View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by UCL Discovery

Generation of Hard Non-Clausal Random Satisfiability Problems

Juan A. Navarro and **Andrei Voronkov**
University of Manchester
School of Computer Science
`{navarroj,voronkov}@cs.man.ac.uk`

Abstract

We present the results from experiments with a new family of random formulas for the satisfiability problem. Our proposal is a generalization of the random k-SAT model that introduces non-clausal formulas and exhibits interesting features such as (experimentally observed) sharp phase transition and the easy-hard-easy pattern. The experimental results provide some insights on how the use of different clausal translations can affect the performance of satisfiability solving algorithms. We also expect our model to provide diverse and challenging benchmarks for developers of SAT procedures for non-clausal formulas.

Introduction

The problem of propositional satisfi ability (SAT), that is to decide whether there is a satisfying truth assignment for a given propositional formula, is very interesting both from theoretical and practical viewpoints. As the prime NP-complete problem it plays a fundamental role in the complexity and computation theory. The feasibility of many applications, especially in artifi cial intelligence, relies on the existence of effi cient procedures for solving this problem.

Randomly generated formulas have often been used as benchmarks to evaluate the performance of satisfi ability solving procedures. It is important, as already pointed out (Mitchell, Selman, & Levesque 1992; Mitchell & Levesque 1996), to have a clear understanding of the distribution of such formulas and avoid incorrect conclusions from deceiving experimentation results. An algorithm may quickly solve several thousands of problems not because it is too clever or effective but, unfortunately, because of a bad sampling mechanism with a tendency to produce easy problems.

A model that has been recognized for being able to produce challenging benchmarks for the satisfi ability problem is random k-SAT. Formulas are produced by randomly selecting clauses of length k built from a set with a given number of variables. For one parameter of the distribution, namely the ratio between the number of clauses and the number of variables, an interesting pattern has been observed: the sets of generated formulas exhibit, at some critical region, a sharp transition between almost all being satisfi able to almost none. Moreover, problems generated near this critical region are hard to solve for all existing systems, while problems far from this region are either easy or only moderately hard. Researchers have shown a lot of interest in the study of random k-SAT and related problems (such as determining bounds for the critical region) and, at the same time, hard random 3-SAT formulas became a standard benchmark for testing satisfi ability procedures.

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

The results of the SAT Competition[1] (Berre & Simon 2004), where new and state-of-the-art solvers are tested against several benchmarks, have shown that the best solvers on random 3-SAT are not necessarily the most effective on real life applications and vice versa. One of the possible explanations is that such random formulas, which are just large sets of short and independent clauses, are unable to simulate problems with some information about structure.

In this paper, we present a generalization of the random k-SAT model that can be used to produce test formulas with non-trivial structure. Our proposed *fixed shape model*, which is based on the idea of introducing non-clausal formulas, has several interesting features. First it produces a family of instances controlled by a number of parameters, allowing to evaluate solvers under different settings including critical conditions. Examples from real life applications usually don't allow this amount of control. At the same time, our proposed model produces instances with some level of structure. This makes our model interesting to evaluate solving techniques that try to exploit such structure information. In particular we hope that our model will help to provide evaluation, benchmark and test problems for emerging non-clausal solvers that are currently under development.

The main purpose of this paper is to experimentally study the probability distribution of the formulas generated according to the proposed model. Characteristic features such as sharp phase transition and the existence of hard problems in a critical region are observed. We also perform some experiments to compare the performance of different state-of-the-art solvers in combination with two clausal form translations. We address the question of how the choice of a translation affects the properties of the generated problems and the performance of the solvers when trying to solve them. Our results point out that no translation can be found better that the other, and more research in this direction is needed.

[1]`http://www.satcompetition.org/`

Preliminaries

As usual, formulas are built from variables and propositional logic connectives. A *literal* is either a variable or its negation. Given a set of variables Σ we use Σ^+ to denote the set of literals that can be built using the variables in Σ. A *clause* is a disjunction of literals. A conjunction of clauses, sometimes represented by a set of clauses, is a formula in *conjunctive normal form* (CNF). We use the term *non-clausal* or *non-CNF* to emphasize that we are using arbitrary propositional formulas, not necessarily in conjunctive normal form.

