This paper appears in the *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, NAGOYA, Aichi, Japan, 1997.

Ten Challenges in Propositional Reasoning and Search

Bart Selman, Henry Kautz, and **David McAllester**
AT&T Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974
{selman, kautz, dmac}@research.att.com
http://www.research.att.com/~selman/challenge

Abstract

The past several years have seen much progress in the area of propositional reasoning and satisfiability testing. There is a growing consensus by researchers on the key technical challenges that need to be addressed in order to maintain this momentum. This paper outlines concrete technical challenges in the core areas of systematic search, stochastic search, problem encodings, and criteria for evaluating progress in this area.

1 Introduction

Propositional reasoning is a core problem in many areas of artificial intelligence. In recent years this area has seen growing interest and activity, due to advances in our ability to solve large problem instances, including ones that encode real-world problems such as planning and diagnosis. Contributions to the area of propositional deduction and satisfiability testing have come from research communities in artificial intelligence, operations research, and theoretical computer science. A set of key technical challenges have begun to emerge from interactions within and among these research groups. We will describe some of these challenges and discuss why they are important for progress in the field.

For many years the problem of propositional reasoning received relatively little attention. It appeared that nothing could be done to improve upon the performance of the Davis-Putnam procedure (1960), which held its position as the most efficient satisfiability testing algorithm for three decades. Furthermore, most researchers in artificial intelligence felt that the representational power of propositional logic was too limited, and turned their attention to first-order logic or even more powerful formalisms, such as various modal and non-monotonic logics. Several factors contributed to a renewed interest in propositional reasoning. First, new algorithms were discovered, including ones based on stochastic local search as well as systematic search, that have better scaling properties than the basic Davis-Putnam algorithm. Second, improvements in machine speed, memory size, and implementations extended the range of the algorithms. Third, researchers began to develop and solve propositional encodings of interesting, real-world problems such as planning and diagnosis, with others on the horizon, such as natural language processing and machine learning. Such encodings were not even being considered a few years ago because they were thought to be far too large to be handled by any method. Between 1991 and 1996 the size of hard satisfiability problems that could be feasibly solved grew from ones involving less than 100 variables to ones involving over 10,000 variables.

Progress has been spurred by the interaction of researchers in AI, operations research, and theory, in making available benchmark problems (DIMACS, see Trick and Johnson 1996), holding joint workshops (Ginsberg 1995), and sharing algorithms and code. There is a growing consensus, however, that certain key technical challenges must be addressed in order to continue to increase the range of problems that can be practically solved. Work on propositional reasoning can be grouped in three core areas: systematic search; stochastic search; and problem encodings. For each area, we will discuss concrete challenges that have arisen. We stress that these challenges are not particularly our own invention, but rather summarize the issues that frequently arise in informal discussions among researchers. Progress on any of these challenges would directly extend the usefulness of the propositional reasoning approach to problems in AI.

Because of the large number of references involved, we have not attempted to cite all related literature. However, a web page for these challenge problems has been constructed with a more complete bibliography, further details on the challenge problems, and pointers to existing satisfiability testing procedures. See http://www.research.att.com/~selman/challenge.

With each challenge, we give our best estimate of a time frame within which we think the challenge can be met. For the harder challenges, even partial solutions will constitute a significant contribution to the field.

2 Evaluation Criteria

The main evaluation criteria that has been adopted in the satisfiability testing (SAT) community is empirical performance on shared benchmark problems. We believe that this remains the best way to evaluate responses to any of the algorithmic challenges we will present. The alternative, evaluation by theoretical analysis, is problematic. Worst-case complexity results are all usually exponential. Furthermore, theoretical average case results are difficult to obtain, and can even be misleading because of the difficulty of formally characterizing problem distributions. Even when a sound theoretical analysis is devised, it may only characterize the asymptotic performance of the algorithm, and that asymptote may lie too far out (*e.g.*, to problems containing trillions of variables) to be of practical consequence. Of course, theoretical analysis can be a very useful *complement* to empirical evaluation, and can help us gain insights into *why* an algorithm works well or poorly on a given problem distribution. (In fact, one of our non-algorithmic challenges, as we will see, is to develop a theoretical analysis that explains *why* local search works well on certain problem distributions.)

