Random SAT Instances à la Carte [1]

Carlos ANSÓTEGUI [a], Maria Luisa BONET [b] and Jordi LEVY [b]
[a] *DIEI, UdL, Lleida, Spain. carlos@diei.udl.cat*
[b] *LSI, UPC, Barcelona, Spain. bonet@lsi.upc.edu*
[c] *IIIA, CSIC, Bellaterra, Spain. levy@iiia.csic.es*

Abstract. Many studies focus on the generation of hard SAT instances. The hardness is usually measured by the time it takes SAT solvers to solve the instances. In this preliminary study, we focus on the generation of instances that have computational properties that are more similar to real-world instances. In particular, instances with the same degree of difficulty, measured in terms of the tree-like resolution space complexity. It is known that industrial instances, even with a great number of variables, can be solved by a clever solver in a reasonable amount of time. One of the reasons may be their relatively small space complexity, compared with randomly generated instances.

We provide two generation methods of k-SAT instances, called geometrical and the geo-regular, as generalizations of the uniform and regular k-CNF generators. Both are based on the use of a geometric probability distribution to select variables. We study the phase transition phenomena and the hardness of the generated instances as a function of the number of variables and the base of the geometric distribution. We prove that, with these two parameters we can adjust the difficulty of the problems in the phase transition point. We conjecture that this will allow us to generate random instances more similar to industrial instances, of interest for testing purposes.

Keywords. Random SAT Models, Satisfiability

Introduction

SAT is a central problem in computer science. Many other problems in a wide range of areas can be solve by encoding them into boolean formulas, and then using state-of-the-art SAT solvers. The general problem is NP-complete in the worst case, and in fact a big percentage of formulas (randomly generated instances) need exponential size resolution proofs to be shown unsatisfiable [CS88,BSW01,BKPS02]. Therefore, solvers based on resolution need exponential time to decide their unsatisfiability. Nevertheless, state-of-the-art solver have been shown of practical use working with *real-world* instances. As a consequence the development of these tools has generated a lot of interest.

The celebration of SAT competitions has become an essential method to validate techniques and lead the development of new solvers. In these competitions there are three categories of benchmarks, randomly generated, crafted, and industrial instances. It

[1] Research partially supported by the research projects TIN2007-68005-C04-01/02/03 and TIN2006-15662-C02-02 funded by the CICYT. The first author was partially supported by the José Castillejo 2007 program funded by the *Ministerio de Educación y Ciencia*.

is difficult for solvers to perform well on all of them. This has lead researchers to say that randomly generated and industrial instances are of distinct nature. It has been postulated that real-world or industrial instances have a *hidden structure*. In [WGS03] it is proved that real-world formulas contain a small number of variables that, when instantiated, make the formula easy to solve. In [ABLM08] it is shown that real-world instances have a smaller tree-like resolution space complexity than randomly generated instances with the same number of variables.

The practical applicability of SAT solvers forces them to try to be good in the industrial category. However the number of benchmarks in this category is limited. Also, the instances are isolated ones, not a family of instances, one for every number of variables. And finally, they do not have a parameterized degree of difficulty. On the other hand, random formulas can be easily generated with any size, hence with the desired degree of difficulty. Moreover, they can be generated automatically on demand, what makes their use in competitions more fair, because they are not known in advance by participants. It would be interesting to be able to generate instances with the good properties of both categories.

This project was stated in "Ten Challenges in Propositional Reasoning and Search" [SKM97] and in "Ten Challenges Redux : Recent Progress in Propositional Reasoning and Search" [KS03] as the tenth challenge:

> Develop a generator for problem instances that have computational properties that are more similar to real-world instances[...] Many SAT benchmarks today are encodings of bounded-model checking verification problems. While hundreds of specific problems are available, it would be useful to be able to randomly generate similar problems by the thousands for testing purposes.

Also Rina Dechter in [Dec03] proposes the same objective.

In this paper we want to make a contribution in this direction. Since we want to generate as many instances as needed, we define generators of random formulas. There are two models of generators of random formulas, the uniform and the regular random k-CNF generators. The first one has been studied for a long time and consists in selecting uniformly and independently m clauses from the set of all clauses of size k on a given set of n variables. The second one is studied in [BDIS05] and consists in selecting uniformly one formula from the set of formulas with m clauses of size k, and n variables, where all literals have nearly the same number of occurrences, i.e. either $\lfloor \frac{k \cdot m}{2n} \rfloor$ or $\lfloor \frac{k \cdot m}{2n} \rfloor + 1$. We generalize these two models by selecting variables following a geometric distribution, in the first case, and by allowing a power decreasing number of occurrences of the variables in the second case. We also compare these frequency distribution on the variables with what it is observed in some real-world instances.