A *truth assignment* is a function that maps the set of variables into $\{\mathbf{true}, \mathbf{false}\}$. An arbitrary formula can be evaluated under a truth assignment following the standard rules of propositional calculus. A formula is said to be *satisfiable* if there is at least one truth assignment that evaluates the formula to $\mathbf{true}$, and *unsatisfiable* otherwise. A satisfying truth assignment is sometimes called a *solution*.

A *complete* SAT procedure is one that takes as input a formula and always terminates returning either *'yes'* or *'no'* answering if the formula is satisfiable. One of the most widely used complete SAT procedures is DLL due to Davis, Logemann and Loveland (1962). The procedure performs a backtracking deep-first search in the space of truth assignments, and is usually augmented with several heuristics (unit propagation, learning). The number of *branches* or decisions that the procedure needs to perform in order to solve a problem is usually taken as an indicator of the difficulty of the problem.

The fixed clause-length model

In this section we present the fixed clause-length model, usually referred in literature as random k-SAT, that generates random CNF formulas. This model has three parameters: the number of variables n, the number of clauses m and the length k of the clauses to produce. The parameter $r = m/n$, the ratio of clauses to variables, is often used to replace m.

A formula is generated by selecting clauses uniformly at random from the set of all possible clauses of length k. Slight variations of the model can be found whether trivial clauses (with complementary or repeated literals) are allowed or not. This, however, does not seem to affect the general behavior of the distribution. Extensive research has been done on random k-SAT (Mitchell, Selman, & Levesque 1992; Mitchell & Levesque 1996; Cook & Mitchell 1997) where two main features are frequently pointed out:

Sharp Phase Transition: For each k and n, the probability of a generated formula of being satisfiable changes, as the value of r increases, from almost 1 to almost 0 in a very narrow region. Moreover, as the value of n increases, the transition seems to take place in a much narrower area around some *crossover* point r^*. Friedgut (1999) was able to show that, indeed, the size of the critical region shrinks as n increases. His theoretical result, however, does not give any clues about the value of r^*, or even if such value should actually exist. For random 3-SAT, experimental evidence suggests a value near 4.25, and bounds for the crossover region are also known: $3.52 < r^* < 4.506$ (Kaporis, Kirousis, & Lalas 2003; Dubois, Boufkhad, & Mandler 2000).

The Easy-Hard-Easy Pattern: The difficulty of the generated problems (usually measured as the number of branches explored by a DLL-based algorithm) exhibits a pattern that goes from very easy, for small values of r, to very hard, when r enters the phase transition, to easy (or moderately hard) when r becomes large. This phenomena is usually explained by the fact that, for low values of r, a formula with few clauses is under-constrained and very easy to satisfy. On the other hand, for large r, the formula is over-constrained and a complete SAT procedure can quickly find contradictions to finish the search. The hardest problems appear in the transition region where there are just enough clauses to make the problem potentially unsatisfiable, but not too many to make it easy for a solver to determine. The difficulty of a particular distribution of formulas clearly depends on the procedure used to solve it, but several authors have conjectured that this general pattern will hold for any reasonable complete method (Cook & Mitchell 1997).

The fixed shape model

Our proposed model is closely related to the fixed clause-length model introduced in the previous section. We follow the same idea to go from under- to over-constrained areas but, instead of clauses of a fixed length, we use formulas generated according to a particular fixed shape.

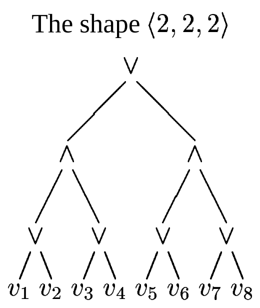
Definition 1. A *shape* is a propositional formula S such that (i) S is built using the conjunction and disjunction connectives only; and (ii) every variable appearing in S has exactly one occurrence in it. An Σ-*instance* of a shape is any formula obtained by replacing every variable in the shape by a literal from the set Σ^+. A *randomly generated* Σ-*instance* of a shape $\mathcal{S}$, is a formula obtained by independently and uniformly choosing literals from the set Σ^+ to replace each variable occurring in $\mathcal{S}$.

In the sequel we assume that Σ is clear from the context and simply speak about instances instead of Σ-instances. The formula $(v_1 \wedge v_2) \vee v_3$ is an example of a shape. Two $\{x_1, x_2, x_3, x_4\}$-instances of this shape are $(\neg x_3 \wedge x_2) \vee \neg x_1$ and $(\neg x_4 \wedge x_3) \vee x_4$. Let us introduce a special kind of shape, called *balanced conjunctive-disjunctive shapes*; informally these are balanced trees of alternating conjunctions and disjunctions.

