ADA Research Group
Technical Report TR-2018-1

Automated Artificial Intelligence (AutoAI)

Holger H. Hoos
Leiden Institute of Advanced Computer Science (LIACS)
Universiteit Leiden
The Netherlands

24 December 2018

Abstract

While there has been research on artificial intelligence (AI) for at least 50 years, we are now standing on the threshold of an AI revolution – a transformational change whose effects may surpass that of the industrial revolution in the first half of the 19th century. There are multiple reasons why AI is rapidly gaining traction now. Firstly, much of our infrastructure is already controlled by computers; so deploying AI systems is technologically quite straightforward. Secondly, in many situations, there is now easy access to large amounts of data, which can be used as a basis for customising AI systems using machine learning. Thirdly, due to tremendous improvements not only in computer hardware, but also in AI algorithms, advanced AI systems can now be deployed broadly and at low cost.

As a result, AI systems are poised to fundamentally change the way we live and work. AI is quickly becoming a major driver of innovation, growth and competitiveness, and is bound to play a crucial role in addressing the challenges we face individually and as societies. However, high-quality AI systems require considerable expertise to build, maintain and operate. For the foreseeable future, AI expertise will be a limiting factor in the broad deployment of AI systems, and – unless managed very carefully – this will lead to uneven access and increasing inequality. It is also likely to cause the wide-spread use of low-quality AI systems, developed without the proper expertise.

Here, we propose to address this problem using AI methods – specifically, automated algorithm design, machine learning and optimisation techniques – to help build and deploy the next generation of AI systems. This gives rise to an approach we refer to as *automated artificial intelligence (AutoAI)*. Ultimately, research on AutoAI aims to make it possible for people who benefit from AI to develop, deploy and maintain AI systems that are performant, robust and predictable, without requiring deep and highly specialised AI expertise. AutoAI will thus dramatically broaden access to high-quality AI systems.

1 Introduction

Effective AI algorithms and systems are difficult to design, implement and maintain. They also require a high level of expertise to deploy successfully in practice, and more stringent requirements on performance, robustness, predictability and safety will make this even harder in the future. A marked shortness of AI experts already causes a considerable bottleneck, and in the near future, this will make it hard to develop and deploy AI in small and medium-size companies, in public service and in nonprofit organisations. Because of the game-changing nature of AI, this causes serious problems. Firstly, competition for AI expertise

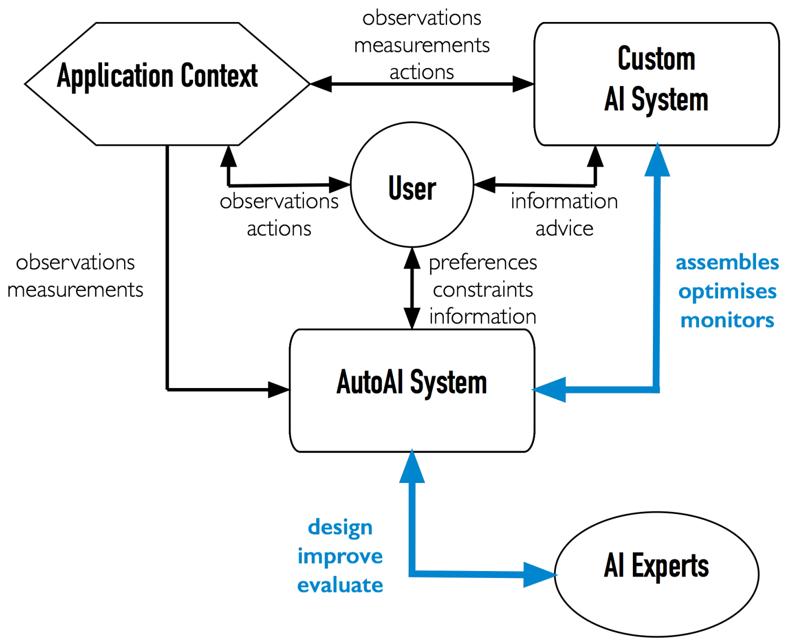

Figure 1: Overview of the AutoAI approach. Rather than directly producing specialised AI systems for specific application situations, AI experts design and maintain an AutoAI system that enables users to develop and deploy customised AI systems tailed to their needs and preferences.

as a limiting resource increases inequity; successful AI deployment can greatly improve the profitability of businesses, and hence their ability to attract more AI expertise, leading to further increases in their competitive advantage. At the level of entire nations and societies, a similar effect occurs, as increased productivity due to effective use of AI-enabled automation boosts the ability to invest in AI education and training, which then brings about further increases in productivity. Secondly, in the face of limited expertise in the area, AI systems will be developed and deployed by non-experts; such systems can be expected to suffer from major shortcomings, not only in terms of performance, but more problematically, in terms of robustness, predictability and safety – highly desirable properties that even today's AI experts find difficult to achieve.