Reasonably good benchmark collections of satisfiability problems, mainly in CNF form, are publically available from a number of sources. These include the DIMACS collection, which was used as a testbed for a large number of algorithms as reported in (Trick and Johnson 1996); a collection of circuit diagnosis problems encoded as SAT (Larrabee 1992); the planning problems Kautz and Selman (1996) devised for their SATPLAN system; hardware and software verification problems from the model checking community; and others. All these collections have the property that published papers exist specifying the best known algorithms and running times for solving each of their instances.

Progress on any of our challenges (when applicable) will be measured by showing that the proposed method outperforms all other known methods on at least *one* set of benchmark problems. Some of our challenges in the area of problem encodings involve finding better ways to represent a real-world problem as SAT. In these cases, the empirical evaluation should involve comparing performance of the best SAT algorithms on the proposed encoding against the best published results for *any* algorithm that can solve the original (*i.e.*, unencoded) problem instances. In such a comparison it is usually not possible to *outperform* the specialized algorithms (although it is sometimes possible, see for example (Kautz and Selman 1996)), but one should provide evidence that solving the encoded problem with general purpose Boolean satisfiability procedures is at least competitive.

Whenever possible, comparisons should also be made to the difficulty of solving the problems when alternative encodings are used, such as may be found in some of the benchmark sets. In cases where alternative encodings are available, one should show that the new encoding yields problems that are easier to solve by at least one state-of-the-art SAT procedure. (For example, it would be counted as progress if one developed a new encoding of planning problems that was good for stochastic search procedures, even if the encoding did not help systematic search procedures.)

In order to keep the criteria for comparison as objective as possible, it is important that results include total running time. This may be adjusted for machine speed, but it is not sufficient to *only* report certain characteristics of the execution, such as "number of nodes expanded" (Johnson 1996). There are many ways to shift the computational effort in search algorithms — for example, to visit fewer nodes by doing more work at each node — and an objective evaluation must consider the entire picture. Comparisons should cite the best results from the Operations Research and computer science theory literature, as well as the AI literature.

Tests should be done on a variety of problem sizes, up to the hardest available instances that the method can solve. It is important to show the limits of the proposed method, that is, where it becomes highly exponential or otherwise fails. Often methods that look good on small instances break down on larger ones (Johnson 1996).

3 Challenging SAT Problems

We shall begin the list of challenges by citing two specific open SAT problems. The first is to develop a way to *prove* that unsatisfiable 700 variable 3-CNF formulas, randomly generated in the "hard" region where the ratio of variables to clauses is 4.3, are in fact unsatisfiable (Mitchell *et al.* 1992; Crawford and Auton 1993). Randomly-generated *satisfiable* formulas of this size are regularly solved by stochastic search algorithms such as GSAT, but they are unable to prove unsatisfiability; and no systematic algorithm has been able to solve hard random formulas of this size. A generator for these hard random problem instances can be found at the DIMACS benchmark archive (Trick and Johnson 1996).

CHALLENGE 1: (1-2 yrs) *Prove that a hard 700 variable random 3-SAT formula is unsatisfiable.*

The second challenge problem is satisfiable. It is an encoding of a 32-bit parity problem, that appears in the DIMACS benchmark set. It appears to be too large for current systematic algorithms. It also defeats the hill-climbing techniques used by current local search algorithms.

CHALLENGE 2: (2-5 yrs) *Develop an algorithm that finds a model for the DIMACS 32-bit parity problem.*

For the second challenge, of course, the algorithm should not be told in advance the known solution! Given the amount of effort that has been spent on these two instances, any algorithm solving one or both will have to do something significantly different from current methods.

4 Challenges for Systematic Search

In the previous section we described two challenging SAT problems. In this section, and the next one, we describe promising ideas whose utility has not yet been demonstrated. In each case the challenge is to demonstrate the utility of a currently known, but as yet unproven, approach to solving SAT problems. In keeping with the discussion in section 2, to demonstrate the utility of an idea one must construct a procedure which uses that idea in an essential way in solving at least one class of problems more effectively than any other known approach.

Our next challenge is using proof systems stronger than resolution. All of the best systematic methods for propositional reasoning are based on creating a resolution proof tree. This includes depth-first search algorithms such as the Davis-Putnam procedure, where the proof tree can be recovered from the trace of the algorithm's execution, but is not explicitly represented in a data structure (the algorithm only maintains a single branch of the proof tree in memory at any one time). Most work on systematic search concentrates on heuristics for variable-ordering and value selection, all in order to the reduce size of the tree.

