Computer-Aided Algorithm Design: Automated Tuning, Configuration, Selection, and Beyond
(Invited Talk Abstract)

Holger H. Hoos
Department of Computer Science
University of British Columbia
hoos@cs.ubc.ca

High-performance algorithms can be found at the heart of many software systems; they often provide the key to effectively solving the computationally difficult problems encountered in the application areas in which these systems are deployed. Examples of such problems include planning, scheduling, timetabling, resource allocation, computer-aided design and software verification. Many of these problems are NP-hard and considered computationally intractable; nevertheless, these 'intractable' problems arise in practice, and finding good solutions to them in many cases tends to become more difficult as economic constraints tighten.

In most (if not all) cases, the key to solving such computationally challenging problems lies in the use of high-performance heuristic algorithms, that is, algorithms that make use of mechanisms whose efficacy can be demonstrated empirically, yet remains inaccessible to the analytical techniques used for proving theoretical complexity results.

High-performance heuristic algorithms are typically constructed in an iterative, manual process in which the designer gradually introduces or modifies components or mechanisms whose performance is then tested by empirical evaluation on one or more sets of benchmark problems. During this iterative design process, the algorithm designer has to make many decisions, ranging from choices of the heuristic mechanisms to be used and the details of these mechanisms to lower-level implementation details, such as data structures. Some of these choices take the form of parameters, whose values are guessed or determined based on limited experimentation.

This traditional approach for designing high-performance algorithms can and often does lead to satisfactory results. However, it tends to be tedious and labour-intensive; furthermore, the resulting algorithms are often unnecessarily complicated, yet fail to realise the full performance potential present in the space of designs that can be built using the same underlying set of components and mechanisms.

As an alternative to the traditional, manual algorithm design process, we advocate an approach that uses fully formalised procedures, implemented in software, to permit a human designer to explore large design spaces more effectively, with the aim of realising algorithms with desirable performance characteristics. *Computer-aided algorithm design* allows human designers to focus on the creative task of specifying a design space in terms of potentially useful components. This design space is then explored using optimisation and machine learning techniques, in combination with significant amounts of computing power, in order to find algorithms that perform well on given sets or distributions of input instances (Hoos 2008).

Automated parameter tuning, algorithm configuration, algorithm portfolios and per-instance algorithm selection are prominent special cases of computer-aided algorithm design and have recently played a pivotal role in improving the state of the art in solving a broad range of challenging combinatorial problems, ranging from propositional satisfiability (SAT) and mixed integer programming to protein structure prediction, course timetabling and planning problems.

In this talk, I will introduce computer-aided algorithm design and discuss its main ingredients: *design patterns*, which provide ways of structuring potentially large spaces of candidate algorithms, and *meta-algorithmic optimisation procedures*, which are used for finding good designs within these spaces. After explaining how this algorithm design approach differs from and complements related approaches in program synthesis, genetic programming and so-called hyper-heuristics, I will illustrate its success using examples from our own work in SAT-based software verification (Hutter et al. 2007), timetabling (Chiarandini, Fawcett, and Hoos

2008) and mixed integer programming (Hutter, Hoos, and Leyton-Brown 2010). Furthermore, I will argue why this approach can be expected to be particularly useful and effective for building better solvers for rich and diverse classes of combinatorial problems, such as planning and scheduling. Finally, I will outline out how *programming by optimisation* – a design paradigm that emphasises the automated construction of performance-optimised algorithm by means of searching large spaces of alternative designs – has the potential to transform the design of high-performance algorithm from a craft that is based primarily on experience and intuition into a principled and highly effective engineering effort.

References

Chiarandini, M.; Fawcett, C.; and Hoos, H. 2008. A modular multiphase heuristic solver for post enrollment course timetabling (extended abstract). In *Proc. PATAT 2008*. 6 pages.

Hoos, H. 2008. Computer-aided design of high-performance algorithms. Technical Report TR-2008-16, University of British Columbia, Department of Computer Science. 6 pages.

Hutter, F.; Babić, D.; Hoos, H.; and Hu, A. 2007. Boosting verification by automatic tuning of decision procedures. In *FMCAD 2007*, 27–34.

Hutter, F.; Hoos, H.; and Leyton-Brown, K. 2010. Automated configuration of mixed integer programming solvers. In *Proc. CPAIOR 2010 (to appear)*.