To alleviate this *AI complexity problem*, we propose a radical, yet logical solution: *Use AI methods to help build and deploy the next generation of AI systems.* We refer to this idea as *automated AI* (short: *AutoAI*) and aim to realise it as a system of assistants that help people develop, deploy and maintain AI systems that are performant, robust and predictable, *without requiring deep and highly specialised AI expertise*. Ultimately, research on AutoAI aims to make it possible for those who benefit from their use to build, calibrate and maintain effective, robust and safe AI systems, entirely without the need for AI experts or software developers.

To reach this goal will require a very substantial research effort, but – leveraging several existing lines of work – it should be possible to realise limited prototypes quite swiftly, and thus to clearly demonstrate the potential of the approach and lay the foundation for advanced AutoAI systems. The overall AutoAI approach is illustrated in Figure 1; notice how the AutoAI system, in interaction with the user, takes over some of the tasks currently carried out by AI experts. By encapsulating and fully formalising expert knowledge in AI, the AutoAI system operationalises that knowledge on a much broader scale than achievable by human experts alone.

2 An illustrative example: AutoAI for smart agriculture

To see how AutoAI could change the way we develop, deploy and maintain AI systems, consider the following example. Imagine a collective of farmers who want to maximise sustainable production of their fields. These farmers would like to use an AI agricultural consultant – *i.e.*, an AI system that can provide timely and relevant advice that complements their own expertise – to help them reach their goal. While they regularly use smart phones, tablets and computers to check weather forecasts and crop prices, and even to coordinate the use of shared equipment, none of them can write software or has any experience with building, using or maintaining AI systems, nor can they afford to hire experts who can do it for them. However, they have access to an advanced AutoAI system and off-the-shelf devices and services to collect data on their fields and use of equipment. Using an AutoAI data collection and integration assistant, they first realise a software platform that allows them to monitor their fields and equipment, using natural-language dialogue to provide the information needed by the assistant to synthesise the platform.

Next, they expand this platform with machine learning capabilities that allow them to forecast yield under various conditions, which helps them decide on types of crops, fertiliser use, planting and harvesting patterns. To do this, they make use of an AutoAI machine learning assistant that guides them in selecting and calibrating suitable machine learning methods, in preparing training data, and in interpreting predictions. The machine learning system synthesised by the assistant is tailored specifically to the situation and needs of this specific group of farmers, while drawing on generic information collected from similar use contexts (*e.g.*, farming applications elsewhere). It does not merely produce predictions, but also clearly indicates the confidence it has in these predictions, generates warnings when that confidence is too low, provides suspected reasons for its low performance and suggests remedies (*e.g.*, collect more or different types of data; allow more time for selecting and calibrating learning procedures; or give access to greater computational resources).

Finally, our farmers use an AutoAI optimisation and planning assistant to extend their system with the capability to guide them in maximising their production, subject to the resources at their disposal, individual and collective preferences, as well as anticipated environmental conditions. Again, the assistant uses natural-language dialogue to elicit the required information, and selects, adapts and integrates system components based on the specific needs and preferences of our farmers. The resulting, custom AI system does not replace the farmers, but rather provides them with effective guidance towards realising their goal of maximising sustainable production. It combines learning, planning and constraint solving components; it interacts on a regular basis with its users – reactively and proactively – to adjust to changing conditions and unexpected developments; and it self-monitors and warns about limited confidence, also suggesting remedies.

3 Background and related work

AI systems are currently constructed by highly skilled experts, in a process that critically relies on their experience and intuition. There are several broad areas of work aimed at automating some aspects of the underlying algorithm design process.

Automated algorithm configuration, selection and portfolio construction has been gaining traction within the AI community over the last decade, but is restricted to performance optimisation when solving a single, precisely defined problem, such as propositional satisfiability, a particular flavour of AI planning or mixed integer programming [see, *e.g.*, 19, 62, 21, 61, 57, 24, 51]. Some recent work on algorithm configuration draws from collections of semantically equivalent, interchangeable components [see, *e.g.*, 27].