However, there are known fundamental limitations on the size of the shortest resolution proofs for certain problems (Haken 1985; Chvatal and Szemeredi 1988). For example, "pigeon hole" problems (showing that n pigeons cannot fit in $n-1$ holes) are intuitively easy, but shortest resolution refutation proofs are of exponential length. Shorter proofs do exist in more powerful proof systems. Examples of proof systems more powerful than resolution include extended resolution, which allows one to introduce new defined variables, and resolution with symmetry-detection, which uses symmetries to eliminate parts of the tree without search. Assuming $NP \neq co-NP$, even the most powerful propositional proof systems would require exponential long proofs worst case — nonetheless, such systems provably dominate resolution in terms of minimum proof size.

However, at this point in time, attempts to mechanize these more powerful proof systems usually yield no computational savings, because it is *harder* to find the small proof tree in the new system, than to simply crank out a large resolution proof. In essence, the overhead in dealing with the more powerful rules of inference consumes all the potential savings. There is promising work in this area (Crawford *et al.* 1996; de la Tour and Demri 1995), but not yet convincing empirical results on a variety of benchmark problems; such evidence would meet our third challenge:

CHALLENGE 3: (2-5 yrs) *Demonstrate that a propositional proof system more powerful than resolution can be made practical for satisfiability testing.*

Our next challenge is the use of integer programming techniques in solving SAT problems. It is straightforward to translate SAT problems into 0-1 integer programming problems (Hooker 1988), and thus it has been argued that integer programming techniques should be useful for propositional reasoning. In fact, however, this has not been shown to be the case. For example, one of the main techniques in integer programming is to compute the *linear relaxation* of the problem, and then to use the (easily-found) solution of the relaxed problem to guide the selection of values in solving the integer problem. However, in most formulations the solution to the linear relaxation of *any* SAT problem simply sets all the variables to the value "1/2" (modulo unit propagation), thus yielding no guidance at all. Therefore we offer a challenge to show that the well-developed body of tools and techniques from Operations Research *does* in fact have something new to offer for propositional reasoning.

CHALLENGE 4: (2-5 yrs) *Demonstrate that integer programming can be made practical for satisfiability testing.*

5 Challenges for Stochastic Search

Stochastic local search has been shown to be a powerful alternative to systematic search for finding models of satisfiable CNF formulas. Stochastic algorithms are inherently incomplete, because if they fail to find a model for a formula one cannot be *certain* that the formula is unsatisfiable. This has led to an asymmetry in our ability to solve satisfiable and unsatisfiable instances drawn from the same problem distribution. Stochastic algorithms can solve hard random satisfiable formulas containing thousands of variables, but we cannot solve unsatisfiable instances of the same size.

Can local search be made to work for proving unsatisfiability? This apparently would require searching in space of refutation proofs, rather than in space of truth assignments. Each state would be an incomplete proof tree, *e.g.*, a proof tree that rules out some fraction of the truth assignments. The neighborhood of a state would be similar proof trees. The step in the local search would try to transform the proof into one that rules out a larger fraction of assignments.

CHALLENGE 5: (5-10 yrs) *Design a practical stochastic local search procedure for proving unsatisfiability.*

Our next challenge is that of distinguishing "dependent" from "independent" variables. SAT encodings of structured problems such as planning and diagnosis often contain large numbers of variables whose values are constrained to be a simple Boolean function of other variables. We call these *dependent* variables. Variables whose values can not be easily determined to be a simple function of other variables are called independent. For a given SAT problem there may be many different ways to classify the variables as dependent and independent. But for most SAT encodings of real-world problems there is a natural division between dependent and independent variables. Since an assignment to the independent variables determines a truth value for each dependent variable, the number of assignments that need be considered in a systematic search is at most 2^n where n is the number of *independent* variables.

We belief that it fairly easy to empirically demonstrate that identification of dependent variables improves the performance of systematic search. It appears to be more difficult to establish that identification of dependent variables improves stochastic search. The challenge, therefore, is to improve local search for problems with dependent variables: the search should concentrate only (or mainly) on the independent variables, and some fast mechanism should then set the dependent variables.

