On the Random Generation of 3-SAT Instances*

Antoine Rauzy
LaBRI, Université Bordeaux I
351, Cours de la Libération
33405 Talence Cedex
FRANCE
rauzy@labri.u-bordeaux.fr

February 28, 1995

Abstract

Deep results have been obtained recently on randomly generated k-SAT instances. They have been shown hard on average, threshold phenomena have been experimentally established and greedy local search methods have been demonstrated very efficient on these instances.

In this paper, we show that it is possible to generate randomly k-SAT instances that are harder (or easier) than those obtained with the usual generation model. The same method permits also the random generation of hard satisfiable instances. This is of a particular interest to evaluate incomplete methods. We report some experimental results for the Davis and Putnam's procedure — that can be considered as a sample of complete methods — and for the Selman's GSAT method as well — that can be considered in turn as a sample of incomplete methods.

1 Introduction

Let $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a Boolean expression in conjunctive normal form, i.e. where each *clause* C_i is a disjunction of literals and each *literal* is either a variable x_i or its negation $\neg x_i$ $(1 \leq i \leq n)$.

The SAT(isfiability) problem consists in determining whether such an expression ϕ is true for some assignment of Boolean values to the variables $x_1, \ldots, x_n$. Cook [Coo71] showed that SAT is NP-complete and it is now the reference NP-complete problem [GJ79]. ϕ is said to be a SAT-instance. A k-SAT instance is a SAT instance whose clauses are all of length k, i.e. contain k literals. There exist linear algorithms to solve the 2-SAT problem (see for instance [APT79]). k-SAT for $k \geq 3$ is equivalent to SAT, i.e. is NP-complete.

*This work is supported by the french inter-PRC project "Classes Polynomiales"

In this paper, we present a new method to generate randomly k-SAT instances that are statistically hard for the usual resolution methods. This method can be adapted to generate randomly satisfiable k-SAT instances that have the same property. In the sequel, by abuse of terminology, we say simply "hard" for "statistically hard for the usual resolution methods".

In order to test the practical efficiency of an algorithm designed to solve SAT, one needs a benchmark. An easy way to build such a benchmark consists in randomly generating k-SAT instances (and more specifically 3-SAT instances). The standard generation model – so-called fixed length model – consists in drawing independently the C clauses of the instance. In order to draw each clause, one first chooses randomly k distinct variables among a set V, all of the variables having the same probability to be drawn, namely $1/|V|$. Then, a sign is randomly chosen for each variable, positive and negative signs having the same probability to be drawn, namely $1/2$.

Other generation models have been proposed but it is has been shown that they build easy instances (see for instance [Fra86]). On the converse, Chvàtal and Szemerédi, in their very deep paper [CS88], showed that the above one captures in some sense the hardness of SAT. More precisely, they showed that:

1. For any $k \geq 3$, beyond a given ratio C/V, the probability to draw a satisfiable k-SAT instance tends to 0 as V tends to infinity.

2. For a given ratio C/V, the mean number of resolution steps required to show that a randomly generated instance is unsatisfiable increases exponentially with V.

Since, as noted by Gallil in [Gal77], resolution based methods can be seen as non-deterministic counterparts of enumerative ones, the Chvàtal and Szemerédi's result holds also for Davis and Putnam's procedure [DLL62] and the related algorithms, and thus for most of complete methods used to solve SAT.

It was known, at least since the paper by Haken [Hak85], that there exist families of parametric formulae — typically those encoding the well known Pigeon-Hole problem — for which the length of the shortest resolution proof cannot be bound by any polynomial. The major interest of the Chvàtal and Szemerédi's result is that it gives a method to build as many such formulae as one wants.

The interest in randomly generated k-SAT instances has been increased recently by an experimental result due to several authors independently [DC91, MSL92, LT93, CA93] that remarked that k-SAT instances obey a 0/1 law. Below a ratio C/V that depends on k, the probability to draw a satisfiable instance tends to 1 as V tends to infinity. Beyond this ratio it tends (quickly) to 0. For $k = 2$ the threshold has been actually established equal to 1 by, among other, Chvàtal and Reed [CR92]. For $k = 3$ and $k = 4$ experimental values have been found (respectively 4.25 for $k = 3$ and 9.8 for $k = 4$). Researchers work hard to find (theoretical) lower and upper bounds for these values and Dubois in [DABC94] gives a general equation $ln(2) - V.2^k - exp(kV/(2^k - 1) = 0$, for which the upper zeros would be very close to the values of the thresholds of k-SAT instances (for k=3 and beyond).