Definition 2. Given d integers $k_1, k_2, \ldots, k_d$ (with $d \geq 0$ and $k_i \geq 2$) we define two sets of formulas $[\![k_1, k_2, \ldots, k_d]\!]$ and $\langle k_1, k_2, \ldots, k_d \rangle$ recursively as follows.

1. If $d = 0$, then the formulas in both $[\![\,]\!]$ and $\langle\,\rangle$ are literals.
2. If $d \geq 1$ then every formula in $[\![k_1, k_2, \ldots, k_d]\!]$ is a conjunction of k_1 formulas in $\langle k_2, \ldots, k_d \rangle$. Likewise, every formula in $\langle k_1, k_2, \ldots, k_d \rangle$ is a disjunction of k_1 formulas in $[\![k_2, \ldots, k_d]\!]$.

Evidently, if we have large enough set of variables Σ, then every set $\langle k_1, \ldots, k_d \rangle$ contains a shape S, moreover $\langle k_1, \ldots, k_d \rangle$ is the set of all instances of this shape (and similar for $[\![k_1, \ldots, k_d]\!]$). For this reason we will sometimes refer to $\langle k_1, \ldots, k_d \rangle$ as a *balanced disjunctive shape* and to $[\![k_1, \ldots, k_d]\!]$ as a *balanced conjunctive shape*. The value d is called the *depth* of the shape.


The shape $\langle 2,2,2\rangle$
$\vee$
$\wedge$ $\wedge$
$\vee$ $\vee$ $\vee$ $\vee$
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8

Sample instances:

$((\neg x_3 \vee \neg x_6) \wedge (\neg x_4 \vee \neg x_3))$
$\vee((\ x_9 \vee \neg x_5) \wedge (\ x_3 \vee\ \ x_7))$

$((\neg x_6 \vee\ \ x_7) \wedge (\neg x_6 \vee\ \ x_2))$
$\vee((\neg x_9 \vee \neg x_7) \wedge (\neg x_1 \vee\ \ x_4))$

$((\ x_4 \vee \neg x_4) \wedge (\ x_8 \vee\ \ x_6))$
$\vee((\ x_7 \vee\ \ x_4) \wedge (\ x_4 \vee\ \ x_2))$

Figure 1: Structure of the shape $\langle 2,2,2\rangle$ together with three sample instances.

Note that the balanced shapes and their instances are formulas in negation normal form (NNF), one example is presented on Figure 1, moreover every formula in NNF is an instance of some shape. We can define then the *random* $\langle k_1,\dots,k_d\rangle$*-SAT* model as follows. The parameters are the number of variables n and a real number r, called *density*. A test formula is produced as the conjunction of $[rn]$ randomly generated $\{x_1,\dots,x_n\}$-instances of $\langle k_1,\dots,k_d\rangle$. Note that the case $\langle k\rangle$ gives us exactly the random k-SAT model. Also there is no need to consider random $[\![k_1,\dots,k_d]\!]$-SAT since this would be equivalent to simply generate $k_1[rn]$ random instances of $\langle k_2,\dots,k_d\rangle$.

Several properties of balanced shapes can be used to characterize hardness of generated random formulas.

Theorem 1. *Let t be an arbitrary but fixed truth assignment. The probability $p_{\langle k_1,\dots,k_d\rangle}$ that t satisfies a random instance of $\langle k_1,\dots,k_d\rangle$ can be calculated as follows.*

$$p_{\langle\,\rangle} = 1/2\,,$$
$$p_{\langle k_1,\dots,k_d\rangle} = 1 - (p_{\langle k_2,\dots,k_d\rangle})^{k_1}\,.$$

Proof. Note that this probability is just the number of instances of the shape that are satisfied by the fixed truth assignment divided by the total number of instances of the shape (with respect to a set Σ with a fixed number of variables). It is possible to count both things using very simple combinatorial arguments. $\square$

Intuitively shapes with a value of p very close to 0 are very hard to satisfy, so a fewer number of them are sufficient to make a randomly generated problem unsatisfiable. Conversely a value of p very close to 1 would make a random instance quite easy to satisfy, so only very large formulas could have a chance of being unsatisfiable. The later effect has been experimentally observed on random k-SAT for large values of k (Mitchell & Levesque 1996) and is confirmed by analytical lower bounds of the crossover region (Achlioptas & Peres 2003).