CHALLENGE 6: (1-2 yrs) *Improve stochastic local search on structured problems by efficiently handling variable dependencies.*

As we have noted, systematic and local search procedures outperform each other on different problem classes. Even putting the issue of incompleteness aside, there exist classes of satisfiable problems for which one or the other approach is clearly the winner. This leads to the general question: Can we develop a procedure that leverages the strengths of each? The obvious way, of course, is to simply run good implementations of each approach in parallel. But is there a more powerful way of combining the two? Recently there has been some intriguing work on using local search to implement the variable ordering heuristic for systematic search (Boufkhad 1996; Mazure *et al.* 1996). These methods and other ways of combining systematic and stochastic search need to be further developed and compared with previous approaches on a range of benchmark problems.

CHALLENGE 7: (1-2yrs) *Demonstrate the succesful combination of stochastic search and systematic search techniques, by the creation of a new algorithm that outperforms the best previous examples of both approaches.*

6 Challenges for Problem Encodings

The value of research on propositional reasoning ultimately depends on our ability to find suitable SAT encodings of real-world problems. As discussed in the introduction, there has been significant recent progress along this front. Examples include classical constraint-based planning (Blum and Furst 1995; Kautz and Selman 1996), problems in finite algebra (Fujita *et al.* 1993), verification of hardware and software, scheduling (Crawford and Baker 1994), circuit synthesis and diagnosis (Larrabee 1992), and many other domains. Experience has shown that different encodings of the same problem can have vastly different computational properties. For example, in planning, "causal" encodings appear to be harder to solve than "state-based" encodings.

A challenge, therefore, is to develop a general characterization of encodings that can be *efficiently* solved. This characterization may involve, for example, understanding the relationship between the encoding and the shape of its search space. Note that the characterization cannot be as simple as stating that the search space has no local minima, because realistic problems will almost certainly have local minima. For example, the SAT encodings of blocks-world planning problems do have deep local minima, yet can be solved by local search. We need to understand why search can escape from local minima in some encodings but not in others. Perhaps one can find easily measurable statistical properties of the encoding that predict the behavior of various search algorithms on a given instance.

CHALLENGE 8: (5-10 yrs) *Characterize the computational properties of different encodings of a real-world problem domain, and/or give general principles that hold over a range of domains.*

Note that evaluation of progress on this last challenge cannot be purely empirical. There will be some subjective judgement necessary in determining whether the characterization is useful and enlightening. *Predictions* of the theory can be put to an empirical test: in the best case, the general principles would suggest new encodings which can be solved more easily than previous ones.

One problem with all of the propositional encodings for real-world problems that have been suggested in the literature is that they are extremely "brittle". If we look at formulations where models that satisfy the encoding correspond to solutions to the original problem instance, we see little relationship between models that "almost" satisfy the encoded problem and candidate solutions that "almost" satisfy the original problem instance. For example, Kautz and Selman (1992) noted that it was easy to find truth-assignments that satisfied all but a single clause of SAT encodings of blocks-world planning problems. These "near models" corresponded to making a series of random motions, and then, in the last state the blocks "magically" (violating physical constraints) arrange themselves in the correct position.

The robustness of encodings, and in particular, the robustness of local search algorithms applied to those encodings, could be improved by finding ways to more closely align the semantics of the SAT instance with the semantics of the source domain:

CHALLENGE 9: (1-2 yrs) *Find encodings of real-world domains which are robust in the sense that "near models" are actually "near solutions".*

As we discussed earlier, benchmarks have been an important driving force behind work on propositional reasoning algorithms. Ideally one would have access to a very large number of real-world problem encodings, so that one could develop solid statistical evidence of the performance of various techniques. In practice the number of real problems that can be obtained is limited. It is extremely time-consuming and knowledge-intensive work to manually create such instances, and the largest and potentially most useful examples are often part of some proprietary project, and thus cannot be shared. Hard randomly-generated problems have proven to be a good alternative for testing — so far, the algorithms that are the best on randomly generated instances are also best on structured problems. However, there is the concern that we may be reaching a point where this is no longer the case, and that of the simple random distributions now used for testing may be driving us in the wrong direction in our research (Johnson 1996). Therefore we present a final challenge, which if answered would provide vital tools for ensuring progress in this field.

CHALLENGE 10: (2-5 yrs) *Develop a generator for problem instances that have computational properties that are more similar to real-world instances.*

It is necessary to provide concrete evidence, empirical and/or theoretical, that the problem distribution so generated closely matches some set of real-world domains.

