Ensemble-based prediction of SAT search behaviour

Breanndán Ó Nualláin Maarten de Rijke [1]
Johan van Benthem

Institute for Language, Logic and Computation
University of Amsterdam
Nieuwe Achtergracht 166
1018 WV Amsterdam
The Netherlands
Email: {bon,mdr,johan}@illc.uva.nl

Abstract

Before attempting to solve an instance of the satisfiability problem, what can we ascertain about the instance at hand and how can we put that information to use when selecting and tuning a SAT algorithm to solve the instance? We argue for an ensemble-based approach and describe an illustrative example of how such a methodology can be applied to determine optimal restart cutoff points for systematic, backtracking search procedures for SAT. We discuss the methodology and indicate how it can be applied to evaluate such strategies as restarts, algorithm comparison, randomization and portfolios of algorithms.

1 *A priori* predictions

Satisfiability is a computationally hard problem [3]. In the face of this intractability, when we need to solve an instance of SAT, there is no option but to set about tackling it with one of the currently available algorithms. But which one? And, once we have selected an algorithm, which parameter settings and strategies should we use?

[1] Supported by the Spinoza project 'Logic in Action' and by a grant from the Netherlands Organization for Scientific Research (NWO) under project number 365-20-005.

This line of thinking leads us to a general question. When presented with a SAT instance which we have not seen before, what *a priori* predictions can we make about it? By *a priori*, we mean before attempting to solve the instance. We allow ourselves a cursory scan of the instance, only permitting ourselves, say $O(n)$ steps.

Among the properties which it would be useful to make predictions about are:

Cost How hard is the instance? Can we predict with some degree of confidence whether we can expect to solve the instance within 5 minutes or 5 days or 5 years? The size of the instance is the best *prima facie* indicator of computational cost.

Solubility Do we expect the instance to be satisfiable or not? The existence of distinct satisfiable and unsatisfiable phases for the 3SAT ensemble of problems can give a indication of the probability that an instance is satisfiable based on the ratio of clauses to variables [2,25,27].

Strong heuristics versus randomization Are strong heuristics (for example, branching rules [17]) likely to lead to a quicker solution? If so, which heuristics? Or would the introduction of randomization be beneficial? [11,12] And, if so, how much randomization is appropriate?

Restart susceptibility For some instances, search cost can vary enormously from run to run [8]. Sometimes, during a search attempt, it may be better to cut one's losses, give up and start again [11,21,29]. Many such restart strategies are possible. Is the instance likely to be susceptible to one of them? And, if so, which one?

Parallelizability To what extent should the search be parallelized? Does the benefit gained from running a number of processes in parallel balance the overhead of coordinating those processes? This issue is closely related to that of restart strategies [14] and similar analysis can be applied.

Algorithm suitability Where more than one SAT algorithm is available, which one is most likely to decide the instance most quickly?

Resource budgets When the available time, number of processors, and/or storage is limited, how can the available resources best be allocated?

Portfolios Is a portfolio of the algorithms more suitable and, if so, how should available resources (processor time, processors, storage, *etc.*) be allotted among the algorithms? [20,9]

When presented with a new instance, two approaches to prediction are possible. One can view the instance as an element drawn from an ensemble. Properties of the instance, such as number of clauses, number of variables, connectivity, *etc.*, can be measured. Predictions about the behaviour of the instance under search can then be made based on knowledge of the behaviour of those instances from the ensemble having those same properties.

Alternatively, one can sidestep the notion of a statistical ensemble and look for directly-measurable parameters of the instance which provide indicators of its behaviour under search.

We refer to the latter as ensemble-independent parameters while we refer to the former approach as ensemble-based prediction.

While it may seem intuitively more direct to measure ensemble-independent parameters than to bring an ensemble into consideration, we will see that difficulties measuring ensemble-independent parameters together with some statistical considerations concerning ensembles suggest that the latter may be more promising.

In the rest of the paper we look at ensemble-independent parameters and a methodology for ensemble-based prediction and compare the two. Then we show how this methodology can be applied by rigorously deriving two criteria for restart strategies. We conclude with a discussion of the methodology.