Hyperparameter optimisation, model selection and neural architecture search also optimise performance for a given class of learning tasks (*e.g.*, supervised classification) and form part of a fast-moving research direction known as *automated machine learning (AutoML)* [see, *e.g.*, 56, 12, 33, 9]. Recently, Google released their Cloud AutoML services [15], which produce custom deep learning models for computer vision, text categorisation and translation, using neural architecture search (NAS). While details are proprietary, highly related publicly accessible research suggests that the design space is rather small and can be explored quite effectively with random search, although reinforcement learning achieves better results for larger time budgets [64]. There are clear indications that by moving to NAS methods based on sequential model-based optimisation, as used in our own work on AutoML [56, 33], substantial further improvements can be achieved [40]. While clearly motivated similarly to AutoAI, Google's Cloud AutoML is restricted to specific learning tasks and to using deep neural networks; to broaden the scope to that of AutoAI, as envisioned here, challenges similar to those outlined in the following would have to be overcome.

Automatic Statistician aims to automate data analysis using statistical and machine learning techniques [54, 41, 28]. Like AutoAI, it is motivated by making these techniques accessible to non-experts, but is focussed more narrowly on data science, and specifically, regression problems. Interestingly, this line of work includes the generation of natural-language, technical reports (similar to more specialised work on automated scaling analysis [44]).

Another very interesting line of research, aimed at automating key processes in data science, deals with automated data preprocessing and cleaning techniques [see, *e.g.*, 60] and with automatically learning constraints from structured data [see, *e.g.*, 7, 31, 48]. This work forms an important basis for AutoAI, which will make use of automated data preprocessing and benefit from the ability to infer constraints or other structured models from examples.

Genetic programming aims to synthesise software with given functionality, using techniques from evolutionary computation, such as genetic algorithms or evolution strategies. Work in this area has achieved some impressive results [see, *e.g.*, 46, 47, 32, 63] and given rise to the active area of search-based software engineering [see, *e.g.*, 16, 5, 49], which aims to automated key tasks of software engineering. However, so far, this line of work has not achieved the ability to synthesise performant solvers for any widely studied AI problem, let alone a broad range of AI problems, as encountered in the context of AutoAI. Nevertheless, techniques from search-based software engineering will likely contribute to achieving a practical AutoAI system.

There is limited work on the synthesis of effective algorithms for classes of AI problems that show significant semantic differences, such as scheduling problems with different constraints and objectives. The Aeon system by Monette *et al.* [42] synthesises algorithms for a broad range of scheduling problems from high-level specifications, using a combination of an expert-designed classification system for scheduling problems and a carefully crafted mapping of problem classes and user-selected solving approaches to fully instantiated scheduling algorithms. While Aeon starts from user-specified high-level formalisations, it is limited to scheduling problems and does not carry out automatic performance optimisation. Still, it gives a first glimpse of some of the capabilities we aim for with AutoAI. It is also closely related to other work in constraint programming (CP) aimed at producing a general-purpose modelling and solving system, sometimes referred to as "the holy grail of CP" [see, *e.g.*, 13].

Finally, as we will explain in the following, our vision of AutoAI prominently includes the ability to assess the performance of AI algorithms and systems before, during and after running them. This aspect extends several lines of work in automated performance prediction [see, *e.g.*, 25, 12, 37, 46].

4 Research challenges in AutoAI

The overall goal of AutoAI is to make it possible for those who benefit from using AI systems to be able to build, calibrate and maintain effective, robust and safe AI systems, without the need for AI experts or software developers. To reach this goal, a major leap beyond the current state of the art in artificial intelligence is needed, and specifically, the following challenges have to be addressed:

- How to automatically construct a performant, trustworthy AI system for a specific use case based on high-level information provided by the user?
- How to assess the performance of an AI algorithm or system, as it used in a specific situation (data, hardware/software platform), especially if that situation differs from those available while it was built and calibrated? How to automatically build this assessment ability into an AI system?
- How to make sure that AI systems, especially ones that are automatically constructed, are as simple as possible, while still working well for their intended use?
- How to make sure that AutoAI can be applied to a broad range of problem domains and situations, especially ones that require combinations of techniques from different areas of AI?