Threshold phenomena also appear for instances having clauses of different lengths [GW94].

In a practical point of view, one can observe that the hardest randomly generated k-SAT instances for Davis and Putnam's procedure are those around the threshold [CKT91, MSL92, DABC94, Rau94] that is where about 50% of the drawn instances are satisfiable. Even for small numbers of variables (say 50) running times suddenly increase near the threshold and quickly decrease after. Satisfiable instances are significantly easier than unsatisfiable ones, but they are hard too, i.e. that running times quickly increase as V increases.

This paper is an attempt to answer the following questions:

1. Is it possible to generate randomly k-SAT instances that are harder to solve than those generated with the fixed length model, or in other words are the latter the hardest possible ?

2. Is it possible to generate randomly satisfiable k-SAT instances that are at least as hard as those generated with the fixed length model ?

The second question is very interesting because Selman, in a series of recent papers (starting with [SLM92]), showed that greedy local search methods could find models of hard instances with 1000 variables and more, i.e. far beyond the limits of Davis and Putnam's procedure (that cannot currently go beyond 400 variables). However, greedy local search methods — a family to which belongs simulated annealing, tabu, ... — are incomplete and thus it would be very interesting to test them on sets of hard satisfiable instances in order to have a precise evaluation of their success rates.

Our starting remark was that a set of clauses is satisfiable if and only if it contains no positive clause, up to a renaming. Thus, in order to be sure to generate satisfiable instances, it suffices to forbid the drawing of positive clauses. However, this cannot be done roughly otherwise one loses the property that positive and negative literals have the same frequency which is actually essential to obtain hard instances. The idea is thus to draw the clauses in two steps: first one draws k distinct variables, second one draws the structure for the clause, i.e. the sequence of the signs of its literals. There are $2^k - 1$ allowed structures and the problem is to define the probability of each structure in such a way that positive and negative literal frequencies are balanced. Of course, the same idea applies also for instances that are not a priori satisfiable (and thus may contain positive clauses).

In the remaining of this paper, we report the experimental results we have obtained with the generation model described above. This new generation model is detailed at the next section. Performances of the Davis and Putnam's procedure — that can be considered as a sample of complete methods — are given section 3. Those of the Selman's GSAT algorithm — that can be considered as a sample of incomplete methods — are given section 4.

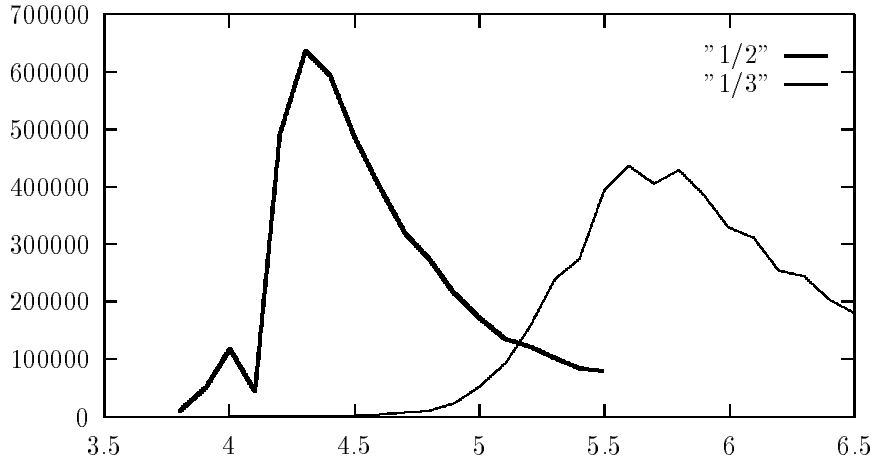

Figure 1: Running times for different probabilities of drawing a positive literal

2 Random Generation of 3-SAT Instances

Preliminary Observations Our first point is to give empirical evidence that the frequencies of positive and negative literals must be balanced in order to obtain hard instances. As the matter of fact, if it is not the case, the generated instances tend to be Horn-sets, i.e. sets of clauses with at most one positive literal per clause (up to a renaming). In [Rau94], it is shown that the Davis and Putnam's procedure (that will be detailed at section 3) is quadratic on Horn-sets. Fig. 1 presents the curves of running times of this procedure for the probabilities 1/3 and 1/2 of tossing a positive literal.

