Artificial Intelligence 232 (2016) 20–42

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

SATenstein: Automatically building local search SAT solvers from components

Ashiqur R. KhudaBukhsh [a,*,1], Lin Xu [b,1], Holger H. Hoos [b], Kevin Leyton-Brown [b]

[a] Department of Computer Science, Carnegie Mellon University, United States
[b] Department of Computer Science, University of British Columbia, Canada

ARTICLE INFO

Article history:
Received 18 December 2013
Received in revised form 5 November 2015
Accepted 7 November 2015
Available online 2 December 2015

Keywords:
SAT
Stochastic local search
Automatic algorithm configuration

ABSTRACT

Designing high-performance solvers for computationally hard problems is a difficult and often time-consuming task. Although such design problems are traditionally solved by the application of human expertise, we argue instead for the use of automatic methods. In this work, we consider the design of stochastic local search (SLS) solvers for the propositional satisfiability problem (SAT). We first introduce a generalized, highly parameterized solver framework, dubbed SATenstein, that includes components drawn from or inspired by existing high-performance SLS algorithms for SAT. The parameters of SATenstein determine which components are selected and how these components behave; they allow SATenstein to instantiate many high-performance solvers previously proposed in the literature, along with trillions of novel solver strategies. We used an automated algorithm configuration procedure to find instantiations of SATenstein that perform well on several well-known, challenging distributions of SAT instances. Our experiments show that SATenstein solvers achieved dramatic performance improvements as compared to the previous state of the art in SLS algorithms; for many benchmark distributions, our new solvers also significantly outperformed all automatically tuned variants of previous state-of-the-art algorithms.

1. Introduction

In Mary Shelley's classic novel *Frankenstein; or, The Modern Prometheus*, a brilliant scientist, Victor Frankenstein, set out to create a perfect human being by combining scavenged human body parts. We pursue a similar idea: scavenging components from existing high-performance algorithms for a given problem and combining them to build new high-performance algorithms. Our idea is inspired by the fact that many new solvers are created by augmenting an existing algorithm with a mechanism found in a different algorithm (see, e.g., [33,53]) or by combining components of different algorithms (see, e.g., [62]). Unlike Victor Frankenstein's creation, we propose to use an automated construction process that enables us to optimize performance with minimal human effort.

* Corresponding author.
E-mail addresses: akhudabu@cs.cmu.edu (A.R. KhudaBukhsh), xulin730@cs.ubc.ca (L. Xu), hoos@cs.ubc.ca (H.H. Hoos), kevinlb@cs.ubc.ca (K. Leyton-Brown).
[1] We thank Paul Cernek for helping to run experiments and contributing to the SATenstein codebase. Ashiqur R. KhudaBukhsh and Lin Xu contributed equally to this work.

http://dx.doi.org/10.1016/j.artint.2015.11.002
0004-3702/

Traditionally, high-performance heuristic algorithms are designed through an iterative, manual process in which most design choices are fixed at development time, usually based on preliminary experimentation, leaving only a small number of parameters exposed to the user. In contrast, we propose a new approach to heuristic algorithm design in which the designer fixes as few design choices as possible, instead exposing all promising design choices as parameters. This approach removes from the algorithm designer the burden of making early design decisions without knowing how different algorithm components will interact on problem distributions of interest. Instead, it encourages the designer to consider many alternative designs, drawing from known solvers as well as novel mechanisms. Of course, such flexible, highly parameterized algorithms must be instantiated appropriately to achieve good performance on a given instance set. With the availability of advanced automated parameter configurators and cheap computational resources, finding a good parameter configuration from a huge parameter space becomes practical (see, e.g., [40,9,13]). Of course, we are not the first to propose building algorithms by using automated methods to search a large design space. Rather, our work can be seen as part of a general and growing trend, fueled by an increasing demand for high-performance solvers for difficult combinatorial problems in practical applications, by the desire to reduce the human effort required for building such algorithms, and by an ever-increasing availability of cheap computing power that can be harnessed for automating parts of the algorithm design process (see also [34]). There are many examples of work along these lines [25,56,12,79,20,18,60,77,57,21].