2 Ensemble-independent parameters

Kolmogorov complexity theory is an elegant and appealing approach to capturing the inherent complexity of individual objects independently of their origin [22]. Attempts have been made to relate the hardness of a SAT instance to its Kolmogorov complexity which is in turn related to its incompressibility. Such attempts are stymied by the fact that Kolmogorov complexity is not computable and only roughly approximable [22].

Vlasie [28] claimed that algorithms for hard problems behave well in general but that certain very hard instances exist. These hard instances have a very particular non-random structure which makes them compressible. However evidence seems not to be consistent with this view [26,15]. Rare, exceptionally long runs appear to be due to infelicitous choices made by the algorithm rather than being due to an inherent hardness in the instance itself. Performing many runs of a (randomized) algorithm shows that very few runs are long. Selman & Kirkpatrick [26] conclude that the practical consequence of this is that none of the instances they tested was inherently hard.

Hogg [15] considers the Entropy of an instance but dismisses it as a parameter since it applies to the process which generates instances rather than the instance itself. As a practical measure which is readily computable, he considers the Approximate Entropy. His experiments show a correlation between approximate entropy and clustering but finds no correlation between instance hardness and approximate entropy. He concludes pessimistically that these results suggest that no additional readily computable problem parameters will predict search cost or number of solutions.

3 Ensemble-based prediction

We consider the 3SAT ensemble, which has been the subject of much empirical investigation [16]. By building up a picture of the runtime behaviour of algorithms for a range of values of the parameters for 3SAT, we can predict the behaviour of a new instance from the behaviours of instances from the measured sample. More generally, since the 3SAT ensemble covers the entire 3SAT problem, we can, in principle, make predictions about any 3SAT instance.

To illustrate how ensemble-based predictions can be applied, we use Bayesian methods to make predictions for restart strategies based on these profiles. Taking two criteria for performing restarts as a point of departure, we derive formulae which can be applied to the empirically measured distributions to provide cutoff points which are optimal for the criteria.

Two criticisms have been levelled at the ensemble-based approach [15]:

(1) The variance of behaviours across an ensemble is very large, limiting the accuracy of predictions based on ensemble averages.
(2) Any given instance can be regarded as having been sampled from many different ensembles. What justification is there for predicting the behaviour of the instance based on that of instances of one of these ensembles?

In answer to the first point, we base our predictions not on averages or other descriptive statistics, but on complete runtime distributions (RTD) for the ensemble. RTDs give a fuller picture of the runtime behaviour without requiring more data to be collected that necessary for calculating the standard descriptive statistics [18]. The justification for the second point can be found in the fundamentals of the theory of statistical inference from samples. The predictions are based on the assumption that the instance has been drawn from the ensemble and, as such, will depend on the variance of the properties across the ensemble and the typicality of the instance for the ensemble. This degree of confidence can be quantified as a measure of the reliability of the prediction.

3.1 Mapping 3SAT

We are currently running a large experiment to collect runtime data for a DPLL style algorithm [7,6] running on 3SAT instances of a range of sizes containing up to 500 variables.

The purpose of the experiment is to generate a data set which can be analyzed to provide answers to many of the questions posed in the first section of this paper. In particular, the data set can be used together with the results of the following section to provide optimal cutoffs for restart strategies.

4 When to Restart

Restart techniques for CSPs have been explored by several authors. Gomes *et al.* [11], noting the significant variability of runtimes on different presentations of a single problem instance, give methods for introducing "controlled randomization" into complete, systematic, backtrack search procedures for CSPs leading to their Randomization and Rapid Restart (RRR) strategy. In [11] and [10] the behaviour of such procedures is modelled using Pareto-Lévy distributions and it is proven that these randomization methods eliminate "heavy tails" in these distributions, where such tails exist. They obtain what they call a "near-optimal" cutoff value by a trial-and-error process of varying the cutoff value.

