SATLIB: An Online Resource for Research on SAT

Holger H. Hoos (hoos@cs.ubc.ca)
Department of Computer Science
University of British Columbia
Vancouver, Canada

Thomas Stützle (stuetzle@informatik.tu-darmstadt.de)
FG Intellektik, FB Informatik, TU Darmstadt
Darmstadt, Germany

Abstract. SATLIB is an online resource for SAT-related research established in June 1998. Its core components, a benchmark suite of SAT instances and a collection of SAT solvers, aim to facilitate empirical research on SAT by providing a uniform test-bed for SAT solvers along with freely available implementations of high-performing SAT algorithms. In this article, we give an overview of SATLIB; in particular, we describe its current set of benchmark problems. Currently, the main SATLIB web site (http://www.informatik.tu-darmstadt.de/AI/SATLIB) and its North American mirror site (http://www.cs.ubc.ca/~hoos/SATLIB) are being accessed frequently by a growing number of researchers, resulting in access rates of about 250 hits per month. To further increase the usefulness of SATLIB as a resource, we encourage all members of the community to utilise it for their SAT-related research and to improve it by submitting new benchmark problems, SAT solvers, and bibliography entries.

Keywords: SAT, Benchmark Problems, SAT Algorithms

1. Introduction

The satisfiability problem in propositional logic (SAT) is one of the most prominent problems in artificial intelligence, logic, theoretical computer science, and various application areas. In the past, a large body of research has focussed on SAT and recently, the interest in SAT seems to be increasing as witnessed by a growing number of SAT-related papers published in journals and presented at major conferences. At the same time, in particular in AI, empirical studies of SAT problems and algorithms have become increasingly popular and lead to important insights into the nature of the problem as well as to dramatic improvements in SAT algorithms which, to date, can solve hard instances with thousands of variables.

Considering the increasing interest in SAT, in 1998 we created SATLIB as an online repository of benchmark problems and solvers for SAT. The original motivation behind SATLIB was to facilitate and encourage empirical studies of SAT algorithms that make use of a common set of benchmark instances and SAT solver implementations in order to improve the assessment of new algorithmic ideas or classes of SAT instances and to enhance the comparability of published empirical results. The advantages of widely used collections of benchmark problems and reference implementations of algorithms have been recognised in many areas of computer science and related fields; consequently, various such collections have been established and made available online. Some well known examples are TSPLIB (containing a variety of TSP and TSP-related instances), ORLIB (providing test instances for a variety of problems from Operations Research), TPTP (a collection of problem instances for theorem provers), and, more recently, CSPLIB (a benchmark library for constraints). For SAT, two collections of benchmark instances have been existing prior to SATLIB: the benchmark collection from the Second DIMACS Challenge [16], and the collection from the Beijing SAT Competition (see http://www.cirl.uoregon.edu/crawford/beijing/). These benchmark sets are available online, but they are static in the sense that no new problems were added after the initial realse, and hence, when SATLIB was created, they were partly out-dated.

From the beginning, the concept of SATLIB was to provide not just a benchmark library for SAT, but rather an increasingly comprehensive resource for SAT-related research including implementations, results from the evaluation of SAT algorithms, a list of people involved in research on SAT, pointers to SAT-related web sites, events, and an annotated bibliography. SATLIB is meant to be an open resource and will be continuously extended in the future.

In the following sections we give an overview of the current release of SATLIB. In particular, we motivate and describe our selection of benchmark instances, outline the SATLIB solver collection, and state the submission guidelines for benchmark instances, solvers, and bibliography entries. We also discuss other currently available components of SATLIB, such as the annotated bibliography, and finish with some concluding remarks and future plans.

2. The SATLIB Benchmark Suite

The most obvious but also the most important function of a benchmark library is to facilitate the use of the same set of problem instances across different studies and thus to enhance the comparability of the respective results. Therefore, these problem instances have to be easily accessible and usable for experimental studies. Different from the situation for CSP, there exists a widely used format for SAT instances: the cnf format from the Second DIMACS Benchmark Challenge. This format is currently accepted by almost all of the best-performing SAT solvers and tools, it is simple to parse and generate, reasonably concise and flexible, portable across different platforms, and human-readable. Therefore, all SATLIB benchmark instances are currently offered in cnf format.