To meet these challenges, several methodological advances are required. Notably, methods are needed for

- automatically assessing and monitoring the performance of AI systems and their components, so that assessment and monitoring mechanisms can be automatically generated and added to a broad range of AI systems;
- automatically configuring AI systems for performance and robustness in a given use case, taking into account not only the kind of data likely to be encountered after deployment, but also resources availability and user preferences;
- assisting designers, maintainers and users of AutoAI systems in trading off software complexity against performance, to achieve simpler systems that are easier to maintain, cheaper to run and less likely to contain bugs;
- automatically assembling high-quality AI systems for a given use case from components, *i.e.*, systems that perform well even when processing data deviating from that used for initial configuration, and that can monitor and assess their own performance.

In the following, we discuss these in some more detail, sketching out plausible paths towards developing such methods, drawing on several lines of existing work.

4.1 From performance prediction to "self-aware" AI techniques

For AI systems to be trustworthy, they need to signal clearly when they "get out of their depth", *i.e.*, when their output (information, advice, actions) should be treated with caution or becomes entirely unreliable. This is especially important for automatically constructed AI systems, where there is no trusted expert to inspire confidence. To realise this "self-assessment" capability, techniques are needed that can gauge the degree of confidence we should have in the output produced by a given AI algorithm when applied to a specific situation (data) – before, during or after running the algorithm. The design of such techniques can draw from existing work in *automated performance prediction* [see, *e.g.*, 25], which already forms the basis for automated algorithm selection, configuration and, in particular, state-of-the-art automated machine learning techniques.

Specifically, this would require the development of next-generation empirical performance models (EPMs) that provide more accurate performance predictions as well as better estimates for the uncertainty associated with these predictions, leveraging automated machine learning (AutoML) techniques [see, *e.g.*, 12, 33]. Furthermore, work should be undertaken to overcome major limitations of current EPMs, notably their extremely limited capability to generalise to data deviating from the distribution on which they have been trained, by leveraging work on generative adversarial learning [14] and learning extrapolable functions [see, *e.g.*, 29, 58, 35]. Finally, leveraging these next-generation EPMs and active learning techniques [see, *e.g.*, 53], which complement each other well in this context, we see a clear path for developing methods for synthesising monitoring mechanisms for given AI algorithms and systems.

4.2 Advanced automated algorithm design

At the core of AutoAI lies the idea of automating the design of AI algorithms and systems. This will require advanced automated algorithm design methods, just as automated algorithm configuration methods [see, *e.g.*, 22] provide the basis for existing AutoML approaches [12, 33]. Towards this end, we envision three major lines of research.

Firstly, we see the need for automated algorithm configuration techniques that are orders of magnitude faster than the current state of the art, because current techniques are too costly for an effective AutoAI approach, especially when taking into account the need for thorough empirical evaluation using advanced statistical techniques. To realise such next-generation algorithm configuration techniques, it seems promising to leverage automated analysis of the combinatorial landscapes searched by algorithm configuration procedures [see, *e.g.*, 50], by automatically selecting between multiple configuration techniques, and by developing novel, 'grey-box' techniques that exploit information about the inner workings of the algorithm being optimised.

Secondly, it will be necessary to devise algorithm configuration and selection techniques that can handle multiple design objectives, resource constraints (such as limitations in memory or computational power) and user preferences. This is important, because the design choices made when building an AI system for a given purpose typically depend on user preferences (what kind of prediction accuracy is desired or needed?) and resource constraints (which kind of device is the system going to run on?); also, there often is more than one design objective (*e.g.*, we want high prediction accuracy of an ML system, but also need the system to run fast on hardware with limited capabilities), and users of the AI system being automatically assembled should be able to explore tradeoffs between competing objectives. To achieve these goals, it should be possible to build on our recent proof-of-concept work on multi-objective algorithm configuration [4]; to leverage the advanced performance prediction methods outlined earlier in order to deal with resource constraint; and to incorporate user preferences in the form of priors and hard constraints into fundamental mechanisms used for algorithm selection and configuration.

Thirdly, AutoAI will require algorithm selection and configuration methods capable of producing results (*i.e.*, design choices) that are robust to deviations from the distributions of data used for training, since we want to produce AI systems that perform well even when the precise circumstances for which they were optimised change. Work on such methods could build on existing approaches for empirical scaling analysis [43, 44], as well as on techniques from transfer and reinforcement learning [see, *e.g.*, 26, 37, 12, 10]. Furthermore, the desired robustness could, at least in part, be achieved by using per-instance algorithm configuration techniques that effectively switch between different configurations of a given AI algorithm or system, depending on characteristics of the given input data, and by synthesising on-line controllers for specific parameters whose values should be adapted while an algorithm, such as an automated reasoning engine, is solving a specific instance of an AI problem.