Other approaches to finding suitable cutoffs have been tried by Kautz *et al.* [21] who used iterative deepening and by Walsh [29] who used iterative deepening along a geometric progression for his Randomization and Geometric Restarts (RGR).

Ad hoc restart strategies have also been used for `SAT` and other CSPs by [4,1] and also for non-systematic methods such as global optimization, stochastic search, and genetic algorithms by Shonkwiler and others [19].

To the best of our knowledge, we provide here the first rigorously derived optimal cutoff points for restart criteria. Our work is similar in spirit to that of [24,23] who, we feel, underestimated the potential of such analysis. Luby *et al.* describe the existence of an optimal strategy as "an interesting theoretical observation" which is "of little value in practice because it requires for its implementation detailed information about the distribution [of runtimes]." We are more optimistic in the light of the following two observations:

(1) These techniques can be used to tune a search algorithm to run on problem instances drawn from a known ensemble. For example, a satisfiability tester may be required for an application which generates instances which are similar enough to lead to a beneficial restart strategy.
(2) If restart techniques can be shown to scale in a systematic way, a restart schedule calculated from running many small instances can be scaled to

improve the behaviour of an algorithm on larger instances.

4.1 Restart Criteria

When we speak of a *run* we will mean the execution of the search algorithm from the beginning until termination without restarts. Thus a run is a complete execution in the standard sense of the word.

The execution of a search algorithm under a restart strategy will consist of a sequence of runs of the search algorithm. Each of the runs, except the last, will have been aborted after a predetermined number of steps, the cutoff, before starting a new run.

4.2 Optimization

Let the random variable $E : \mathbb{N} \to \mathbb{R}$ be the number of steps it takes the search algorithm to halt on a single run. Let η be the probability density function associated with E.

We will consider the following events: [2]

n : the algorithm halts in precisely n steps.

N : the algorithm runs for n steps without halting.

and similarly for events $\mathsf{m}, \mathsf{M}, \mathsf{t}, \mathsf{T}$ *etc.*

The probabilities associated with these events are given by:

$$\mathrm{P}(\mathsf{n}) = \eta(n) \tag{1}$$

$$\mathrm{P}(\mathsf{N}) = \sum_{k>n} \eta(k). \tag{2}$$

We use the notation $\langle \mathsf{X} \rangle$ for the expectation of the event X. The expectation of the run time of the algorithm is given by:

$$\langle \mathsf{n} \rangle = \sum_{k} k\eta(k) \tag{3}$$

We will derive two conditions for restarting, based on two criteria. The first

[2] We write events in sans serif font to distinguish them from symbols denoting numbers and truth values.

will be based purely on the given distribution η and the comparison between continuing the current run and initiating one new run.

4.2.1 One run or two

Stated in words, the criterion for performing a restart is that the expected number of steps for starting from scratch is less than the expected number of steps remaining in the current run. After m steps have been completed, the expected *total* number of steps for the current run is given by $\langle \mathsf{n}|\mathsf{M}\rangle$. Therefore the criterion is to restart at the smallest number of steps m for which:

$$\langle \mathsf{n}|\mathsf{M}\rangle - m > \langle \mathsf{n}\rangle \tag{4}$$

This criterion compares just two runs: the current run and *one* newly initiated run.

To obtain an expression for $\langle \mathsf{n}|\mathsf{M}\rangle$, we apply Bayes' theorem:[3]

$$\begin{aligned} \mathrm{P}(\mathsf{n}|\mathsf{M}) &= \frac{\mathrm{P}(\mathsf{M}|\mathsf{n})\,\mathrm{P}(\mathsf{n})}{\mathrm{P}(\mathsf{M})} &(5)\\ &= \frac{[n > m]\eta(n)}{\sum_{k>m}\eta(k)} &(6)\end{aligned}$$