Another interesting feature of the formulas we generate with either of our models, is that they are around the phase transition point. This means that, for the set of instances with the ratio clauses/variables around the phase transition point, the number of satisfiable formulas that we generate is approximately the same as the number of unsatisfiable ones. We are interested in generating formulas with this precise ratio, because otherwise we could easily produce trivially satisfiable or unsatisfiable instances. Instead we want to obtain reasonably easy formulas, as close to industrial ones as possible, but not trivial as for instances that would have in them a small unsatisfiable core, for instance.

Finally, we would like to mention another property that our instances have. They can be easily parameterized in terms of their hardness. As a notion of hardness we use the space of tree-like resolution, a very close notion to the logarithm of the size of tree-like resolution. The idea is that we can fix in advance the hardness, and then we can obtain as many formulas as we want of that hardness by playing with the number of variables and other parameters of our models of random formula generation.

1. Description of the Models

1.1. Geometric k-CNF

The classical uniform k-CNF random model uses a uniform probability distribution to select the variables used in the clauses. In this subsection we propose a generalization of this model by using other probability distributions. These distributions must have a discrete and finite domain of size n, where n is the number of variables. Therefore, n is a parameter of the distribution, and we need in fact a family of probability distribution functions, one for each value of n. In the particular case of the uniform distribution, for every n, we have $Pr(X = i; n) = 1/n$.

Given a continuous probability distribution ϕ with domain $\big[0, 1\big]$, we can easily generate a family of probability distributions $Pr(X = i; n)$, with discrete domain $i = 0, \ldots, n-1$, as follows. We break the interval $\big[0, 1\big]$ into n pieces, obtaining the points $0/n, 1/n, \ldots, (n-1)/n$. Then $P(X = i; n)$ is defined to be $\phi(i/n)$ with the appropriated normalization. This results into:

$$P(X = i, n) = \frac{\phi(i/n)}{\sum_{j=0}^{n-1} \phi(j/n)}$$

Since $\lim_{n\to\infty} \sum_{j=0}^{n-1} \phi(j/n)\frac{1}{n} = \int_0^1 \phi(x)\, dx$, for big values of n we have:

$$P(X = i, n) = \frac{\phi(i/n)}{\sum_{j=0}^{n-1} \phi(j/n)} \approx_{n\to\infty} \frac{\phi(i/n)}{n \int_0^1 \phi(x)\, dx} = \frac{\phi(i/n)}{n}$$

Therefore, the family of distributions that we obtain by this method is *scalable*. We know that for uniform k-CNF, and for $n \to \infty$, the phase transition point is at $m/n \approx 4.25$, a constant independent of n. Experimentally, we have seen that, to obtain the same result using a different distribution, it must satisfy this scalability condition.

The geometric k-CNF model is a generalization of the uniform k-CNF model where we use the following exponential function to generate the family of probability distributions

$$\phi(x) = \frac{\ln b}{b-1}\, b^x$$

This results into the following family of discrete distributions

$$Pr(X = i; b, n) = \frac{1 - b^{1/n}}{1 - b} b^{i/n}$$

The geometric k-CNF model is a generalization of the uniform k-CNF model where we use an exponential probability distribution with base b. Geometric k-CNF formulas may be generated with the algorithm 1.

Input: n, m, k, b
Output: a k-SAT instance with n variables and m clauses
$F = \emptyset$;
for $i = 1$ **to** m **do**
  repeat
    $C_i = \square$;
    for $j = 1$ **to** k **do**
      c = rand( );
      v = 0;
      while $c > Pr(v; b, n)$**do**
        $v = v + 1$;
        $c = c - Pr(v; b, n)$;
      endwhile
      $C_i = C_i \vee (-1)^{rand(2)} \cdot v$;
    endfor
  until C_i is not a tautology or simplifiable
  $F = F \cup \{C_i\}$
endfor

Algorithm 1. Geometric k-CNF generator. Function `rand()` return a real random number uniformly distributed in [0,1), and `rand(2)` returns either 0 or 1 with probability 0.5.