4.3 Automatic design simplification

In many cases, we do not want to achieve performance at the cost of excessive complexity of an AI algorithm or system. (Here, complexity intuitively refers to size or number of components of a piece of software.) The reason is that complex systems have a higher risk of failure, *e.g.*, due to bugs in components; they also tend to be more difficult to configure robustly for high performance, due to the increased risk of overfitting to given training data.

We therefore see a need to develop methods for the automated simplification of the systems and algorithms produced by an AutoAI system. Specifically, there are three approaches that can lead to such methods: one based on tentatively removing components or replacing them with simpler ones while maintaining performance characteristics; one aimed at allowing the user to explore tradeoffs between the complexity and performance of the AI systems or algorithms constructed by AutoAI; and the third focussed on methods for helping the designers and maintainers of the AutoAI system to keep that system as simple as possible, particularly with respect to the number and complexity of the components it draws on when automatically assembling AI software. The pursuit of these directions can strongly build on existing work on parameter importance analysis [11, 20]; on work from multi-objective optimisation [4, 8], including advances in multi-objective algorithm configuration outlined earlier; and on active testing [36].

4.4 Automated synthesis of AI systems

To fully realise the vision of AutoAI, we need to develop methods that allow users without expertise in AI or software design to build AI systems. One approach towards this goal is based on the following idea. (1) Scripted, but natural language dialogue is used to elicit information from the user. (2) Based on this information, a rule-based procedure assembles a template of the desired system, along with a specification of semantically equivalent components that can be used to instantiate it, as well as user preferences and constraints on possible instantiations. Finally, (3) advanced algorithm design procedures are used to find a valid instantiation that can be expected to perform well, and finally, the resulting design choices are used to instantiate the skeleton, which completes the assembly of the system. To implement this approach, it seems promising to combine ideas from work on the synthesis of scheduling algorithm by Monette *et al.* [42] with advanced automated algorithm design techniques and mechanisms that elicit crucial information about the desired functionality, available data and computational infrastructure from the user. This requires collections of flexible, parametric AI system components for a range of broadly applicable tasks, such as data preprocessing, supervised and unsupervised learning, automated planning and reasoning, general-purpose search and optimisation, along with formal specifications based on which abstract templates for these can be combined into instantiable templates for a broad range of use cases.

5 Possible concerns and objections

In this section we discuss a number of concerns and objections we expect to be raised regarding the concept of AutoAI and its realisation, in the form of questions and answers.

Can AI systems really compete with human experts when it comes to building AI systems? Yes, there is limited evidence that this might be possible, in existing work on AutoML as well as automated algorithm selection and configuration for important and widely studied AI problems. In addition, we do not have to fully replace human expertise, but just to augment and leverage it; specifically, much human AI expertise can be captured in the construction of an AutoAI system and then leveraged when that system is used to automatically construct custom AI solutions for given use cases. This is analogous to the way in which compilers capture human expertise in generating efficient machine code and leverage this on a massive scale. Just like the best human experts might be better in producing highly optimised machine code than a state-of-the-art compiler, human AI experts might be able to produce better results than an AutoAI system at least for some time to come; however, the AI systems constructed by an AutoAI system might be good enough to be effectively alleviate the shortness of AI expertise in many application areas and democratise the use of AI.

Isn't this simply AI applied to AI? Essentially yes, but that doesn't make it easy. Meta-learning is an interesting special case and known to be hard; AutoAI is far more general and challenging for several reasons. Firstly, in a typical use case for AutoAI, there is likely to be less data relative to the number of design choices and parameters; this is already the case for the far more restricted case of AutoML, where overfitting is known to be a challenge [see, *e.g.*, 56]. Secondly, by applying AI to the design of algorithms and systems for solving complex problems, we have to deal with even higher computational cost than encountered in most other AI applications.

Won't AutoAI make AI systems less transparent and human-understandable? No. AI systems constructed by AutoAI should in many cases be simpler (in terms of fewer components), thanks to the use of automated simplification techniques and the ability to realise full performance potential of simple designs – which is challenging for human AI experts, as is evident, *e.g.*, from the efficacy of automated hyperparameter optimisation of simple machine learning methods. Furthermore, key parts of the design process for AI systems that are conventionally based on human intuition are fully formalised in AutoAI, making them easier to analyse, challenge and understand. Through the ability to constrain the automated design process to certain choices and components that are easier to understand for humans, AutoAI makes it easier to realise explainable AI.