Theorem 2. *The probability that a random instance of* $\langle k_1,\dots,k_d\rangle$*-SAT, with n variables and density r, is satisfiable tends to 0 as $n\to\infty$ for all $r > \log 2/\log(1/p)$. With the value of p calculated as in Theorem 1.*

Proof. A fixed truth assignment t satisfies a conjunction of $[rn]$ instances of $\langle k_1,\dots,k_d\rangle$ with probability $p^{[rn]}$. The expected number of satisfying assignments is $2^n p^{[rn]}$. This value (and the probability of the instance of being satisfiable) tends to 0 as $n\to\infty$ when $r > \log 2/\log(1/p)$. $\square$

This simple argument, useful to estimate the location of the critical region, has also been used to give an easy upper bound of the random k-SAT crossover point (Cook & Mitchell 1997).

CNF Translations

While the formulas generated by our proposed model are non-CNF in essence, modern SAT solvers and procedures are designed under the assumption that their input is a CNF formula. There is a recent interest on the design of non-clausal satisfiability testing algorithms (Thiffault, Bacchus, & Walsh 2004; Giunchiglia & Sebastiani 2000; Stachniak 2002), but mature implementations are not readily available (see Related Work). In order to measure the *hardness* of our formulas we decided to translate them into CNF first and then use a standard clausal solver. This raises the important question on how the choice of a particular translation could affect the performance of existing solving procedures.

To test our formulas we used two kinds of translations. The *standard translation* (equivalence preserving) is simply based on distributive properties of disjunction and conjunction. It is well known that such translation causes an exponential increase in the size of the problem.

Theorem 3. *The standard translation of the balanced shape $\langle k_1,\dots,k_d\rangle$ produces a CNF formula with clauses of the same length. Moreover, the length is the product of all the k_i with odd i.*

Table 1 illustrates this theorem showing the clause lengths of several shapes. The second we consider is an *optimized translation* (structure preserving) that uses the so called *naming technique* to avoid an exponential size increase by introducing new variables (Plaisted & Greenbaum 1986). Let $F[G]$ be an NNF formula with a distinguished subformula G. We assume that G is not a literal, hence G occurs in F positively. We can transform $F[G]$ by (a conjunction of) two formulas $F[p]$ and $\neg p\vee G$, where p is a fresh variable. It is not hard to argue that this transformation preserves satisfiability. The optimized translation takes a formula in NNF and repeatedly applies this transformation until a formula in CNF is obtained. We did a careful implementation that does not introduce new variables unnecessarily.

The optimized translation has the main advantage of keeping the size of the translated formula small (linear with respect to the original), at the cost of introducing new variables. It was interesting to see how modern solvers cope with this increase in the number of variables and determine whether the introduced optimizations are useful or not.

Results

In order to experimentally observe the distribution of our randomly generated formulas we started by running small simulations with different shapes and parameter values on a variety of solvers. Table 1 shows properties of formulas tested at this stage. In this table, 'vars' is the number of

shape	vars	p	r_u	weight	length		fresh
$\langle 3, 2\rangle$	6	0.578	1.26	7.59	3	2.14	3
$\langle 3\rangle$	3	0.875	5.19	15.57	3	3.00	0
$\langle 2, 4, 2\rangle$	16	0.533	1.10	17.61	4	2.89	2
$\langle 6, 3\rangle$	18	0.551	1.16	20.95	6	2.21	6
$\langle 2, 2, 3, 2\rangle$	24	0.557	1.18	28.41	6	2.27	14
$\langle 4\rangle$	4	0.938	10.83	43.32	4	4.00	0
$\langle 3, 3, 2\rangle$	18	0.807	3.23	58.11	6	3.00	3
$\langle 2, 2, 4, 2\rangle$	32	0.716	2.08	66.46	8	2.32	18
$\langle 2, 5, 3\rangle$	30	0.763	2.56	76.78	6	3.82	2
$\langle 5\rangle$	5	0.969	21.83	109.16	5	5.00	0
$\langle 2, 2, 2, 2, 2\rangle$	32	0.880	5.42	173.51	8	2.95	10