1.2. Geo-regular k-CNF

In regular k-CNF formulas all literals occur nearly the same number of time, i.e. $\lfloor \frac{k \cdot m}{2n} \rfloor$ or $\lfloor \frac{k \cdot m}{2n} \rfloor + 1$ times. In geo-regular k-CNF we want them to occur a deterministic number of times, but with a frequency given by $P(X = i; n, b)$. Geo-regular k-CNF formulas are generated as follows. We construct a multiset bag with approximately $Pr(X = v; B, n) \frac{k\,m}{2}$ copies of the literals v and $\neg v$, for each variable $v = 0, \ldots, n-1$. Then, we make a random partition of bag into m subsets (clauses) of size k, such that none of these clauses is a tautology or is simplifiable. Algorithm 2 describes this procedure. Notice that, when a tautology or simplifiable clause is generated, we discard all generated clauses, not just the last one.

2. Experimental Results

In this section we present a series of experimental results on the location of the phase transition point and the hardness of unsatisfiable instances.

The phase transition point divides the space of CNF formulas into two regions: the under-constrained region below the phase transition point, and the over-constrained re-

Input: n, m, k, b
Output: a k-SAT instance with n variables and m clauses
$bag = \emptyset$;
for $v = 1$ **to** n **do**
    $bag = bag \cup \{\lfloor Pr(v; B, n)\frac{k\,m}{2}\rfloor$ copies of $v\}$;
    $bag = bag \cup \{\lfloor Pr(v; B, n)\frac{k\,m}{2}\rfloor$ copies of $\neg v\}$;
endfor
$S =$ subset of $k\,m - |bag|$ literals from $\{1, \ldots, n, \neg 1, \ldots, \neg n\}$
    maximizing $Pr(v; b, n)\frac{k\,m}{2} - \lfloor Pr(v; b, n)\frac{k\,m}{2}\rfloor$
$bag = bag \cup S$;
repeat
    $F = \emptyset$;
    for $i = 1$ **to** m **do**
        $C_i =$ random sub-multiset of k literals from bag
        $bag = bag \setminus C_i$
        $F = F \cup \{C_i\}$;
    endfor
until F does not contain tautologies or simplifiable clauses

Algorithm 2. Geo-regular k-CNF generator

gion above it. We recall two interesting properties about the phase transition point. First, around half of the random generated instances at the phase transition point are unsatisfiable, this percentage decreases across the under-constrained region and increases across the over-constrained region. The variation in the percentage of unsatisfiable instances around the phase transition point gets sharper as the number of variables increases. So, for a big number of variables, if known, the phase transition point can be used as a precise predictor of satisfiability. Second, the hardest problems empirically seem to be found near the phase transition point. Therefore, it makes sense to test candidate algorithms on these problems. However, there is not yet a method for predicting the phase transition point. It is known that for 3-SAT, for the uniform generation method, the phase transition point is located around the clause/variable ratio 4.25 and, for the regular model the ratio is around 3.55.

We measure the hardness of unsatisfiable instance as the tree-like resolution space complexity, as proposed in [ABLM08]. Contrarily to the cpu time, or number of nodes, etc. this measure is independent of the solver/machinery used in the experiments. This measure is the minimum Strahler of a tree-like resolution tree. The Strahler is defined for binary trees as follows

$$hs(\bullet) = 0$$
$$hs(f(T_1, T_2)) = \begin{cases} hs(T_1) + 1 & \text{if } hs(T_1) = hs(T_2) \\ \max\{hs(T_1), hs(T_2)\} & \text{otherwise} \end{cases}$$

where $\bullet$ is a node and T_1 and T_2 are trees.

The space complexity is hard to compute, therefore we approximate it as the upper-bound given by the Strahler of the search tree of an execution of an extended version of the SAT solver satz [LA97]. Due to the kind of lookahead performed by satz, we

consider the leaves to have an Strahler number of 2. In [ABLM08], it is said that, for small formulas, this upper bound coincides with the real space complexity.

Each data point presented at the following figures represents the results on the computation of 200 instances, generated either by the geometric or geo-regular methods. The length of the clauses, k, was set to 3. As we have said, for solving the instances we have used an extended version of the SAT solver satz [LA97]. We have run the experiments on a 1Ghz machine with 1Gbyte of RAM. All the instances took at most 3000 seconds to be solved.