Ratios C/V, where C is the number of clauses and V is the number of variables, are indicated as abscissae. These curves are pictured by extrapolating points obtained for the values 3.8, 3.9, 4.0, 4.1, ... of the ratio C/V. Each point is obtained by means of 100 draws of 3-SAT instances built over 200 variables. In remaining of this paper, each presented curve has been obtained in the same way. 200 is almost the maximum number of variables we can handle within reasonable running times (say few seconds) with our implementation of the Davis and Putnam's procedure on our computer (a SUN sparc station IPX). 100 draws is also a tradeoff between running times and precision of the measures. In our experience, at least first order phenomena are captured with this sample size.

On Fig. 1, one can see first that a difficulty peek — corresponding to a threshold phenomenon — still exists, even shifted to the right and attenuated, for the probability 1/3 of drawing a positive literal. Second, that the top of the peek is higher for the probability 1/2 than for the probability 1/3.

If variables have different probabilities to be drawn, the situation is slightly more complicated, but not essentially different. In [GS94], Génisson and Saïs showed that instances

with unbalanced variable frequencies are in general easier to solve than those in which all literals have approximately the same number of occurrences. However, this question would require further investigations.

Anyway, it seems clear that hard instances are obtained when on the one hand positive and negative literals and on the other hand all of the variables have the same probability to be drawn.

Structure-Driven Generation The idea behind the variations of the generation model we have studied is to toss a *structure* for each clause rather than to draw individually the sign of each literal. We call structure the sequence of the signs of the literals of the clause. For a clause of length, k there are 2^k possible structures. For instance, for a clause of length 3, the 8 possible structures are $---$, $--+$, $-+-$, $-++$, $+--$, $+-+$, $++-$ and $+++$.

We denote by $p(\star\star\star)$ the probability given to the drawing of the structure $\star\star\star$. In the standard generation model, all of the structures are given the same probability to be drawn, namely $1/2^k$. However, it is possible to draw instances with balanced sign frequencies in which the structures are given different probabilities to be drawn. To be such, these probabilities must obey the two following conditions.

$$\sum_{\star\in\{+,-\}} p(\star\star\star) = 1 \tag{1}$$

$$\begin{aligned} &3 \times p(---) + 2 \times p(--+) + 2 \times p(-+-) + p(-++) + 2 \times p(+--) + p(+-+) + p(++-) \\ &= \\ &p(--+) + p(-+-) + 2 \times p(-++) + p(+--) + 2 \times p(+-+) + 2 \times p(++-) + 3 \times p(+++) \end{aligned} \tag{2}$$

Equation 1 is the basic rule of probabilities. The left member of equation 2 denotes the expectation of the number of negative literals in a clause. Its right member denotes the expectation of the number of positive literals in a clause. This equation can be simplified as follows.

$$3 \times p(---) + p(--+) + p(-+-) + p(+--) = p(-++) + p(+-+) + p(++-) + 3 \times p(+++) \tag{3}$$

However, for sake of symmetry, we over-constrain the condition 3 by means of conditions 4 and 5.

$$p(---) = p(+++) \tag{4}$$

$$p(--+) = p(-+-) = p(-++) = p(+--) = p(+-+) = p(++-) \tag{5}$$

Finally, equations 1, 4 and 5 define a single equation with two unknowns:

$$2 \times \alpha + 6 \times \beta = 1 \tag{6}$$

Where α stands for the probability given to the drawing of the structures $---$ and $+++$ and β stands for the probability given to the drawing of the structures $--+$, $-+-$, $-++$, $+--$, $+-+$ and $++-$.

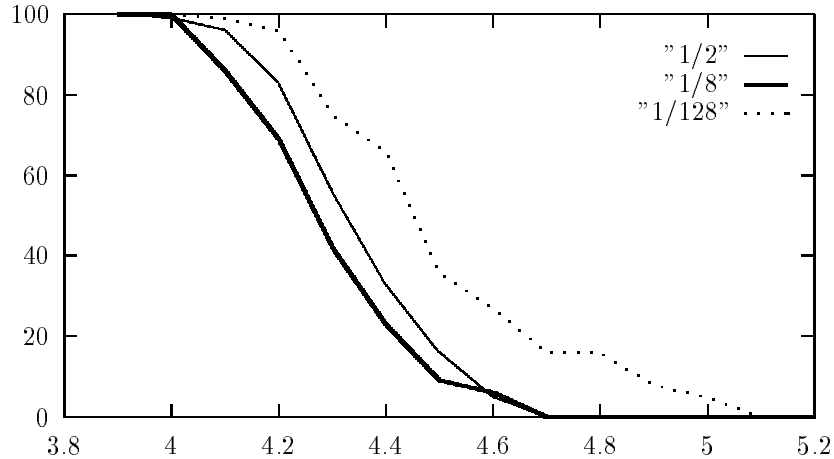

Figure 2: Percentages of satisfiable instances for different values of α

Note that the fixed length generation model is obtained for $\alpha = \beta = 1/8$. Now, if $\alpha = 0$ (and $\beta = 1/6$), the generated instances are always satisfiable and admit at least two solutions: the assignments giving respectively the value 0 and the value 1 to all of the variables (each clause contains at least a negative literal and at least a positive literal). Finally, if α reaches its maximum, namely $1/2$ (in which case $\beta = 0$), the generated instances contain only positive and negative clauses.