Won't AutoAI exaggerate inequity caused by AI? No; while AutoAI techniques and systems need to be built by human experts, their use will require less AI expertise, facilitating broad application and better leverage of human expertise. Furthermore, by encapsulating much of the specialised expertise required for building conventional AI systems into the AutoAI framework, we amplify the reach of that scarce expertise and greatly expand the group of individuals and organisations that can benefit from it. To ensure broad access to the benefits of this new way of constructing AI systems, it will be necessary, however, to develop AutoAI in publically funded projects, rather than to leave this area completely to industry,

Won't this lead to the loss of even more jobs through automation? No. There is a general expectation that there will be a shortage of AI experts in the foreseeable future. Compensating for jobs lost as a result of AI deployment / automation by means of training more AI experts is somewhat unrealistic, since the PhD-level expertise currently needed to develop effective and trustworthy AI systems is difficult and costly to obtain. AutoAI will make it possible for a much broader pool of talent to effectively develop, deploy and maintain AI systems and thus create, rather than destroy jobs.

Doesn't this increase the risk of run-away AI? No. Although AutoAI does provide some techniques that would necessarily be required for any kind of general AI (notably some of the self-assessment and self-improvement capabilities outlined earlier), research on AutoAI is not aimed at producing general, human-level AI. Instead, the ultimately goal is to make it possible for people without deep and highly specialised AI expertise to develop, deploy and maintain AI systems that are performant, robust and predictable. The AI systems thus obtained will be limited in scope and, in fact, highly customised to the needs of their users. They will also be constructed taking into account the users' preferences and constraints. The AutoAI system that designs them will necessarily have access to a broad range of AI techniques and use powerful techniques to assemble customised systems from them. However, neither the former nor the latter will be geared towards general, human-level intelligence. Overall, our vision of AutoAI is closely aligned with the goals of human-centred AI, notably with the emphasis on AI systems and techniques that complement, rather than replace, human intelligence and capabilities.

6 Conclusions

Just as the future of machine learning lies, to a significant extent, in the development and broad use of AutoML techniques, the future of AI lies in an analogous direction we dub *AutoAI*. AutoAI not only holds the key to addressing the talent bottleneck (which incurs a serious risk of broad deployment and use of poor-quality AI systems), but also provides a path towards broad access to predictable, robust and performant AI systems, and thus advances democratisation of AI.

We strongly believe that by building on several lines of active research, it is possible in the near future to design working prototypes of AutoAI systems. Meeting the challenges arising in this context requires methological advances that will have broad benefits beyond the area of AutoAI; for example, the "self-aware" AI techniques outlined in Section 4.1 would doubtlessly be useful in the context of traditional approaches to the design, deployment and operation of AI systems.

Overall, we are convinced that AutoAI offers substantial benefits, along with a wealth of intriguing research challenges. We therefore expect AutoAI to quickly become one of the most vibrant and fastest-moving research areas within AI, sustained by a large and energetic community of researchers and practitioners. We hope that the vision and ideas outlined in this report will help to initiate this effort and provide some guidance to those interested in pursuing it.

Acknowledgements: Some of the ideas presented in this document have been discussed with members of the ADA Research Group and colleagues at LIACS; I specifically thank Mitra Baratchi, Jan van Rijn, Aske Plaat and Simon Portegies-Zwart for valuable comments.

References

[1] Roberto Amadini, Maurizio Gabbrielli, and Jacopo Mauro. Sunny: a lazy portfolio approach for constraint solving. *Theory and Practice of Logic Programming*, 14(4-5):509–524, 2014.

[2] Earl T Barr, Mark Harman, Yue Jia, Alexandru Marginean, and Justyna Petke. Automated software transplantation. In *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, pages 257–269. ACM, 2015.

[3] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Masahiro Fujita, and Yunshan Zhu. Symbolic model checking using sat procedures instead of bdds. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 317–320. ACM, 1999.

[4] Aymeric Blot, Holger H. Hoos, Laetitia Vermeulen-Jourdan, Marie-Éléonore Kessaci-Marmion, and Heike Trautmann. MO-ParamILS: a multi-objective automatic algorithm configuration framework. In *Proceedings of the 10th International Conference on Learning and Intelligent Optimization (LION 10)*, 2016.

[5] Thelma Elita Colanzi, Silvia Regina Vergilio, Wesley Klewerton Guez Assuno, and Aurora Pozo. Search based software engineering: Review and analysis of the field in Brazil. *Journal of Systems and Software*, 86(4):970 – 984, 2013. SI : Software Engineering in Brazil: Retrospective and Prospective Views.