2.1. The Location of the Phase Transition Point

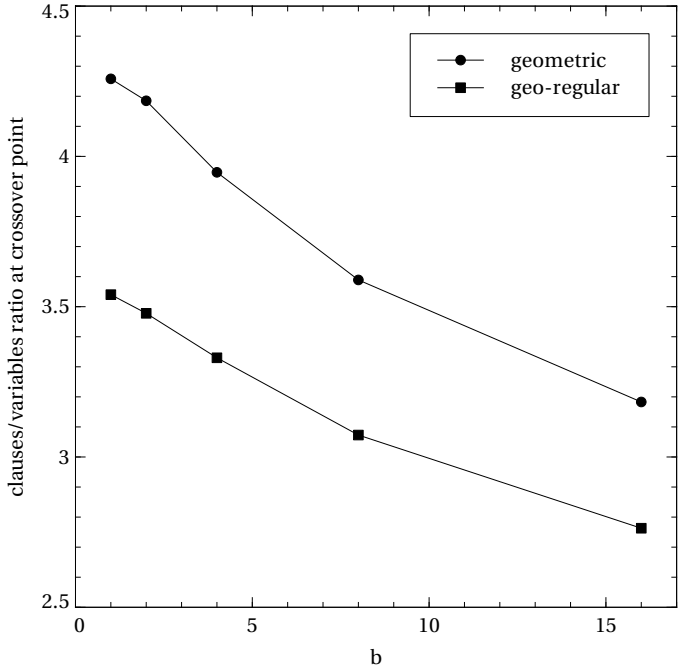

Figure 1. Clause/variable ratio at the phase transition point as a function of b.

We have also identified the existence of a phase transition point for the two proposed generation methods, geometric and geo-regular, for different values of the base $b \in \{1, 2, 4, 6, 16\}$. Figure 1, shows the clause/variable ratio at the phase transition points. Notice that for $b = 1$, the geometric and geo-regular are the uniform and regular random k-CNF generators, respectively.

For the geometric (geo-regular) method we computed the phase transition points varying the variables from 40 to 300 (140 to 420), incrementing by 20. According to the number of variables the mean execution time varies from less than one second to 3000 seconds. The phase transition point, for each number of variables, was determined by selecting the clause/variable ratio that produced a number of unsatisfiable clauses closest to 50%. That numbers varied from 48.5% to 51.5%. We think that for a larger number

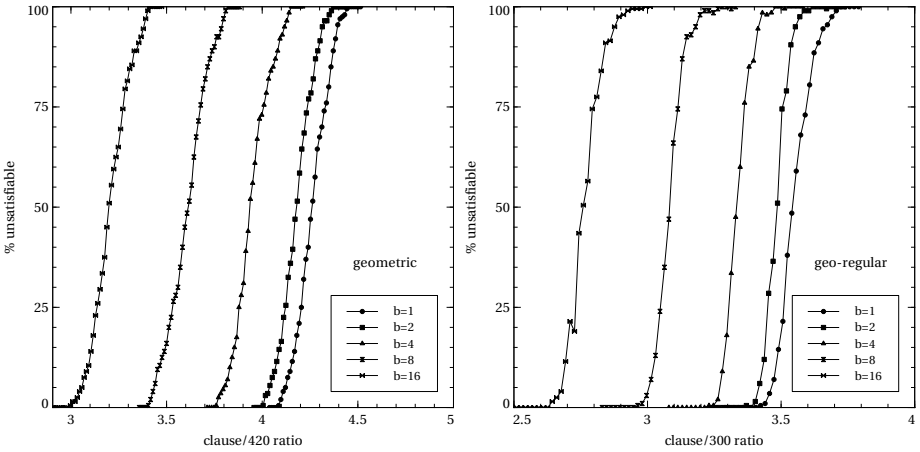

Figure 2. Percentage of unsatisfiable formulas as a function of the clause/variable ratio, with fixed number of variables.

of instances we would get number closer to 50%. Finally, in order to provide the ratio at the phase transition point, for each value of b, we computed the mean for each number of variables. As we can see in Figure 1, for both models as we increase b the clause/variable ratio decreases.

The second experiment we have conducted, shows the percentage of unsatisfiable instances as a function of the clause/variable ratio, for different values of b, and for the two models. In one case the number of variables was fixed to 420, and we varied the number of clauses by 4. In the other case we did the same with the number of variables fixed to 300. It is known that the satisfiability/unsatisfiability transition becomes sharper as we increase the number of variables. However, as we can see in Figure 2, the sharpness does not seem to depend on the value of b.

2.2. Problem Hardness at the Phase Transition Point

As reported in previous studies, the hardest instances are usually located at the phase transition region. We decided to study the hardness of the unsatisfiable instances as a function of b and the number of variables for the geometric and geo-regular methods.