This raises a question: what happens for positive values of α ? Curves of percentages of satisfiable instances for the values $1/2$, $1/8$ and $1/128$ of α are pictured Fig. 2 (we do not draw more curves for sake of clarity of the picture, but the presented results are confirmed by measures for a dozen of values of α).

On Fig. 2, one observes the following points.

1. For the values of α greater than $1/8$, up to its maximum value $1/2$, the curves are very similar to the one obtained for $\alpha = 1/8$. The greater α is, the more the corresponding curve is close to the 0/1 law. Moreover, its seems that threshold value is the same for all these values of α.

2. The threshold phenomenon is still observable for very low values of α, in other words a very low number of positive clauses suffices to make instances unsatisfiable. However, the percentage of satisfiable instances increases as α decreases, for a given value of the ratio C/V.

As we will see, these observations can be directly translated as a measure of hardness of instances: the more α is high, the more generated instances are hard to solve.

Generation of Satisfiable Instances Our first aim when starting this work was to generate randomly hard satisfiable instances in order to test efficiently the success rate of incomplete methods. Our starting remark was that an instance is satisfiable if and only if it contains no clause without negative literal, up to a renaming. We call renaming the operation that substitutes simultaneously and for some variables, positive literals for negative literals and conversely negative literals for positive literals. Given a total assignment that satisfies a set of clauses S, in order to obtain from S a set without positive clause it suffices to rename the variables taking the value 1 in the assignment.

For instance, the set $S = (a \vee b \vee \neg c) \wedge (\neg a \vee b) \wedge (\neg b \vee \neg c) \wedge (c \vee a)$ is satisfied by the assignment $[a \leftarrow 1, b \leftarrow 1, c \leftarrow 0]$ and thus the renaming of the literals built over a and b transforms S into a set S' that contains no positive clause ($S' = (\neg a \vee \neg b \vee \neg c) \wedge (a \vee \neg b) \wedge (b \vee \neg c) \wedge (c \vee \neg a)$).

The idea is thus to perform structure driven generations in which the probability given to the drawing of the structure $+++$ is zero.

In order to keep the property that the expectation of the number of positive and negative literals are equal, the generation must still obey conditions 1 and 3, but now $p(+++) = 0$. For sake of symmetry, we also impose the conditions 7 and 8.

$$p(--+) = p(-+-) = p(+--) \tag{7}$$

$$p(-++) = p(+-+) = p(++-) \tag{8}$$

Finally, equations 1, 3, 7 and 8 define a system of two equations with three unknowns:

$$\alpha + 3 \times \beta + 3 \times \gamma = 1 \tag{9}$$

$$\alpha + \beta = \gamma \tag{10}$$

Where α stands for $p(---)$, β stands for $p(--+)$, $p(-+-)$ and $p(+--)$, and γ stands for $p(-++)$, $p(+-+)$ and $p(++-)$.

α may vary from 0, in which case $\beta = \gamma = 1/6$, to $1/4$, in which case $\beta = 0$ and $\gamma = 1/4$. We will see in the next section the effects of these variations on the practical difficulty of finding a model of the generated instances.

3 Experimental Results with the Davis and Putnam's Procedure

Algorithm Let S be a set of clauses. A literal is said *monotone* (in S) if its opposite does not occur in S. A literal is said *unit* if it occurs in a unit clause of S (i.e. if it is the unique literal of this clause). We denote by $S_{p \leftarrow v}$ the set S in which the Boolean value v has been assigned to the variable p. I.e. the set of clauses obtained from S by deleting the clauses that contain an occurrence of p satisfied by the assignment and by deleting from the other clauses the occurrences of p falsified by the assignment. By extension, we denote by S_l the set S in which the literal l has been satisfied.