[6] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[7] Luc De Raedt, Andrea Passerini, and Stefano Teso. Learning constraints from examples. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18)*, pages 7965–7970, 2018.

[8] Kalyanmoy Deb. Multi-objective optimization. In *Search methodologies*, pages 403–449. Springer, 2014.

[9] Katharina Eggensperger, Marius Thomas Lindauer, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Efficient benchmarking of algorithm configuration procedures via model-based surrogates. *Machine Learning*, 107:15–41, 2018.

[10] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 1436–1445, July 2018.

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

[12] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS-15)*, pages 2755–2763, Cambridge, MA, USA, 2015. MIT Press.

[13] Eugene C. Freuder. Progress towards the holy grail. *Constraints*, 23(2):158–171, 2018.

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[15] Google. Cloud AutoML website. `https://cloud.google.com/automl/`, last visited on 30 August 2018.

[16] Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza, and Shin Yoo. *Search Based Software Engineering: Techniques, Taxonomy, Tutorial*, pages 1–59. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[17] Holger H. Hoos. Programming by optimization. *Communications of the ACM*, 55(2):70–80, 2012.

[18] Holger H Hoos, Tomáš Peitl, Friedrich Slivovsky, and Stefan Szeider. Portfolio-based algorithm selection for circuit qbfs. In *International Conference on Principles and Practice of Constraint Programming*, pages 195–209. Springer, 2018.

[19] Frank Hutter, Domagoj Babic, Holger H. Hoos, and Alan J. Hu. Boosting verification by automatic tuning of decision procedures. In *Formal Methods in Computer-Aided Design, 7th International Conference, FMCAD 2007, Austin, Texas, USA, November 11-14, 2007, Proceedings*, pages 27–34, 2007.

[20] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 754–762. JMLR.org, 2014.

[21] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Automated configuration of mixed integer programming solvers. In *Proceedings of the 7th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2010)*, pages 186–202, 2010.

[22] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION 5)*, pages 507–523, 2011.

[23] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[24] Frank Hutter, Marius Lindauer, Adrian Balint, Sam Bayless, Holger H. Hoos, and Kevin Leyton-Brown. The configurable SAT solver challenge (CSSC). *Artificial Intelligence*, 243:1–25, 2017.

[25] 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 Panos M. Pardalos, Mauricio G.C. Resende, Chrysafis Vogiatzis, and Jose L. Walteros, editors, *Learning and Intelligent Optimization*, pages 36–40, Cham, 2014. Springer International Publishing.

[26] K. Jamieson and A. Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS-16)*, pages 240–248, 2016.

[27] Ashiqur R. KhudaBukhsh, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Satenstein: Automatically building local search SAT solvers from components. *Artificial Intelligence*, 232:20–42, 2016.

[28] Hyunjik Kim and Yee Whye Teh. Scaling up the Automatic Statistician: Scalable structure discovery using Gaussian processes. In Amos Storkey and Fernando Perez-Cruz, editors, *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS-18)*, pages 575–584, 2018.

[29] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR-17)*, April 2017.

[30] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*. Addison-Wesley, Reading, MA, USA, 2015.

[31] Samuel Kolb, Sergey Paramonov, Tias Guns, and Luc De Raedt. Learning constraints in spreadsheets and tabular data. *Machine Learning*, 106(9):1441–1468, Oct 2017.

[32] Pavel Kordík, Jan Černý, and Tomáš Frýda. Discovering predictive ensembles for transfer learning and meta-learning. *Machine Learning*, 107(1):177–207, Jan 2018.

[33] Lars Kotthoff, Chris Thornton, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, 18:25:1–25:5, 2017.

[34] Wolfgang Küchlin and Carsten Sinz. Proving consistency assertions for automotive product data management. *Journal of Automated Reasoning*, 24(1-2):145–163, 2000.

[35] Rui Leite and Pavel Brazdil. Active testing strategy to predict the best classification algorithm via sampling and metalearning. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 309–314, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[36] Rui Leite, Pavel Brazdil, and Joaquin Vanschoren. Selecting classification algorithms with active testing. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 117–131. Springer, 2012.

[37] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 2016.

[38] Marius Lindauer, Holger H. Hoos, Kevin Leyton-Brown, and Torsten Schaub. Automatic construction of parallel portfolios via algorithm configuration. *Artificial Intelligence*, 244:272–290, 2017.

[39] Marius Thomas Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. AutoFolio: an automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015.