Table 1: Properties of some balanced shapes

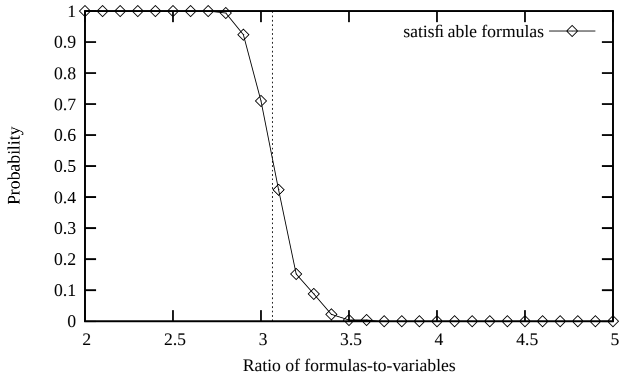


Figure 2: Probability of satisfi ability of 70-variable random $\langle 3, 3, 2\rangle$-SAT formulas.

variables in the shape and p is the probability that a random formula is satisfi ed by a truth assignment, see Theorem 1. Then r_u is an upper bound of the crossover region, see Theorem 2. The 'weight' of each shape is the product of the number of variables and r_u, it serves to compare the size of the generated formulas (measured as an the number of literals in them) in the hard region. Consider the $\langle 2, 4, 2\rangle$ shape for example, although it is bigger and more complex than a simple length 4 clause, we only need a few instances of them ($\lceil 1.1n\rceil$ instead of $\lceil 10.8n\rceil$) to produce hard formulas. Low weight shapes are interesting because they seem appropriate to generate hard and short problems. The 'length' column shows the length of the resulting clauses for the standard translation, as in Theorem 3, and the average clause length for the optimized one. Finally 'fresh' is the number of fresh variables introduced by the optimized translation for each generated instance of the shape.

Using this information we designed several experiments whose results we detail now. At this stage we considered three solvers: zChaff (2004.5.13), a carefully engineered implementation of the DLL procedure (Moskewicz *et al.* 2001); march_eq (2004.3.20, 100% lookahead), which integrates equivalence reasoning techniques (Heule *et al.* 2004); and kcnfs (2003.2.12), a solver with effi cient heuristics to solve random k-SAT formulas (Dubois & Dequen 2001). The selection of the solvers was made using the results of the SAT Competition 2004 as a reference. Moreover, we wanted to use very diverse solvers in order to observe how

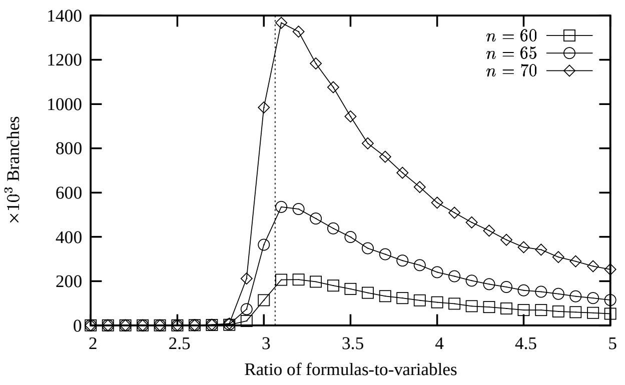


Figure 3: Number of branches required by zChaff on 60, 65, and 70-variable formulas of random $\langle 3, 3, 2\rangle$-SAT.

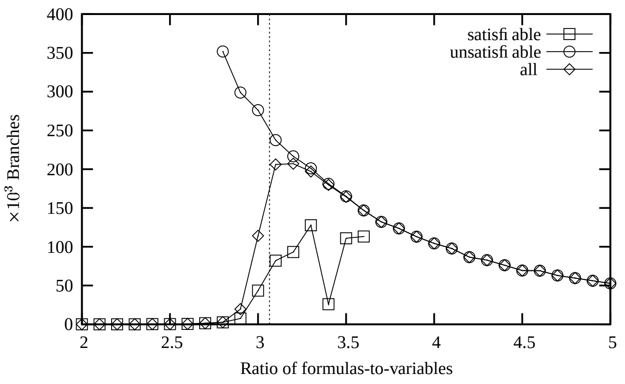


Figure 4: Number of branches of 60-variable formulas of random $\langle 3, 3, 2\rangle$-SAT, factored according to satisfi ability.

different strategies and clausal translations perform in this setting with mixed randomness and structure. The experiments were run in parallel on 45 computers, each having an Intel III 1GHz CPU and 512Mb RAM.