Generally, benchmark sets should contain a large variety of different types of problem instances, such that they can be used as a basis for evaluating different types of algorithms in as unbiased as possible a way. Furthermore, different types of studies will focus on problem instances with different properties, and a benchmark set becomes more useful if it can support a broader range of studies. SATLIB offers four different types of problems: randomly generated native SAT instances, SAT-encoded, randomly generated problem instances from other domains, instances from direct applications of SAT, and SAT-encoded instances from other application domains. For most of these problem types, there are instances of different sizes. For randomly generated instances, such as Random-3-SAT, SATLIB provides standardised test-sets sampled from the underlying distributions. Since it is known that for these distributions, instance hardness may often vary strongly between the sampled instances [12, 15], making standardised test-sets available rather than only providing the generators offers considerable advantages for the comparability and reproducibility of empirical results. For encoded instances, we avoid encodings which increase the difficulty of solving the instances significantly.

Generally, we focus on problem instances which are intrinsically hard or difficult to solve for a broad range of algorithms and avoid instances which are known to be trivially solvable. While easy instances can be sometimes useful for illustrating or investigating properties of specific algorithms (for example polynomially solvable instances which are hard for certain, otherwise high-performing algorithms), we believe that they should not be used as general benchmark problems, as this can easily lead to heavily biased evaluations and assessments of the usefulness of specific algorithms. Hence, SATLIB's benchmark collection comprises mostly instances which are known to be hard for a wide range of SAT algorithms.

To avoid some well-known pitfalls of benchmarking [10, 11], benchmark libraries should generally not be static, but allow to be updated with new challenging problems. As elaborated in Section 4, SATLIB therefore encourages the submission (and retraction) of benchmark problems and thus lives of the contribution of people involved in SAT research. In the following, we give a brief overview of the benchmark suite currently available through SATLIB.

2.1. Uniform Random-3-SAT

Uniform Random-3-SAT is a family of SAT instance distributions obtained by randomly generating 3-CNF formulae in the following way: For an instance with n variables and k clauses, each of the k clauses is constructed from 3 literals which are randomly drawn from the $2n$ possible literals (the n variables and their negations) such that each possible literal is selected with the same probability of $1/2n$. Clauses are not accepted for the construction of the problem instance if they contain multiple copies of the same literal or if they are tautological (i.e., they contain a variable and its negation). Each choice of n and k thus induces a distribution of Random-3-SAT instances. Uniform Random-3-SAT is the union of these distributions over all n and k.

One particularly interesting property of uniform Random-3-SAT is the occurrence of a phase transition phenomenon, i.e., a rapid change in solubility which can be observed when systematically increasing (or decreasing) the number k of clauses for fixed problem size n [19, 25]. More precisely, for small k, almost all formulae are underconstrained and therefore satisfiable; when reaching some critical $k = k^*$, the probability of generating a satisfiable instance drops sharply to almost zero. Beyond k^*, almost all instances are overconstrained and thus unsatisfiable. For Random-3-SAT, this phase transition occurs approximately at $k^* = 4.26n$ for large n; for smaller n, the critical clauses/variable ratio k^*/n is slightly higher [2, 25]. Furthermore, for growing n the transition becomes increasingly sharp.

Empirical analyses show that problem instances from the phase transition region of uniform Random-3-SAT tend to be particularly hard for both systematic SAT solvers [2] and stochastic local search algorithms [28]. Striving to test their algorithms on hard problem instances, many researchers used test-sets sampled from the phase transition region of uniform Random-3-SAT (see [7, 23, 24] for some examples). Although similar phase transition phenomena have been observed for other subclasses of SAT, including uniform Random-k-SAT with $k \geq 4$, these have never gained the popularity of uniform Random-3-SAT. Maybe one of the reasons for this is the prominent role of 3-SAT as a prototypical and syntactically particularly simple $\mathcal{NP}$-complete problem.