Although our general approach is not specifically tailored to a particular domain, in this work we address the challenge of constructing stochastic local search (SLS) algorithms for the propositional satisfiability problem (SAT): an NP-complete problem of great interest to the scientific and industrial communities alike. SLS-based solvers are important because they have exhibited consistently dominant performance for several families of SAT instances; they also play an important role in state-of-the-art portfolio-based automated algorithm selection methods for SAT [79]. Substantial research and engineering effort has been expended in building SLS algorithms for SAT since the late 1980s (see, e.g., [67,33,62]), with new solvers being introduced every year.

We leveraged this rich literature (discussed in detail later) to design `SATenstein-LS`. This algorithm draws mechanisms from 25 high-performance SLS SAT solvers and also incorporates many novel strategies. The resulting design space contains a total of 2.01×10^{14} candidate solvers, and includes most existing, state-of-the-art SLS SAT solvers that have been proposed in the literature. We demonstrate experimentally that our new, automatically-constructed solvers dramatically outperform the best SLS-based SAT solvers currently available (with the default parameter configurations manually tuned by their authors) on six well-known SAT instance distributions, ranging from hard random 3-SAT instances to SAT-encoded factoring and software verification problems. In most cases, our new solvers also significantly outperform the best SLS-based SAT solvers even when we automatically tune the originally exposed parameters of every one of these incumbent solvers. Because SLS-based SAT solvers are the best known methods for solving most of our benchmark distributions, our new solvers represent a substantial advance in the state of the art for solving the respective sub-classes of SAT. On one of the two instance families for which this is not the case—SAT-encoded number factoring problems—our new solvers narrow the gap between the performance of the best SLS algorithms and the best DPLL-based solvers.

This paper[2] is organized as follows. Section 2 discusses related work; we describe the design and implementation of `SATenstein-LS` in Section 3. We then describe the setup we used for empirically evaluating `SATenstein-LS` (Section 4) and present the results from our experiments (Section 5). Section 6 presents some general conclusions and an outlook on future work.

2. Related work

The propositional satisfiability problem (SAT) asks, for a given propositional formula F, whether there exists a complete assignment of truth values to the variables of F under which F evaluates to true (see, e.g., [8]). F is called satisfiable if there exists at least one such assignment and unsatisfiable otherwise. A SAT instance is usually represented in conjunctive normal form (CNF), i.e., as a conjunction of disjunctions of literals, where each literal is a propositional variable or the negation of variables. Each disjunction of literals is called a clause. In this case, the goal for a SAT solver is to find a variable assignment that satisfies all clauses of a given CNF formula or to prove that no such assignment exists.

2.1. Local-search SAT solvers

Over the past decades, considerable research and engineering effort has been invested into designing and optimizing algorithms for SAT. State-of-the-art SAT solvers include tree-search algorithms (see, e.g., [69,28,7,16,2,30]), local search algorithms (see, e.g., [42,61,50,64,62,11]) and resolution-based preprocessors (see, e.g., [70,15,3]). Every year, competitions are held, in which new state-of-the-art solvers emerge. The trend of continuing performance improvement in SAT competitions suggests that there is room for even further enhancements of current solver technology.

[2] An early version of the work described in this article was published at IJCAI [45]. This article substantially extends that work in five key ways. (1) It describes `SATenstein-LS`'s architecture in considerably more detail, and (2) presents all-new experiments based on longer configuration runs, albeit on the same distributions. (3) Our comparison with tuned versions of challengers (Section 5.2) is entirely new, as is (4) our comparison with complete solvers (Section 5.3). (5) We extended `SATenstein-LS` with a local search strategy found in the recent high-performance SAT solver, `Sattime`, and compared the performance of the augmented `SATenstein-LS` with three recent SLS-based SAT solvers, including `Sattime`.