There are two reasons for studying the hardness at this point. First, we plan to use our problem generators in this region, because it is where we expect to get random problems more similar to the industrial instances. These instances are (almost) minimally unsatisfiable or have very few models, and the problems generated in this region are expected to have also these properties. Second, notice that the hardness is only computed for unsatisfiable instances. As discussed in [ABLM08], there are several possible definitions for the hardness of satisfiable formulas. However, in the phase transition point, the hardness mean computed over satisfiable formulas (for a reasonable definition of hardness of satisfiable formula) and the mean computed over unsatisfiable formulas coincide.

Figure 3 shows the results. Each data point was generated as described in the previous subsection. At the first row of figure 3, we can see that for both models the hardest setting is for $b = 1$, as expected, and the strahler number grows linearly with the number

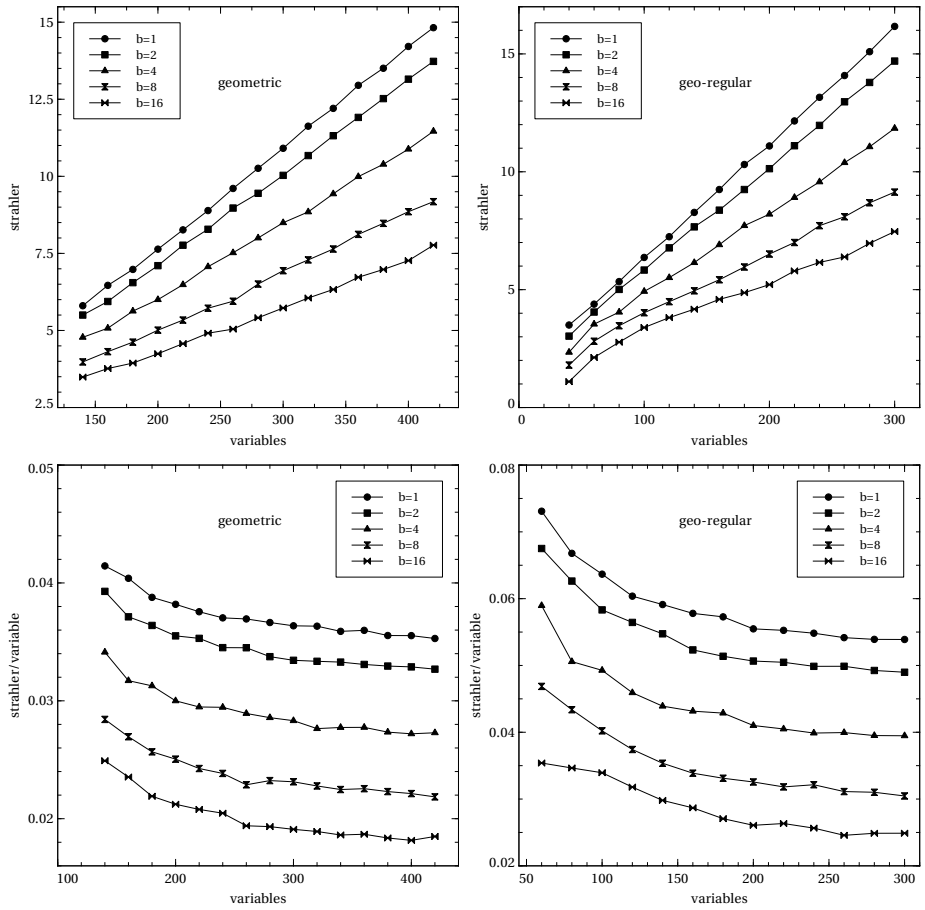

Figure 3. Strahler and strahler/variable ratio as a function of the number of variables.

of variables. For larger b's, the growth becomes smoother. Same behavior is observed for the geometric and geo-regular methods, although geo-regular clearly dominates in terms of hardness the geometric method for smaller values of b's. The second row at the figure 3 shows the strahler/variable ratio as a function of the number of variables. It is interesting to see that it seems that this ratio tends to a fixed point as we increase the number of variables. Finally, figure 4 shows this potential fixed point as a function of b.

3. Conclusions

We have proposed a generalization of the uniform and the regular k-CNF random models, by generalizing the probability distribution used on the selection of variables to a geometric distribution with base b. We have experimentally located the phase transition point, for several values of the base b. We have also studied the problem hardness as a function of the number of variables and the base.

