2022.
- 21 Marius Lindauer, Matthias Feurer, Katharina Eggersperger, André Biedenkapp, and Frank Hutter. Towards assessing the impact of bayesian optimization’s own hyperparameters. *CoRR*, abs/1908.06674, 2019.
- 22 Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.
- 23 Heinz Mühlenbein. How genetic algorithms really work: mutation and hillclimbing. In *Proc. of PPSN '92*, pages 15–26, 1992.
- 24 Felix Neutatz, Marius Lindauer, and Ziawasch Abedjan. Automl in heavily constrained applications. *CoRR*, abs/2306.16913, 2023.
- 25 Valerio Perrone and Huibin Shen. Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In *Proc. of NeurIPS '19*, pages 12751–12761, 2019.
- 26 Gresa Shala, André Biedenkapp, Noor H. Awad, Steven Adriaensen, Marius Lindauer, and Frank Hutter. Learning step-size adaptation in CMA-ES. In *Proc. of PPSN '20*, pages 691–706, 2020.
- 27 David Speck, André Biedenkapp, Frank Hutter, Robert Mattmüller, and Marius Lindauer. Learning heuristic selection with dynamic algorithm configuration. In *Proc. of ICAPS '21*, pages 597–605, 2021.
- 28 Alexander Tornede, Difan Deng, Theresa Eimer, Joseph Giovanelli, Aditya Mohan, Tim Ruhkopf, Sarah Segel, Daphne Theodorakopoulos, Tanja Tornede, Henning Wachsmuth, and Marius Lindauer. AutoML in the age of large language models: Current challenges, future opportunities and risks. *CoRR*, abs/2306.08107, 2023.
- 29 Ingo Wegener and Carsten Witt. On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. *Journal of Discrete Algorithms*, 3:61–78, 2005.
- 30 Gellért Weisz, András György, Wei-I Lin, Devon R. Graham, Kevin Leyton-Brown, Csaba Szepesvári, and Brendan Lucier. ImpatientCapsAndRuns: Approximately optimal algorithm configuration from an infinite pool. In *Proc. of NeurIPS '20*, 2020.
- 31 Lei Zhang, Yuge Zhang, Kan Ren, Dongsheng Li, and Yuqing Yang. MLCopilot: Unleashing the power of large language models in solving machine learning tasks. *CoRR*, abs/2304.14979, 2023.



## Participants

- Thomas Bäck  
Leiden University, NL
- Bernd Bischl  
LMU München, DE
- Chris Cameron  
University of British Columbia –  
Vancouver, CA
- Nguyen Dang  
University of St Andrews, GB
- Benjamin Doerr  
Ecole Polytechnique – Palaiseau,  
FR
- Carola Doerr  
Sorbonne University – Paris, FR
- Tome Eftimov  
Jozef Stefan Institute –  
Ljubljana, SI
- Marcus Gallagher  
The University of Queensland –  
Brisbane, AU
- Nikolaus Hansen  
INRIA Saclay –  
Palaiseau, FR
- Pascal Kerschke  
TU Dresden, DE
- Lars Kotthoff  
University of Wyoming –  
Laramie, US
- Martin S. Krejca  
Ecole Polytechnique –  
Palaiseau, FR
- Johannes Lengler  
ETH Zürich, CH
- Kevin Leyton-Brown  
University of British Columbia –  
Vancouver, CA
- Marius Lindauer  
Leibniz Universität  
Hannover, DE
- Shengcai Liu  
A\*STAR – Singapore, SG
- Manuel López-Ibáñez  
University of Manchester, GB
- Katherine M. Malan  
UNISA – Pretoria, ZA
- Mario Andrés Muñoz  
The University of Melbourne, AU
- Nelishia Pillay  
University of Pretoria, ZA
- Shinichi Shirakawa  
Yokohama National  
University, JP
- Thomas Stützle  
Free University of Brussels, BE
- Niki van Stein  
Leiden University, NL
- Diederick Vermetten  
Leiden University, NL
- Marcel Weber  
LMU München, DE
- Carsten Witt  
Technical University of Denmark  
– Lyngby, DK