Table I. Uniform Random-3-SAT test-sets; the `uf*` test-sets contain only satisfiable, the `uuf*` test-sets only unsatisfiable instances.

test-set	instances	clause-len	vars	clauses
uf50-218 / uuf50-218	$2 \times 1{,}000$	3	50	218
uf75-325 / uuf75-325	2×100	3	75	325
uf100-430 / uuf100-430	$2 \times 1{,}000$	3	100	430
uf125-538 / uuf125-538	2×100	3	125	538
uf150-645 / uuf150-645	2×100	3	150	645
uf175-753 / uuf175-753	2×100	3	175	753
uf200-860 / uuf200-860	2×100	3	200	860
uf225-960 / uuf225-960	2×100	3	225	960
uf250-1065 / uuf250-1065	2×100	3	250	1,065

In SATLIB we included test-sets of satisfiable and unsatisfiable instances sampled from Uniform Random-3-SAT distributions from the solubility phase transition region at ca. 4.26 clauses per variable [19, 25], where the average instance hardness for both, systematic and stochastic local search algorithms is maximal [2, 28]. Satisfiable and unsatisfiable instances were separated using fast complete SAT algorithms. This separation is particularly advantageous when using the test-sets for evaluating incomplete SAT algorithms which cannot determine unsatisfiability. Table I shows the characteristics of the test-sets currently included in SATLIB; these range from $n = 50$ to $n = 250$ variables with 100 instances per test-set, except for $n = 50$ and $n = 100$, for which test-sets with 1,000 instances each are provided.

2.2. Graph Colouring

The Graph Colouring problem (GCP) is a well-known combinatorial problem from graph theory: Given a graph $G = (V,E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E \subseteq V \times V$ is the set of edges connecting the vertices, find a colouring $C : V \mapsto \mathbb{N}$, such that neighbouring vertices have different colours. When transforming GCP into SAT, a decision variant is encoded for which the objective is to find a colouring for a given number of colours. (The optimisation variant, which searches for a colouring with a minimal number of colours, can be solved by solving a series of such decision problems.)

SATLIB contains seven test-sets based on 3-colourable flat random graphs with 50 to 200 vertices which were created using Joe Culberson's random graph generator (available from http://web.cs.ualberta.ca/~joe/Coloring, Joe Culbersons's Graph Coloring Page.). The connectivity (edges per vertex) of these graphs was adjusted in such a way that the instances have maximal hardness (on average) for graph colouring algorithms like the Brelaz heuristic [9]. Each test-set contains 100 instances except for the 50 vertex test-set, which comprises 1,000 instances. The GCP instances were transformed into SAT by using a straightforward, yet efficient encoding known from the literature [4]. The characteristic of the SATLIB test-sets thus obtained are shown in Table II.

Table II. SAT-encoded Graph Colouring test-sets (flat random graphs).

test-set	instances	vertices	edges	colours	vars	clauses
flat50-115	1,000	50	115	3	150	545
flat75-180	100	75	180	3	225	840
flat100-239	100	100	239	3	300	1,117
flat125-301	100	125	301	3	375	1,403
flat150-360	100	150	360	3	450	1,680
flat175-417	100	175	417	3	525	1,951
flat200-479	100	200	479	3	600	2,237

Another interesting class of Graph Colouring Problems is obtained by morphing regular ring lattices with random graphs [6]: A (type B) p-morph of two graphs $A = (V,E_1)$ and $B = (V,E_2)$ is a graph $C = (V,E)$ where E contains all the edges common to A and B, a fraction p of the edges from $E_1 \setminus E_2$ (the remaining edges of A), and a fraction $1 - p$ of the edges from $E_2 \setminus E_1$. The problems considered here are obtained by morphing regular ring lattices, where the vertices are ordered cyclically and each vertex is connected to its k closest vertices in this ordering, and random graphs from the well-known class G_{nm}. The morphing ratio p controls the amount of structure in the problem instances, and by varying p, the behaviour of various algorithms depending on the degree of structure and randomness can be studied [6].