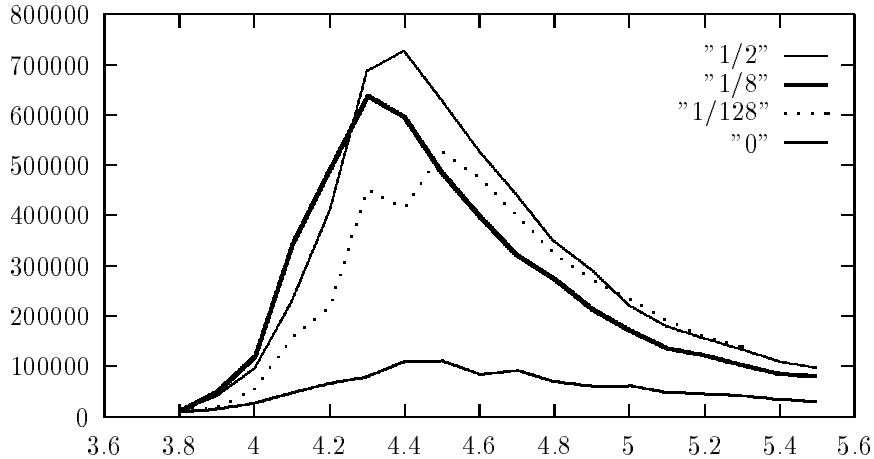

Figure 3: Mixed instances: Running times for different values of α

The Davis and Putnam's procedure [DLL62], can be sketched as follows.

let S the set of clauses to be tested. If the satisfiability of S is not immediately decidable (i.e. if S is not empty and does not contain an empty clause), then there are two cases:

- S contains a monotone or an unit literal l. In this case, one checks the satisfiability of S_l.
- S does not contain a monotone or an unit literal. In this case, a variable p and a Boolean value v are chosen and the algorithm is called recursively first on $S_{p \leftarrow v}$ and second on $S_{p \leftarrow 1-v}$, if $S_{p \leftarrow v}$ is not satisfiable.

In practice, a good heuristics to choose the next literal to assign improves dramatically the performances of the algorithm. Of course, a tradeoff must be found between the cost of a heuristics and its efficiency. For this study, we used a very simple heuristics so called firstfail. It chooses the most frequent variable in the shortest clauses. The chosen value is those corresponding to the most frequent literal. It gives pretty good results and it is better (in our experience) than many other proposed ones, in particular the one of Jeroslow and Wang [JW90] (For a discussion about heuristics see [DABC94]). Anyway, the presented results do not depend on a particular heuristics (as far as our tests with a few number of heuristics are representative).

Performances on Mixed Instances Curves of running times for the value 1/2, 1/8, 1/128 and 0 of α (i.e. the probability of the structures $---$and $+++$) are displayed Fig. 3.

This curves enlighten the following phenomena.

1. If there is no positive and negative clauses ($\alpha = 0$), instances are trivial.

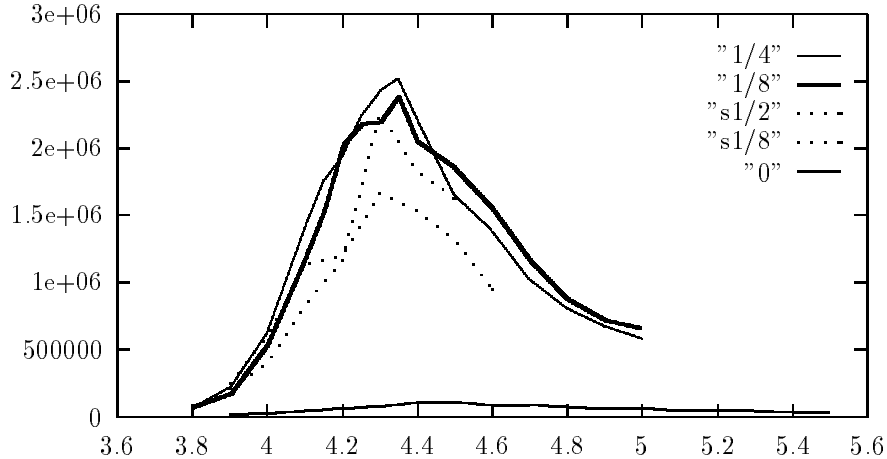

Figure 4: Satisfiable instances: Running times for different values of α

2. For any value of α, one observes a difficulty peek near the threshold.

3. The hardest instances are those drawn with $\alpha = 1/2$, i.e. those that contain only negative and positive clauses. However these instances are not essentially harder than those generated with the fixed length model ($\alpha = 1/8$).