In a fi rst experiment we performed an analysis of random $\langle 3, 3, 2\rangle$-SAT formulas generating 500 samples for each parameter value. The purpose of this experiment was to obtain an accurate description of the probability distribution of this shape. Figure 2 shows an already familiar picture: the probability that a generated formula is satisfi able changes from almost 1 to almost 0 in a narrow region around the 0.5 probability point, in this case close to $r = 3.07$.

Figure 3 shows the median of the number of branches required by zChaff to solve these formulas. The easy-hard-easy pattern is reproduced with the hardest problems near the crossover point. The same basic pattern was found in all our experiments with different solvers and shapes. Compared to analogous results on random 3-SAT we can see that the transition from easy to hard is much more sudden (increasing from a few hundred to more than 1.3 million branches in a region of length 0.3), while the decay after leaving the critical region is gradual and slow. Figure 4, which presents these results factored into satisfi able and unsatisfi able groups, suggests that most of the satisfi able for-

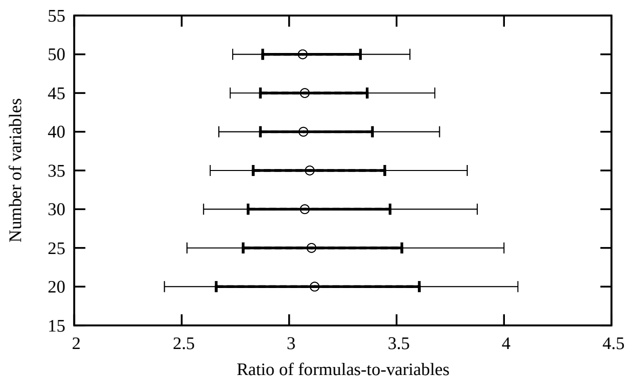


Figure 5: Scaling 0.1 and 0.01-windows for $n = 20, \ldots, 50$ on random $\langle 3, 3, 2\rangle$-SAT formulas.

Translation	clauses	length	variables
standard	2240	4	140
optimized	1260	2.89	420

Table 2: Statistics of clausal transformations for random $\langle 2, 4, 2\rangle$-SAT with 140 variables at $r = 1.0$.

mulas are rather easy to solve; while the unsatisfi able ones, several order of magnitudes harder, dominate the behavior of the curve as soon as they appear. This fi gure also shows, however, that the few satisfi able formulas to the right of the crossover point can also have a signifi cant diffi culty.

Using a more intense sampling near the critical region (1000 test cases per data point) we could observe the so called scaling window effect. Let ϵ be a real number ($0 < \epsilon < 0.5$) the ϵ*-window* is the interval of values of r where the probability of satisfi ability lies within ϵ and $1 - \epsilon$. Figure 5 shows how the length of the 0.1 and 0.01-windows (the former denoted with a thicker plot line) decreases as the value of n increases, the crossover point is also marked with a small circle. This serves to provide observable evidence that sharp phase transition can take place.

We also wanted to compare the performance of each solver with respect to the clausal form translation applied. For this fi nal test we generated a smaller set of problems (50 samples per point) with instances of random $\langle 2, 4, 2\rangle$-SAT. In this case the crossover point was found near the $r = 1.0$ sample. Table 2 shows some statistics that compare the clausal representations provided by the two translations. In Table 3 the total CPU time usage of the solvers on each translation is shown.

Figure 6 shows the relation between the two translations for several values for r. The symbol $\mathsf{branch}(x, y)$ denotes

Translation	zChaff	march_eq	kcnfs
standard	431.5 min	58.3 min	14.8 min
optimized	722.8 min	31.9 min	19.1 min

Table 3: CPU time for each solver and translation to solve $\langle 2, 4, 2\rangle$-SAT with 140 variables.