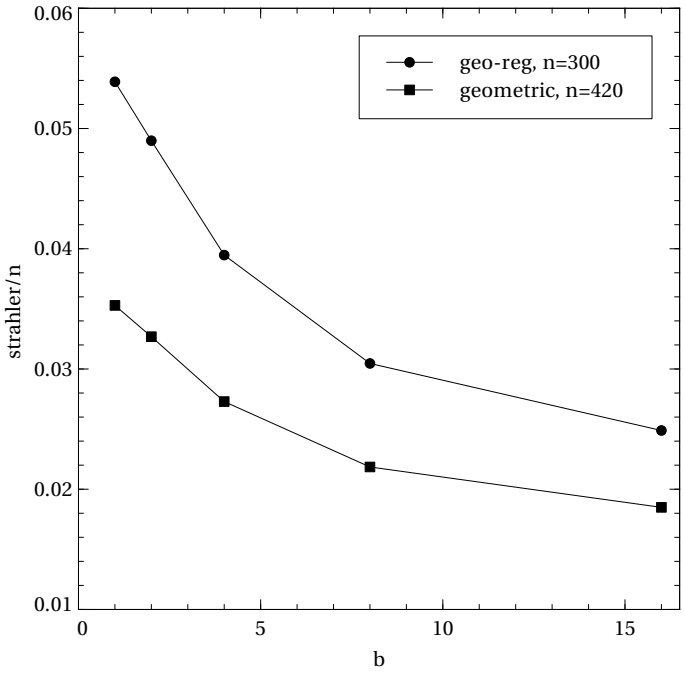

Figure 4. Strahler/variable ratio as a function of b, for big numbers of variables.

This will allow us to generate instances, at the phase transition point, of any given number of variables and *hardness* by adjusting the value of the parameter b. This is an important result since in order to do the same with the standard generators (uniform and regular random) we have to move to the under-constrained or over-constrained regions, where we find less interesting problems.

References

[ABLM08] Carlos Ansótegui, María Luisa Bonet, Jordi Levy, and Felip Manyà. Measuring the hardness of sat instances. In *Proc. of the 23th AAAI Conference on Artificial Intelligence, AAAI'08*, 2008.

[BDIS05] Yacine Boufkhad, Olivier Dubois, Yannet Interian, and Bart Selman. Regular random k-sat: Properties of balanced formulas. *J. Autom. Reasoning*, 35(1-3):181–200, 2005.

[BKPS02] Paul Beame, Richard M. Karp, Toniann Pitassi, and Michael E. Saks. The efficiency of resolution and davis–putnam procedures. *SIAM J. Comput.*, 31(4):1048–1075, 2002.

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow - resolution made simple. *J. ACM*, 48(2):149–169, 2001.

[CA96] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1-2):31–57, 1996.

[CS88] Vasek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988.

[Dec03] Rina Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[JS96] Roberto J. Bayardo Jr. and Robert Schrag. Using csp look-back techniques to solve exceptionally hard sat instances. In Eugene C. Freuder, editor, *Proc. of the Second Int. Conf. on Principles and Practice of Constraint Programming, CP'96*, volume 1118 of *Lecture Notes in Computer Science*, pages 46–60. Springer, 1996.

[KS99] Henry A. Kautz and Bart Selman. Unifying sat-based and graph-based planning. In Thomas Dean, editor, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence, IJCAI'99*, pages 318–325. Morgan Kaufmann, 1999.

[KS03] Henry A. Kautz and Bart Selman. Ten challenges redux: Recent progress in propositional reasoning and search. In Francesca Rossi, editor, *Proc. of the 9th Int. Conf. on Principles and Practice of Constraint Programming, CP 2003*, volume 2833 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2003.

[LA97] Chu Min Li and Anbulagan. Look-ahead versus look-back for satisfiability problems. In *Proc. of the 13th Int. Conf. on Principles and Practice of Constraint Programming, CP'07*, volume 4741 of *Lecture Notes in Computer Science*, pages 341–355. Springer, 1997.

[MSL92] David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and easy distributions of sat problems. In *Proc. of the 10th National Conf. on Artificial Intelligence*, pages 459–465, 1992.

[Sel00] Bart Selman. Satisfiability testing: Recent developments and challenge problems. In *Proc. of the 15th Annual IEEE Symposium on Logic in Computer Science, LICS'00*, page 178, 2000.

[SKM97] Bart Selman, Henry A. Kautz, and David A. McAllester. Ten challenges in propositional reasoning and search. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence, IJCAI'97*, pages 50–54, 1997.

[WGS03] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence, IJCAI-03*, pages 1173–1178. Morgan Kaufmann, 2003.