4. Even if the number of positive and negative clauses is quite low (exemplifies here with $\alpha = 1/128$), the generated instances are hard.

Finally, experiments we have performed with other values of α seem to show that the hardness of instances, for a given value of the ration C/V, increases continuously as α increases.

Performances on Satisfiable Instances Fig. 4 permits the comparison of running times of the Davis and Putnam's procedure on satisfiable instances for the two generation models. The curves named `s1/2`, `s1/8` and `0` have been obtained by drawing mixed instances for the values 1/2, 1/8 and 0 of α and by considering among them the satisfiable ones only (this explains why the curve stops at the value 4.7 of the ratio C/V, beyond there is a too small probability of drawing a satisfiable instance). The curves named 1/4 and 1/8 have been obtained with the model described in the previous section — that ensures to draw satisfiable instances —, for the values 1/4 and 1/8 of α.

This curves enlighten the following phenomena.

1. For both models, one observes a difficulty peek near the threshold.

2. For both models, the difficulty of solving satisfiable instances increases as α increases, even if there is not a big difference between the different values of α.

3. By generating directly satisfiable instances, one obtains instances that are harder than those obtained by peeking satisfiable instances among mixed instances.

This last point is questioning. On the one hand, we would like to generate satisfiable instances that are as hard as those generated with the standard model, and our result is thus not absolutely satisfying. On the other hand, our main goal was to generate hard satisfiable instances and we succeed in some sense beyond our initial hope.

It raises two questions:

- Is there a value of α such that the instance generated with the second model are as hard as those generated with the standard model (for the Davis and Putnam's procedure) ? At this point, we are not able to answer this question.

- In a more fundamental point of view, it would be interesting to understand why instances are difficult. It could be the case that the number of negative clauses is a good parameter of the actual difficulty, but this must be understood up to renaming. Note that the Davis and Putnam's procedure + the first fail principle is absolutely stable w.r.t. renaming. I.e. that by renaming some of the variables, one produces only minor variations in the computation times. Renaming could be used to standardize instances but not to make predictions about the hardness of the instance because finding a renaming that gives the minimum number of negative clauses (and keeping the number of positive clauses to zero) is surely as hard as finding a model.

4 Experimental Results with the Selman's GSAT

Algorithm GSAT performs greedy local search for a satisfying assignment of a set of clauses. The procedure starts with a randomly generated truth assignment. It then changes ("flips") the assignment of a variable chosen with a given heuristics. Such flips are repeated until either a satisfying assignment is found or a pre-set maximum number of flips (called max-flips) is reached. This process is repeated as needed up to a maximum of max-tries times. The whole algorithm is pictured Fig. 5. The heuristics we use for our tests is called "Random walk" by Selman and Kautz in [SK93]. It works as follows. With a probability 1/2, a variable is randomly picked in unsatisfied clauses, and with a probability 1/2 a variable is randomly picked among those that give the largest decrease in the total number of unsatisfied clauses. This heuristics allows to escape from local minima and is considered as a pretty good one by Selman and al. (see [SK93]).

For experiments described in this section, we have considered 3-SAT instances built over 200 variables. Max-tries was fixed to 10 and max-flips to 40000. These parameters according to the Selman's papers are a reasonable tradeoff. Anyway, phenomena we observe are in some sense independent of particular values of max-tries and max-flips (provided they are sufficiently large).

```
procedure GSAT
Input: a set of clauses S, max-flips and max-tries.
Output: a satisfying truth assignment of S, if found.
begin
   for i := 1 to max-tries
      σ := a randomly generated truth assignment
      for j := 1 to max-flips
         if σ satisfies S then return σ
         p := a variable chosen with a given heuristics.
         σ := σ with the truth assignment of p reversed.
      end for
   end for
   return "no satisfying assignment found".
end
```

Figure 5: The procedure GSAT

Performances Fig. 6 permits the comparison of the percentages of instances that have been shown satisfiable by GSAT for the two generation models. The curves named `s1/2` and `s1/8` have been obtained by drawing mixed instances for the values 1/2 and 1/8 of α and by considering among them the satisfiable ones only (this explains why the curve stops at the value 4.7 of the ratio C/V, beyond there is a too small probability of drawing a satisfiable instance). The curves named 1/4 and 1/8 have been obtained by drawing directly satisfiable instances, for the values 1/4 and 1/8 of α.

This curves enlighten the following phenomena.

1. For both models and all of the values of α, GSAT finds a solution for almost all of the instances below the threshold. This just confirms Selman's results.