SATLIB contains 9 test-sets `sw100-8-lp`x`-c5`, where $x \in \{0,1,\dots\}$ indicates a morphing ratio of 2^{-x}. Each of these test-sets contains 100 instances which were generated using a generator program provided by Toby Walsh and then encoded into SAT using the same encoding as the graph colouring instances described above [4]. The underlying graphs have 100 vertices and 400 edges, the regular ring lattice used for morphing connects each vertex to its 8 nearest neighbours in the cyclic ordering. For $0 < p < 1$, the chromatic number of the graphs thus obtained varies. Using a special graph colouring program provided by Joe Culberson, we filtered out all instances with $c \neq 5$ colours (the regular ring lattice, i.e., the morphed graph for $p = 0$, has chromatic number 5). One additional test-set, `sw100-8-p0-c5` contains the unique problem instance corresponding to the ring lattice graph (morphing ratio $p = 0$). All instances in these test-sets are satisfiable and contain 500 variable and 3,100 clauses each. [6] show that for small p (around 0.01), these instances are hardest for local search algorithms, such as WalkSAT [26], while for higher p (around 0.2), some of the instances are extremely hard for systematic search algorithms, such as SATZ [20].

2.3. PLANNING INSTANCES

Recently it has been shown that some AI Planning problems can be efficiently solved by encoding them into SAT and then finding models of the SAT formulae using standard SAT algorithms. This approach has shown to be competitive with or even to outperform state-of-the-art general-purpose planning algorithms [18]. In SATLIB we included SAT encodings from two well-known planning domains, the Blocks World domain and the Logistics domain. In Blocks World Planning, starting from some initial configuration, a number of blocks has to be moved to reach some given goal situation; in Logistics Planning, packages have to be moved between locations in different cities using trucks and airplanes with limited capacity.

Our benchmark set contains the four largest Blocks World Planning instances and four Logistics Planning instances from Henry Kautz's and Bart Selman's SATPLAN distribution. These instances are described in Table III; despite the reductions mentioned above, they are still very large when compared to other instances of our benchmark suite.

The SAT encoding used for generating the benchmark instances relies critically on techniques for reducing the size of the CNF formulae. These concern the particular way of defining the propositional variables as well as the application of well-known propositional reduction strategies, like unit propagation and subsumption, which are used to simplify the formulae before applying stochastic local search. These reductions can be computed in polynomial time and eliminate a number of propositional variables whereby the search space is efficiently reduced. Details on the SAT encoding used to generate the benchmark instances can be found in [18, 17].

Table III. SAT-encoded Blocks World Planning and Logistics Planning instances.

instance	packages / blocks	plan steps	vars	clauses
logistics.a	8	11	828	6,718
logistics.b	5	13	843	7,301
logistics.c	7	13	1,141	10,719
logistics.d	9	14	4,713	21,991
bw_large.a	9	6	459	4,675
bw_large.b	11	9	1,087	13,772
bw_large.c	15	14	3,016	50,457
bw_large.d	19	18	6,325	131,973

2.4. Instances from older benchmark sets

Finally, SATLIB also contains the instances from the benchmark set established during the Second DIMACS Challenge [16] and those of the Beijing SAT Competition. These benchmark sets contain satisfiable as well as unsatisfiable instances of widely different size and hardness. Most of them are SAT-encoded instances from other problem domains, including graph colouring, boolean function learning, test pattern generation for VLSI circuits, and inductive inference; furthermore, there is also a number of instances which have been generated using various random techniques. Because of space restrictions, we cannot give detailed descriptions of all these instances here; however, such descriptions can be found in SATLIB.

It should be noted that some of the larger DIMACS and Beijing instances contain several thousands of variables and tenthousands of clauses. Yet, their size is not always indicative of their hardness. In the course of extensive evaluations of different SAT algorithms (see also Chapter **?? [add reference to other H+S chapter]**) we found that some of these benchmark instances could be solved using polynomial preprocessing techniques. Furthermore, there are instances of SAT-encoded problems which can be efficiently solved in the original problem domain, but seem to be very hard to current techniques for solving SAT problems. These aspects of instance hardness, as far as we are aware of them, are noted in the problem descriptions included in SATLIB.