[40] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *arXiv preprint arXiv:1712.00559*, 2017.

[41] James Robert Lloyd, David K Duvenaud, Roger B Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, pages 1242–1250, 2014.

[42] Jean-Noël Monette, Yves Deville, and Pascal Van Hentenryck. Aeon: Synthesizing scheduling algorithms from high-level models. In John W. Chinneck, Bjarni Kristjansson, and Matthew J. Saltzman, editors, *Operations Research and Cyber-Infrastructure*, pages 43–59, Boston, MA, 2009. Springer US.

[43] Zongxu Mu, Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. On the empirical scaling of running time for finding optimal solutions to the TSP. *Journal of Heuristics*, 2018 (to appear).

[44] Zongxu Mu and Holger H. Hoos. Empirical scaling analyser: An automated system for empirical analysis of performance scaling. In Sara Silva and Anna Isabel Esparcia-Alcázar, editors, *Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015, Companion Material Proceedings*, pages 771–772. ACM, 2015.

[45] Neil Newman, Alexandre Fréchette, and Kevin Leyton-Brown. Deep optimization for spectrum repacking. *Communications of the ACM*, 61(1):97–104, 2017.

[46] Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the 2016 Annual Conference on Genetic and Evolutionary Computation (GECCO-16)*, pages 485–492, 2016.

[47] Randal S. Olson, Ryan J. Urbanowicz, Peter C. Andrews, Nicole A. Lavender, La Creis Kidd, and Jason H. Moore. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, pages 123–137. Springer International Publishing, 2016.

[48] Sergey Paramonov, Samuel Kolb, Tias Guns, and Luc De Raedt. Tacle: Learning constraints in tabular data. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management (CKIM-17)*, pages 2511–2514, 2017.

[49] Justyna Petke, Mark Harman, William B Langdon, and Westley Weimer. Specialising software for different downstream applications using genetic improvement and code transplantation. *IEEE Transactions on Software Engineering*, 44(6):574–594, 2018.

[50] Yasha Pushak and Holger H. Hoos. Algorithm configuration landscapes: More benign than expected? In *Proceedings of the 15th International Conference on Parallel Problem Solving from Nature (PPSN-18)*, 2018, to appear.

[51] Mattia Rizzini, Chris Fawcett, Mauro Vallati, Alfonso Emilio Gerevini, and Holger H. Hoos. Static and dynamic portfolio methods for optimal planning: An empirical analysis. *International Journal on Artificial Intelligence Tools*, 26(1):1–27, 2017.

[52] Hannes Schwarz, Lars Kotthoff, Holger Hoos, Wolf Fichtner, and Valentin Bertsch. Using automated algorithm configuration to improve the optimization of decentralized energy systems modeled as large-scale, two-stage stochastic programs. *Annals of Operations Research*, 2018 (to appear).

[53] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[54] Automatic Statistician. website. `https://cloud.google.com/automl`, last visited on 30 August 2018.

[55] Paul Stephan, Robert K Brayton, and Alberto L Sangiovanni-Vincentelli. Combinational test generation using satisfiability. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(9):1167–1176, 1996.

[56] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 847–855. ACM, 2013.

[57] Mauro Vallati, Chris Fawcett, Alfonso Gerevini, Holger H. Hoos, and Alessandro Saetti. Automatic generation of efficient domain-optimized planners from generic parametrized planners. In *Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS 2013, Leavenworth, Washington, USA, July 11-13, 2013.*, 2013.

[58] Jan N. van Rijn, Salisu Mamman Abdulrahman, Pavel Brazdil, and Joaquin Vanschoren. Fast algorithm selection using learning curves. In Elisa Fromont, Tijl De Bie, and Matthijs van Leeuwen, editors, *Advances in Intelligent Data Analysis XIV*, pages 298–309, Cham, 2015. Springer International Publishing.

[59] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[60] Gust Verbruggen and Luc De Raedt. Towards automated relational data wrangling. In *Proceedings of AutoML 2017 @ ECML-PKDD: Automatic selection, configuration and composition of machine learning algorithms*, pages 18–26, 2017.

[61] Lin Xu, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Evaluating component solver contributions to portfolio-based algorithm selectors. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 228–241. Springer, 2012.

[62] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008.

[63] Steven R. Young, Derek C. Rose, Thomas P. Karnowski, Seung-Hwan Lim, and Robert M. Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, MLHPC '15, pages 4:1–4:5, New York, NY, USA, 2015. ACM.

[64] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *arXiv preprint arXiv:1707.07012*, 2017.