From this expression we can determine the expectation for the total number of steps in those runs which do not halt during the first m steps:

$$\begin{aligned} \langle \mathsf{n}|\mathsf{M}\rangle &= \sum_n n\mathrm{P}(\mathsf{n}|\mathsf{M}) &(7)\\ &= \sum_n n\frac{[n > m]\eta(n)}{\sum_{k>m}\eta(k)} &(8)\\ &= \sum_{n>m} \frac{n\eta(n)}{\sum_{k>m}\eta(k)} &(9)\\ &= \frac{\sum_{k>m} k\eta(k)}{\sum_{k>m}\eta(k)} &(10)\end{aligned}$$

Plugging these expressions into Criterion 4, we get:

$$\frac{\sum_{k>m} k\eta(k)}{\sum_{k>m}\eta(k)} - m > \sum_k k\eta(k). \tag{11}$$

[3] We use the notation that the expression $[S]$ has the value 1 if the proposition S is true and 0 otherwise. So $[n > m] = 1$ when $n > m$ and 0 otherwise. [13]

Given a runtime frequency distribution η, we should restart the algorithm after m steps if the algorithm has not yet halted and m is the smallest positive integer satisfying Criterion 11.

4.2.2 Taking account the probability of satisfiability

Given an instance of 3SAT, the ratio of clauses to variables, m/n gives a good indication of the probability that the instance is satisfiable. Evidence indicates that the transition from the satisfiable region to the unsatisfiable region is approximately given by: $m = 4.25n$ [5]. That is, for values of m, n such that $m < 4.25n$, the vast majority of instances are satisfiable while the vast majority of instances are unsatisfiable for other values of m, n.

On this basis, we can put our prior belief concerning the satisfiability of a given instance to use. A more or less precise prior can be assumed but, for convenience, let us take for our prior belief:

$$\psi = \begin{cases} 1, & m < 4.25n \\ 0, & m > 4.25n \end{cases} \tag{12}$$

Now, instead of lumping the frequencies of all runs together into one distribution, let us measure separately the runtime frequency distributions η_s and η_u of the instances that turn out to be satisfiable and unsatisfiable respectively. These distributions are quite distinct.

Let s and u be the events that an instance is satisfiable and unsatisfiable respectively. As in the previous subsection, we consider a run that has not yet halted after m steps and calculate the consequences of this for our beliefs about how many more steps the run will take, how many steps a restarted run would take and the satisfiability of the instance.

By Bayes' theorem,

$$\mathrm{P}(\mathsf{s}|\mathsf{M}) = \frac{\mathrm{P}(\mathsf{M}|\mathsf{s})\,\mathrm{P}(\mathsf{s})}{\mathrm{P}(\mathsf{M})} \tag{13}$$

Since s and u partition the space of all events, we can write $\mathrm{P}(\mathsf{M}) = \mathrm{P}(\mathsf{M}|\mathsf{s})\mathrm{P}(\mathsf{s}) + \mathrm{P}(\mathsf{M}|\mathsf{u})\mathrm{P}(\mathsf{u})$, giving

$$\mathrm{P}(\mathsf{s}|\mathsf{M}) = \frac{\mathrm{P}(\mathsf{M}|\mathsf{s})\,\mathrm{P}(\mathsf{s})}{\mathrm{P}(\mathsf{M}|\mathsf{s})\,\mathrm{P}(\mathsf{s}) + \mathrm{P}(\mathsf{M}|\mathsf{u})\,\mathrm{P}(\mathsf{u})} \tag{14}$$

$$= \left[1 + \frac{\mathrm{P}(\mathsf{u})\,\mathrm{P}(\mathsf{M}|\mathsf{u})}{\mathrm{P}(\mathsf{s})\,\mathrm{P}(\mathsf{M}|\mathsf{s})}\right]^{-1} \tag{15}$$

$$= \left[1 + \left(\frac{1}{\psi} - 1\right) \frac{\sum_{k>m} \eta_u(k)}{\sum_{k>m} \eta_s(k)}\right]^{-1} \tag{16}$$

We interpret $\mathrm{P}(\mathsf{s}|\mathsf{M})$ as the posterior belief concerning the satisfiability of the instance in the knowledge that a run has failed to halt after m steps. We write ψ_1 for $\mathrm{P}(\mathsf{s}|\mathsf{M})$. To calculate a form for Criterion 4 which takes advantage of this information we need expressions for $\mathrm{P}(\mathsf{n})$ and $\mathrm{P}(\mathsf{n}|\mathsf{M})$. These expressions are given in terms of the posterior ψ_1.