3. The SATLIB Solver Collection

The SATLIB Solvers Collection contains the original distributions of some of the most powerful SAT algorithms we are currently aware of. It comprises solvers based on stochastic local search as well as systematic search algorithms, all of which are coded in C or C++ and are relatively easy to compile and run in any Unix-like environment. The solvers included in SATLIB generally accept the DIMACS cnf-format used for all the SATLIB benchmark instances. Hence, they can be easily used for the comparative analysis of newly developed SAT algorithms, for solving SAT problems of interest, or for analysing the relative advantages and disadvantages of the different implementations.

The solvers based on stochastic local search comprise the two probably best known families of local search algorithms for SAT: GSAT [27] and the more recent WalkSAT [26]. The implementations available from SATLIB cover the basic algorithms as well as numerous variants, such as GWSAT [26], GSAT/TABU, HSAT [7], WalkSAT/TABU, Novelty, R-Novelty [24], Novelty$^+$ and R-Novelty$^+$ [13]. We refer to Chapter **?? [add reference to other H+S chapter]** for a detailed investigation into the performance of stochastic local search algorithms for SAT.

The systematic solvers, all of which are variants of the well known Davis-Logemann-Loveland procedure [3], fall into different categories. Some of the underlying algorithms, such as POSIT [5], NTAB [2], and SATZ [20], rely mainly on powerful branching rules to direct the search. Others, including GRASP [22], REL_SAT [1], and SATO [29], use various mechanisms, such as dynamic-backtracking techniques, to avoid the disadvantages of chronological backtracking. Recently, some high-performing systematic solvers have been randomised to further improve their performance [8]; SATZ-rand and REL_SAT-rand, which are variants of the basic implementations of these algorithms, fall into this category.

Note that many of the techniques these algorithms are based on, can be combined in various ways. Some of the solvers currently available realise such combinations while others are currently being extended to incorporate techniques initially used only by certain other solvers. Generally, over the past few years, the development of SAT algorithms has progressed rapidly, and we expect that new developments will lead to even more powerful solvers in the near future.

4. Submission Guidelines

To increase the usefulness and scope of SATLIB, we depend on contributions from the members of the research community. These submissions will help to further extend the SATLIB benchmark suite with challenging new instances (in particular, SAT-encoded instances from other application) and the solvers collection by new high-performing algorithms. To facilitate the use of SATLIB for testing and evaluating SAT algorithms and to ensure a uniform format and appearance, we ask contributors to follow some simple guidelines:

Benchmark Problems
All problem instances should be encoded in DIMACS cnf format. The description of this format and examples are available from the SATLIB Benchmark Section. The cnf files should be packaged into `*.tar.gz` or `*.zip` files. If you think it makes sense to divide the set of instances you are submitting into several packages which should be distributed separately, you may bundle them as you see fit.

You should provide information on the instances, at least number of variables, clauses, some background on the problem class, and give evidence for the potential interest of the instance for the SAT community (such as: they are hard for a specific class of solvers, they stem from interesting applications of SAT, ...). For examples, see the descriptions in the Benchmarks Section of SATLIB. The description you send should ideally be in the form of a HTML document, but can also be a Postscript or PDF file. Please do not send research papers instead of the problem description, unless the description of the problem class the benchmark problems are taken from is the main topic of the paper.

Solvers
All solvers should accept DIMACS cnf format (see above) or be bundled with a wrapper script (preferably perl) which translates cnf into the input format accepted by the solver. All sources and a makefile for at least Unix/Linux environments should be bundled into a `*.tar.gz` (or `*.zip`) file.

You should provide information on the solver (complete/incomplete, algorithm sketch, ...), possibly a short description in form of an HTML document. You may also send a research paper (or references to papers) if the solver is the main topic of the paper. Ideally, you should also submit evaluation data for your solver applied to SATLIB benchmark problems.

References for the Annotated Bibliography
Please send us the entries either as plain text or in BibTeX format. Submitted entries should contain all bibliographically relevant information. Additionally, you should include a URL if the paper is available online, as well as a short list of keywords and/or concise notes on its contents.