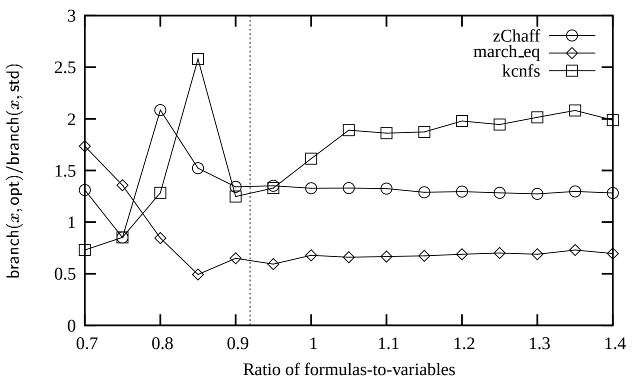


Figure 6: Effectiveness of the optimized translation for solving $\langle 2, 4, 2\rangle$-SAT with zChaff, march_eq and kcnfs.

the total number of branches used to solve 50 problems, with a fi xed value of r, for each combination of a solver x and a translation y. The proportion $\mathsf{branch}(x, \mathsf{opt})/\mathsf{branch}(x, \mathsf{std})$ helps to provide a fair comparison indicating how the use of the optimized instead of the standard translation improved (< 1) or deteriorated (> 1) the performance of the solver. It is quite surprising that no translation can be found better than the other. The solvers zChaff and kcnfs showed a better performace with the standard translation and, conversely, march_eq found more useful the optimized one. We could suspect that, since march_eq incorporates equivalence reasoning, the use of the optimized translation helps the solver to fi gure out the structure of the problem. While, for the other two solvers, the introduction of new variables by the optimized translation has the undesirable effect of increasing the total space searched. We performed similar experiments with other shapes and parameter values. Due to lack of space it is not possible to include more details, but the general observations already discussed were also found in the other experiments.

Related Work

The study and development of non-clausal procedures for the satisfi ability problem is quite recent. Some authors have initiated a search of tractable classes of non-clausal problems (Altamirano & Escalada-Imaz 2000), while others look for possible ways to generalize the DLL method (Thiffault, Bacchus, & Walsh 2004; Giunchiglia & Sebastiani 2000; Stachniak 2002). It would be very interesting to test our formulas with the system NoClause described in (Thiffault, Bacchus, & Walsh 2004), however we encountered some portability issues with the current version that prevented us from doing some experimentation. In the work of Stachniak (2002) a fi rst attempt to build hard non-clausal formulas is made, they are instances of $[\![2, 2, [rn], 3]\!]$, however no evidence of sharp phase or diffi culty patterns were reported.

Although most of the research on randomly generated SAT problems is focused on the random k-SAT model, other variants can also be found in literature. Monasson and Zecchina (1997) proposed a random $(2 + p)$-SAT model that, based on insights from statistical mechanics, mixes 2- and 3-SAT clauses. Other variable length models have also been considered (constant probability, or expected length) but they were found unsuitable for the production of hard problems (Mitchell, Selman, & Levesque 1992; Mitchell & Levesque 1996).

Generation of structured hard instances for the SAT problem has been usually done by translating problems from other domains (graphs, combinatorics, optimization, ...) into propositional boolean formulas. Other generators, such as XOR-SAT (Barthel *et al.* 2002), are particularly designed to produce only satisfi able instances. There has also been a lot of interest in the more general setting of random constraint satisfaction problems (Gent *et al.* 2001), and generalized satisfi ability problems (Creignou & Daude 2002).

Conclusions

Extensive research on the satisfi ability problem has derived in a deeper understanding on this and many other important problems in AI and related fi elds. Very effi cient implementations of solving procedures are now easily available, and the performance of state-of-the-art solvers keeps improving each year. We believe that, in the near future, the research in this area will focus on the development of new theories and procedures that, extending current known approaches, will be able to handle general classes of formulas that encode information in a more succinct and effi cient way.

In this paper we have presented a model that allows to randomly generate hard non-clausal formulas. We expect this procedure to provide diverse and challenging material to evaluate the performance of current and next generation solvers that have started to introduce non-clausal features. We have carried out a careful experimental observation of the properties exhibited by these formula distributions where the familiar sharp phase phenomenon and easy-hard-easy patterns were found.

A second contribution of this paper, not done before to the best of our knowledge, is a fi rst study on how the use of a particular clausal translation affects the performance of existing CNF solving procedures. We believe that the results obtained will motivate researchers to experiment with this translations and discover new approaches to effi ciently deal with problems containing non-clausal information.