By the same reasoning as above we can write

$$\mathrm{P}(\mathsf{n}) = \mathrm{P}(\mathsf{n}|\mathsf{s})\,\mathrm{P}(\mathsf{s}) + \mathrm{P}(\mathsf{n}|\mathsf{u})\,\mathrm{P}(\mathsf{u}) \tag{17}$$

$$= \eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1) \tag{18}$$

and, by a further application of Bayes' Theorem,

$$\mathrm{P}(\mathsf{n}|\mathsf{M}) = \frac{\mathrm{P}(\mathsf{M}|\mathsf{n})\,\mathrm{P}(\mathsf{n})}{\mathrm{P}(\mathsf{M})} \tag{19}$$

$$= \frac{\mathrm{P}(\mathsf{M}|\mathsf{n})\,\mathrm{P}(\mathsf{n})}{\mathrm{P}(\mathsf{M}|\mathsf{s})\,\mathrm{P}(\mathsf{s}) + \mathrm{P}(\mathsf{M}|\mathsf{u})\,\mathrm{P}(\mathsf{u})} \tag{20}$$

$$= \frac{[n>m](\eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1))}{\sum_{k>m} \eta_s(k)\psi_1 + \sum_{k>m} \eta_u(k)(1-\psi_1)} \tag{21}$$

The corresponding expectations are given by:

$$\langle \mathsf{n} \rangle = \sum_n n\mathrm{P}(\mathsf{n}) \tag{22}$$

$$= \sum_n n(\eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1)) \tag{23}$$

$$\langle \mathsf{n}|\mathsf{M} \rangle = \sum_n n\mathrm{P}(\mathsf{n}|\mathsf{M}) \tag{24}$$

$$= \sum_n n\frac{[n>m](\eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1))}{\sum_{k>m} \eta_s(k)\psi_1 + \sum_{k>m} \eta_u(k)(1-\psi_1)} \tag{25}$$

$$= \frac{\sum_{n>m} n(\eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1))}{\sum_{k>m} \eta_s(k)\psi_1 + \sum_{k>m} \eta_u(k)(1-\psi_1)} \tag{26}$$

Plugging these expressions into Criterion 4, we get:

$$\frac{\sum_{n>m} n(\eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1))}{\sum_{k>m} \eta_s(k)\psi_1 + \sum_{k>m} \eta_u(k)(1-\psi_1)} - m > \sum_n n(\eta_s(n)\psi_1 + \eta_u(n)(1-\psi_1)) \tag{27}$$

Should this criterion be applied and the algorithm restarted, then a criterion for a further restart can be derived but this time using ψ_1 as the prior in place of ψ. By iterating, we can find a schedule of restarting points for the first, second, and subsequent restarts until termination. This allows us to, not so much start from scratch, but start again each time in the knowledge that one or more runs, of given lengths, have failed to terminate. This knowledge influences our belief that the instance is satisfiable.

For any instance of 3SAT with values of m and n, we can use Criterion 27 to calculate, in advance, an optimal sequence of restart cutoffs. We call such a sequence a restart schedule for the instance.

5 Discussion

We have considered the relative merits of using ensemble-independent parameters and ensemble-based methods for making *a priori* predictions about SAT instances. We propose a methodology for ensemble-based prediction based on Bayesian methods. We have illustrated this methodology by deriving two criteria for restart strategies for systematic, backtracking search procedures for SAT. The methodology can be applied to provide testable results for comparison of algorithms, optimization of algorithm portfolios, parallelization of algorithms, and algorithm randomization, among others.

We note that the results presented here have not been tested empirically to evaluate just how accurately ensemble-based predictions of the above properties can be made. As mentioned earlier, we are currently collecting a large data set for the 3SAT ensemble. Future work will be directed at analyzing these data to evaluate the practical usefulness of this approach.