2. For both models and all of the values of α, curves are very similar around the threshold. It means that we succeed in generating randomly satisfiable instances that are at least as hard for GSAT as those generated with the fixed length model.

3. For both models and all of the values of α, the percentage of instances shown to be satisfiable decreases dramatically just after the threshold (for values around 4.7 of the ratio C/V). This percentage even falls near 0% for $\alpha = 1/4$ (in the second model).

4. For all of the value of α, with the second generation model, the percentage of instances shown to be satisfiable increases more or less quickly as the ratio C/V increases (after the minima).

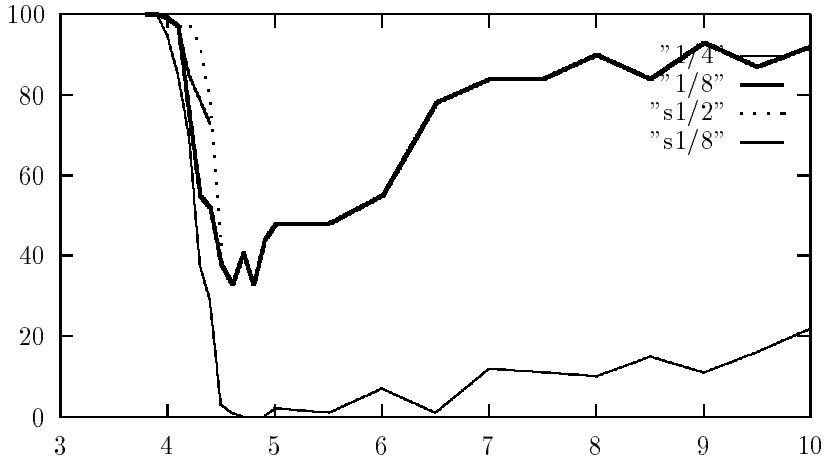

Figure 6: Satisfiable instances: Percentages of success for different values of α

5. In the second model, the more α is high the less GSAT finds solutions (especially after the threshold).

These last points could be explain by variations in the number of solutions or in the structure of solutions that the variations of α introduce:

- It could be the case that for a given value of the ratio, the number of solutions decreases as α increases. This is confirmed by preliminary experiments with instances with a small number of variables (between 50 and 100).
- It could be also the case that for a given value of the ratio, the number of deep local minima increases as α increases, which causes some troubles to GSAT.

However, we do not have practical methods to count the number of solutions or study their structures for not too small instances. And it is not sure that observations made for instances with a quite small number of variables (say between 50 and 100) could be safely extended for instances with 200 variables and beyond. Anyway, this requires further investigations both in a practical and in an analytical points of view. In a practical point of view, several directions could be explored in addition to Davis and Putnam like procedures, including the algorithm proposed by Dubois in [Dub91] and methods derived from Bryant's Binary Decision Diagrams [BRB90].

Finally, Fig. 7 shows the curves of running times of GSAT for both models and different values of α. There is not many things to say about this curves excepted that they are similar before and around the threshold and very instable after. Even with a big number of tested instances, there are big variations in the observed performances.

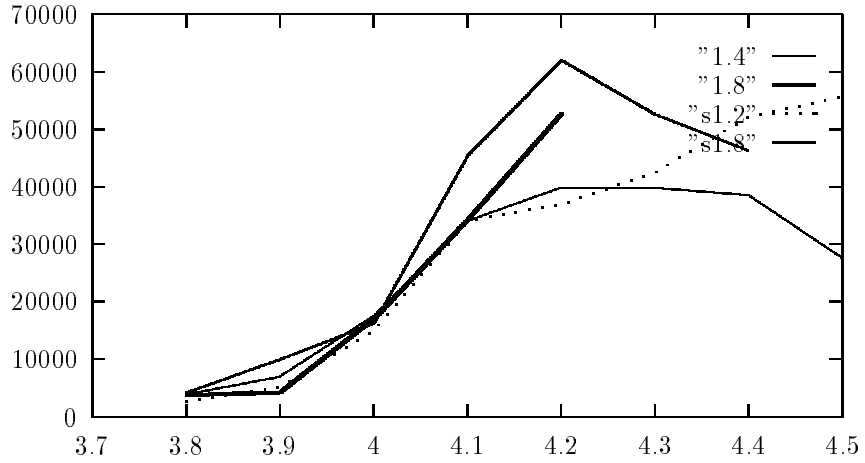

Figure 7: Running times on satisfiable instances for values of α

5 Conclusion

In this paper, we have proposed a method to generate k-SAT instances and satisfiable k-SAT instances that draws clause structures rather than to draw individually each literal signs.