General Instructions
Please send your submissions as email attachments to `hoos@cs.ubc.ca`. When your submission is included in SATLIB, you will be automatically added to the “People Involved in SAT Research” section of SATLIB. If you want to have a link to your home page added, please send us the URL, otherwise, we will link to your email address.

Please be patient. We will try to include your submission into SATLIB as soon as possible, however, we usually will collect a number of update requests and submissions before releasing a new version of SATLIB.

5. Other SATLIB Components

While the Benchmark Suite and the Solver Collection are SATLIB's core components, SATLIB also provides a number of additional features which intend to support and stimulate SAT-related research. A list of people involved in research on SAT gives easy access to the homepages (or, if not available, to the email addresses) of active members of the SAT community. A list of SAT-related events and activities provides an overview over conferences, workshops, or journal special issues. Finally, SATLIB contains a collection of links to related sites, such as webpages on specific SAT or SAT-related topics or other benchmark libraries.

Only recently, with support from Toby Walsh and Ian Gent, a first version of an annotated bibliography on SAT has been added to SATLIB. The individual entries contain bibliographic data as well as URLs for electronic versions, some keywords and/or notes on the contents for most papers. Currently, this bibliography is still very incomplete. But we hope that with the support of the community, this will become a very useful resource, which will facilitate the access to an extensive collection of publications on SAT and may thus help in finding and locating relevant research papers as well as in providing a first orientation to new members of the SAT community.

6. Conclusions and Future Developments

We described SATLIB, an online resource for SAT-related research first established in 1998. SATLIB comprises a benchmark library as well as source code distributions for a number of the most popular and best-performing SAT algorithms known to date. These core components are aimed at facilitating empirical studies of SAT algorithms, which in the past have been instrumental in improving the performance of SAT solvers as well as in enhancing our understanding of their behaviour. SATLIB also contains various other components, including an annotated bibliography of publications on SAT, which are provided to support and stimulate SAT-related research activities.

Future releases of SATLIB will hopefully provide an extended benchmark collection and new SAT solvers, as well as a considerably extended version of the annotated SAT bibliography. In this context, we have to rely on the support from the SAT community — the continuing usefulness of SATLIB as a resource crucially depends on submissions and feedback from fellow researchers. Furthermore, we plan to extend SATLIB in various ways. Firstly, we intend to provide more tools, such as implementations of converters between different formats of SAT problems, implementations of polynomial simplification strategies for SAT instances, or tools for the empirical evaluation of the performance of SAT solvers. Secondly, we want to provide results on the performance of various SAT solvers, in particular the ones available through SATLIB, on the SATLIB Benchmark Suite. Finally, we consider to extend SATLIB to problems tightly related to SAT, such as MAX-SAT, QSAT, or the recently introduced stochastic SAT and dynamic SAT problems [21, 14].

Generally, we hope that SATLIB will continue to be a useful resource which, utilised and supported by the SAT community, may contribute to support and advance SAT-related research.

Acknowledgements

We thank all members of the community who contributed to SATLIB. In particular we would like to thank the following people for submitting benchmark problems, solvers, references, or providing some help in any other way: Roberto Bayardo, Wolfgang Bibel, James Crawford, Joe Culberson, Jon W. Freeman, Ian Gent, Carla Gomes, Jens Gottlieb, Henry Kautz, Chu-Min Li, Hans van Maaren, João P. Marques da Silva, Jussi Rintanen, Bart Selman, Geoff Sutcliff, Michael Trick, Toby Walsh, Joost Warners, and Hantao Zhang. Furthermore, we gratefully acknowledge the support of the Intellectics Group at the Computer Science Department of Darmstadt University of Technology (Germany), and the Laboratory for Computational Intelligence at the Computer Science Department of the University of British Columbia (Canada), who currently host the SATLIB web site. This work was in part supported by a Postdoctoral Fellowship awarded by the University of British Columbia to Holger H. Hoos and by a Marie Curie Fellowship awarded to Thomas Stützle (CEC-TMR Contract No. ERB4001GT973400).