References

[1] Roberto J. Bayardo and Robert C. Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the National Conference on Artificial Intelligence*, pages 203–208. AAAI, 1997.

[2] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings IJCAI-91*, pages 331–337, Sydney, Australia, 1991.

[3] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computation*, pages 151–158. Association for Computing Machinery, 1971.

[4] J. M. Crawford and A. B. Baker. Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1092–1097. AAAI, 1994.

[5] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-SAT. In *in [16]*, pages 31–57. Elsevier, 1996.

[6] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Comm. ACM*, 5:394–397, 1962.

[7] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, (7):201–215, 1960.

[8] Ian P. Gent and Toby Walsh. Easy problems are sometimes hard. *Artificial Intelligence*, pages 335–345, 1994.

[9] C. P. Gomes and B. Selman. Algorithm portfolio design: Theory vs. practice. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 190–197, Providence, RI, 1997.

[10] Carla Gomes and Bart Selman. On the fine structure of large search spaces. In *IEEE ICTAI'99*, 1999.

[11] Carla P. Gomes, Bart Selman, and Henry A. Kautz. Boosting combinatorial search through randomization. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98*, pages 431–437, Madison, Wisconsin, USA, 1998. AAAI Press / The MIT Press.

[12] Carla P. Gomes, Bart Selman, Ken McAloon, and Carol Tretkoff. Randomization in backtrack search : Exploiting heavy-tailed profiles for solving hard scheduling problems. In *AIPS98*, Pittsburg, Pa, USA, June 1998.

[13] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: a Foundation for Computer Science.* Addison-Wesley, 1988.

[14] T. Hogg and C. Williams. Expected gains from parallelizing constraint solving for hard problems. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 331–336, 1994.

[15] Tad Hogg. Which search problems are random? In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98*, pages 438–443, Madison, Wisconsin, USA, 1998. AAAI Press / The MIT Press.

[16] Tad Hogg, Bernardo A. Huberman, and Colin P. Williams, editors. *Special Volume on Frontiers in Problem Solving: Phase Transitions and Complexity*, volume 81(1–2) of *Artificial Intelligence.* Elsevier, March 1996.

[17] J. N. Hooker and V. Vinay. Branching rules for satisfiability. *J. Automated Reasoning*, 15:359–383, 1995.

[18] Holger H. Hoos. *Stochastic Local Search — Methods, Models, Applications.* PhD thesis, Technischen Universität Darmstadt, 1998.

[19] X. Hu, R. Shonkwiler, and M. Spruill. Random restart in global optimization, 1994.

[20] Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275:51–54, January 3 1997.

[21] Henry Kautz and Bart Selman. Unifying SAT-based and graph-based planning. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99*, pages 318–325, Stockholm, Sweden, July 1999. Morgan Kaufmann.

[22] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and its Applications.* Springer-Verlag, 2nd edition, 1997.

[23] Michael Luby and Wolfgang Ertel. Optimal parallelization of Las Vegas algorithms. In Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors, *STACS 94, 11th Annual Symposium on Theoretical Aspects of Computer Science*, volume 775, pages 463–474. Springer, February 1994.

[24] Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. *Info. Proc. Lett.*, 47(4):173–180, September 1993.

[25] David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 459–465. AAAI, 1992.

[26] Bart Selman and Scott Kirkpatrick. Critical behaviour in the computational cost of satisfiability testing. In *in [16]*, pages 273–295. Elsevier, 1996.

[27] Bart Selman, David Mitchell, and Hector J. Levesque. Generating hard satisfiability problems. In *in [16]*, pages 17–29. Elsevier, 1996. (an extension of [25]).

[28] Dan R. Vlasie. The very particular structure of the very hard instances. In *Proc. of the 13th National Conf. on Artificial Intelligence (AAAI96)*, pages 266–270, Menlo Park, Ca., 1996. AAAI Press.

[29] Toby Walsh. Search in a small world. In Thomas Dean, editor, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99*, pages 1172–1177, Stockholm, Sweden, July 1999. Morgan Kaufmann.