References

Achlioptas, D., and Peres, Y. 2003. The threshold for random k-SAT is $2^k(\ln 2 - O(k))$. In *STOC '03: Proceedings of the thirty-fi fth annual ACM symposium on Theory of computing*, 223–231. San Diego, CA, USA: ACM Press.

Altamirano, E., and Escalada-Imaz, G. 2000. An efficient proof method for non-clausal reasoning. In *ISMIS '00: Proceedings of the 12th International Symposium on Foundations of Intelligent Systems*, 534–542. Springer-Verlag.

Barthel, W.; Hartmann, A.; Leone, M.; Ricci-Tersenghi, F.; Weigt, M.; and Zecchina, R. 2002. Hiding solutions in random satisfiability problems: a statistical mechanics approach. *Phys. Rev. Lett.* 88.

Berre, L., and Simon. 2004. Fifty-five solvers in Vancouver: the SAT 2004 competition. In *Theory and Applications of Satisfi ability Testing: 7th International Conference, SAT 2004*, Lecture Notes in Computer Sciences.

Cook, S. A., and Mitchell, D. G. 1997. Finding hard instances of the satisfiability problem: A survey. In *Discrete Mathematics and Theoretical Computer Science*, volume 5 of *DIMACS*. American Mathematical Society.

Creignou, N., and Daude, H. 2002. Random generalized satisfiability problems. In *SAT 2002: Fifth International Symposium on the Theory and Applications of Satisfi ability Testing*.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem-proving. *Communications of the ACM* 5:394–397.

Dubois, O., and Dequen, G. 2001. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In *Proc. of the 17th International Joint Conference on Artifi cial Intelligence (IJCAI' 01)*.

Dubois, O.; Boufkhad, Y.; and Mandler, J. 2000. Typical random 3-SAT formulae and the satisfiability threshold. In *SODA'00: Proc. of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, 126–127. San Francisco, CA: Society for Industrial and Applied Mathematics.

Friedgut, E. 1999. Sharp thresholds of graph properties and the k-SAT problem. *Journal of the American Mathematical Society* 12(4):1017–1054.

Gent, I. P.; Macintyre, E.; Prosser, P.; Smith, B. M.; and Walsh, T. 2001. Random constraint satisfaction: Flaws and structure. *Constraints* 6(4):345–372.

Giunchiglia, E., and Sebastiani, R. 2000. Applying the Davis-Putnam procedure to non-clausal formulas. In *AI*IA '99: Proceedings of the 6th Congress of the Italian Association for Artifi cial Intelligence on Advances in Artifi cial Intelligence*, 84–94. Springer-Verlag.

Heule, M.; van Zwieten, J.; Dufour, M.; and van Maaren, H. 2004. March_eq: Implementing additional reasoning into an efficient lookahead SAT solver. In *Theory and Applications of Satisfi ability Testing: 7th International Conference, SAT 2004*, Lecture Notes in Computer Sciences.

Kaporis, A. C.; Kirousis, L. M.; and Lalas, E. 2003. Selecting complementary pairs of literals. In *Proc. LICS'03 Workshop on Typical Case Complexity and Phase Transitions*.

Mitchell, D. G., and Levesque, H. J. 1996. Some pitfalls for experiments with random SAT. *Artifi cial Intelligence* 81:111–125.

Mitchell, D. G.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Proceedings of the Tenth National Conference on Artifi cial Intelligence (AAAI-92)*, 459–465.

Monasson, R., and Zecchina, R. 1997. Satistical mechanics of the random k-SAT model. *Phys. Rev. E* 56:1357–1370.

Moskewicz, M.; Madigan, C.; Zhao, Y.; Zhang, L.; and Malik, S. 2001. Chaff: Engineering an efficient SAT solver. In *39th Design Automation Conference (DAC 2001)*.

Plaisted, D. A., and Greenbaum, S. 1986. A structure-preserving clause form translation. *J. Symb. Comput.* 2(3):293–304.

Stachniak, Z. 2002. Going non-clausal. In *SAT 2002: Fifth International Symposium on the Theory and Applications of Satisfi ability Testing*.

Thiffault, C.; Bacchus, F.; and Walsh, T. 2004. Solving non-clausal formulas with DPLL search. In *Theory and Applications of Satisfi ability Testing: 7th International Conference, SAT 2004*, number 3258 in Lecture Notes in Computer Sciences, 663–678.