References

1. R.J. Bayardo Jr. and R.C. Schrag. Using CSP Look-back Techniques to Solve Real World SAT Instances. In *Proceedings of AAAI'97*, pages 203–208. MIT Press, 1997.
2. J.M. Crawford and L.D. Auton. Experimental Results on the Crossover Point in Random 3SAT. *Artificial Intelligence*, 81(1–2):31–57, 1996.
3. M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem Proofing. *Communications of the ACM*, 5:394–397, 1962.
4. J. de Kleer. A Comparison of ATMS and CSP Techniques. In *Proceedings of IJCAI'89*, pages 290–296. Morgan Kaufmann Publishers, 1989.
5. J.W. Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1995.
6. I.P. Gent, H.H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proceedings of AAAI'99*, pages 654–660. MIT Press, 1999.

7. I.P. Gent and T. Walsh. Towards an Understanding of Hill–Climbing Procedures for SAT. In *Proceedings of AAAI'93*, pages 28–33. MIT Press, 1993.
8. C.P. Gomes, B. Selman, and H. Kautz. Boosting Combinatorial Search Trough Randomization. In *Proceedings of AAAI'98*, pages 431–437. MIT Press, 1998.
9. T. Hogg. Refining the Phase Transition in Combinatorial Search. *Artificial Intelligence*, 81:127–154, 1996.
10. J.N. Hooker. Needed: An Empirical Science of Algorithms. *Operations Research*, 42(2):201–212, 1994.
11. J.N. Hooker. Testing Heuristics: We Have It All Wrong. *Journal of Heuristics*, pages 33–42, 1996.
12. H.H. Hoos. *Stochastic Local Search — Methods, Models, Applications*. infix-Verlag, Sankt Augustin, Germany, 1999.
13. H.H. Hoos. *On the Run-time Behaviour of Stochastic Local Search Algorithms for SAT*. In *Proceedings of AAAI'99*, pages 661–666. MIT Press, 1999.
14. H.H. Hoos and K. O'Neill. Stochastic Local Search Methods for Dynamic SAT — an Initial Investigation. Technical Report TR-00-01, Computer Science Department, University of British Columbia, 2000.
15. H.H. Hoos and T. Stützle. Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. *Artificial Intelligence*, 112(1–2):213–232, 1999.
16. D.S. Johnson and M.A. Trick, editors. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.
17. H. Kautz, D. McAllester, and B. Selman. Encoding Plans in Propositional Logic. In *Proceedings of KR'96*, pages 374–384, 1996.
18. H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proceedings of AAAI'96*, volume 2, pages 1194–1201. MIT Press, 1996.
19. S. Kirkpatrick and B. Selman. Critical Behavior in the Satisfiability of Random Boolean Expressions. *Science*, 264:1297–1301, 1994.
20. C.M. Li and Anbulagan. Look-Ahead Versus Lock-Back for Satisfiability Problems. In *Proceedings of CP'97*, pages 341–355. Springer Verlag, 1997.
21. M. Littman Initial Experiments in Stochastic Satisfiablility. In *Proceedings of AAAI'99*, pages 667–672. MIT Press, 1999.
22. J.P. Marques-Silva and K.A. Sakallah. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.
23. B. Mazure, L. Sais, and É. Grégoire. Tabu Search for SAT. In *Proceedings of AAAI'97*, pages 281–285, 1997.
24. D. McAllester, B. Selman, and H. Kautz. Evidence for Invariants in Local Search. In *Proceedings of AAAI'97*, pages 321–326. MIT Press, 1997.
25. D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proceedings of AAAI'92*, pages 459–465. MIT Press, 1992.
26. B. Selman, Henry A. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In *Proceedings of AAAI'94*, pages 337–343. MIT Press, 1994.
27. B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of AAAI'92*, pages 440–446. MIT Press, 1992.
28. T. Yokoo. Why Adding More Constraints Makes a Problem Easier for Hill-Climbing Algorithms: Analyzing Landscapes of CSPs. In *Proceedings of CP'97*, pages 357–370. Springer Verlag, 1997.
29. H. Zhang. SATO: An Efficient Propositional Prover. In *Proceedings of CADE-97*, pages 308–312. Vol 1104 in LNAI, Springer Verlag, 1997.