7 Conclusions

We have presented a series of technical challenge problems in the area of propositional reasoning and search. We believe that progress towards solving these problems will directly extend the usefulness of the propositional reasoning approach to problems in artificial intelligence, and computer science in general.

References

Blum, A. and Furst, M.L. (1995). Fast planning through planning graph analysis. *Proc. IJCAI-95*, Canada.

Boufkhad, Y. (1996) Aspects probabilistes et algorithmiques du probleme de satisfiabilite. Ph.D. Thesis, Univ. of Paris, 1996.

Cook, S. and Mitchell, D., Finding Hard Instances of the Satisfiability Problem: A Survey. *DIMACS Series in Discr. Math. and Theoretical Comp. Sci.*. (to appear)

Crawford, J.M. and Auton, L.D. (1993) Experimental results on the cross-over point in satisfiability problems. *AAAI-93* (1993) 21–27. (Ext. version in Artif. Intel.)

Crawford, J. and Baker, A.B. (1994). Experimental results on the application of satisfiability algorithms to scheduling problems. *Proc. AAAI-94*, Seattle, WA.

Crawford, J.M. , Ginsberg, M., Luks, E., and Roy, A. (1996). *Proc. KR-96*, Boston, MA, 148–158.

Chvatal, V. and Szemeredi, E. (1988). Many hard examples for resolution. *J. of the ACM*, 35(4) (1988) 759–208.

Davis, M. and Putnam, H. (1960) A computing procedure for quantification theory. *J. of the ACM*, 7 (1960).

de la Tour, T. and Demri, S. (1995). On the complexity fo extending ground resolution with symmetry rules. *Proc. IJCAI-95*, Montreal, Canada, 289–295.

Dubois, O., Andre, P., Boufkhad, Y., and Carlier, J. SAT versus UNSAT. *DIMACS Series in Discr. Math. and Theoretical Comp. Sci.*, Vol. 26, 1996, 415–433.

Fujita, M., Slaney, J., and Bennett, F. (1993). Automatic Generation of Results in Finite Algebra *Proc. IJCAI*, 1993.

Ginsberg, M. (1995). First Intl. Workshop on AI&OR. Timberline, Oregon, OR (1995).

Haken, A. (1985). The intractability of resolution. *Theoretical Computer Science* 39 (1985) 297–308.

Hooker, J.N. (1988). Hooker, J.N., Resolution vs. cutting plane solution of inference problems: Some computational experience. *Oper. Res. Letter*, 7(1) (1988) 1–7.

Johnson, D. (1996). Experimental Analysis of Algorithms: The Good, the Bad, and the Ugly. Invited Lecture, *AAAI-96*, Portland, OR. See also http://www.research.att.com/ dsj/papers/exper.ps.

Larrabee, T. (1992) Efficient generation of test patterns using Boolean satisfiability, *IEEE Trans. on CAD*, vol 11, 1992, pp 4-15.

Kautz, H. and Selman, B. (1992) Planning as Satisfiability. *Proceedings ECAI-92*, Vienna, Austria, 1992, 359–363.

Kautz, H. and Selman, B. (1996) Pushing the envelope: planning, propositional logic, and stochastic search. *Proc. AAAI-96*, Portland, OR, 1996.

Mazure, B. Sais, L. Gregoire E. (1996). Boosting complete techniques thanks to local search methods. *Proc. Math & AI*, 1996.

Mitchell, D., Selman, B., and Levesque, H.J. (1992) Hard and easy distributions of SAT problems. *Proc. AAAI-92*, San Jose, CA (1992) 459–465.

Papadimitriou, C.H. (1993). *Computational Complexity.* Addison Wesley, 1993.

Selman, B. , Kautz, H., and Cohen, B. (1994). Noise Strategies for Local Search. *Proc. AAAI-94*, Seattle, WA, 1994, 337–343.

Selman, B., Levesque, H.J., and Mitchell, D. (1992) A new method for solving hard satisfiability problems. *Proc. AAAI-92*, San Jose, CA, 440–446.

Trick, M. and Johnson, D. (Eds.) (1996) Cliques, Coloring and Satisfiability, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26. For benchmarks, see URL: http://dimacs.rutgers.edu/Challenges/index.html.

Williams, B.C. and Nayak, P. (1996) A model-based approach to reactive self-configuring systems. *Proceedings AAAI-96*, Portland, OR. 971–978.