We provide experimental evidence that the generated instances could be at least as hard as those generated with the standard model for both complete methods — such as the Davis and Putnam's procedure — and incomplete methods — such as the Selman's GSAT algorithm. More precisely, we show that the hardness of an instance is strongly related to the percentage of negative clauses it contains (up to a renaming).

As a future work it would be interesting to investigate, both in a practical and in an analytical points of view, in which way this percentage influences the number of solutions or the structure of the solutions.

Acknowledgment

I would like to thank Richard Génisson and Lakdhar Saïs, from the LIM (Marseilles) and Olivier Dubois from the LAFORIA (Paris), for the many fruitful discussions we had on SAT.

References

[APT79] B. Aspvall, M. Plass, and R. Tarjan. A Linear Time Algorithm for Testing the

Truth of Certain Quantified Boolean Formulae. *Information Processing Letter*, 8(3):121–123, 1979.

[BRB90] K. Brace, R. Rudell, and R. Bryant. Efficient Implementation of a BDD Package. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*. IEEE 0738, 1990.

[CA93] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (Washington, D.C., AAAI'1993)*, pages 21–27, 1993.

[CKT91] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the Really Hard Problems Are. In *Proceedings of the International Joint Conference of Artificial Intelligence, IJCAI'91*, 1991.

[Coo71] S.A. Cook. The Complexity of Theorem Proving Procedures. In *Proceedings of the 3rd Ann. Symp. on Theory of Computing, ACM*, pages 151–158, 1971.

[CR92] V. Chvàtal and B. Reed. Miks gets some (the odds are on his side). In *Proceedings of the 33rd IEEE Symp. on Foundations of Computer Science*, pages 620–627, 1992.

[CS88] V. Chvátal and E. Szemerédi. Many Hard Examples for Resolution. *JACM*, 35(4):759–768, october 1988.

[DABC94] O. Dubois, P. André, Y. Boufkhad, and J. Carlier. SAT versus UNSAT, 1994. Position paper, DIMACS chalenge on Satisfiability Testing, to appear.

[DC91] O. Dubois and J. Carlier. Sur le problème de satisfiabilité. Communication at the Barbizon Workshop on SAT, october 1991.

[DLL62] M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem Proving. *JACM*, 5:394–397, 1962.

[Dub91] O. Dubois. Counting the Number of Solutions for Instances of Satisfiability. *Theoretical Computer Science*, 81:49–64, 1991.

[Fra86] J. Franco. On the probabilistic performances of algorithms for the satisfiability problem. *Information Processing Letters*, 23:103–106, 1986.

[Gal77] Z. Gallil. On the Complexity of Regular Resolution and the Davis and Putnam's Procedure. *Theoretical Computer Science*, 4:23–46, 1977.

[GJ79] M.R. Garey and D.S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completness*. Freeman, San Fransisco, 1979.

[GS94] R. Génisson and L. Saïs. Some ideas on random generation of k-SAT instances. In J.M. Crawford and B. Selman, editors, *Proceedings of post-conference workshop on Experimental Evaluation of Reasonning and Search Methods, AAAI'94*, pages 91–92, 1994.

[GW94] I.P. Gent and T. Walsh. The SAT Phase Transition. In A.G. Cohn, editor, *Proceedings of 11th European Conference on Artificial Intelligence, ECAI'94*, pages 105–109. Wiley, 1994.

[Hak85] A. Haken. The intractability of the resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[JW90] R.J. Jeroslow and J. Wang. Solving Propositional Satisfiability Problems. *Annals of Mathematics and Artificial Intelligence*, 1:167–188, 1990.

[LT93] T. Larrabee and Y. Tsuji. Evidence for a satisfiability threshold for random 3cnf formulas. In H. Hirsh and al., editors, *Proceedings of Spring Symposium on Artificial Intelligence and NP-Hard Problems (Stanford CA 1993)*, pages 112–118, 1993.

[MSL92] D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proceedings Tenth National Conference on Artificial Intelligence (AAAI'92)*, 1992.

[Rau94] A. Rauzy. On the Complexity of the Davis and Putnam's Procedure on Some Polynomial Sub-Classes of SAT. Technical Report 806-94, LaBRI, URA CNRS 1304, Université BordeauxI, 9 1994. Submitted to Journal of Logic Programming.

[SK93] B. Selman and H.A. Kautz. Domain Independent Extensions to GSAT : Solving Large Structured Satisfiability Problems. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI'93)*, 1993.

[SLM92] B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, 1992.