Towards Industrial-Like Random SAT Instances*

Carlos Ansótegui
DIEI, UdL
Jaume II 69, Lleida, Spain

Maria Luisa Bonet
LSI, UPC
J. Girona 1–3, Barcelona, Spain

Jordi Levy
IIIA, CSIC
Campus UAB, Bellaterra, Spain

Abstract

We focus on the random generation of SAT instances that have computational properties that are similar to real-world instances. 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. This is not possible, in general, with classical randomly generated instances. We provide different generation models of SAT instances, extending the uniform and regular 3-CNF models. They are based on the use of non-uniform probability distributions to select variables. Our last model also uses a mechanism to produce clauses of different lengths as in industrial instances. We show the existence of the phase transition phenomena for our models and we study the hardness of the generated instances as a function of the parameters of the probability distributions. We prove that, with these parameters we can adjust the difficulty of the problems in the phase transition point. We measure hardness in terms of the performance of different solvers. We show how these models will allow us to generate random instances similar to industrial instances, of interest for testing purposes.

1 Introduction

SAT is an NP-complete problem in the worst case, and in fact a big percentage of formulas need exponential size resolution proofs to be shown unsatisfiable. Nevertheless, state-of-the-art solvers have been shown of practical use working with *real-world* (industrial) 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 is difficult for solvers to perform well on all of them. This has led researchers to say that randomly generated and industrial instances are of distinct nature. It has been postulated that industrial instances have a *hidden structure* that can be exploited. In [12] it is proved that industrial instances contain a small number of variables that, when instantiated, make the formula easy to solve. In [1] it is shown that industrial 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" [11] and in "Ten Challenges Redux : Recent Progress in Propositional Reasoning and Search" [8] as the tenth challenge. Also Rina Dechter in [4] proposes the same objective. In this paper we want to make a contribution in this direction, defining new generators models of random formulas.

Our models are based on 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 [2; 3] 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, to build our first four models (geometric, georegular, powerlaw and powregular), by selecting variables following two distinct probability distributions, one geometric $P(i) = C\, b^{-i}$ and another one power-law $P(i) = C\, i^{-\beta}$. Both distributions include the classical definition as a special case ($b = 1$ or $\beta = 0$). On the other hand, when we move away from these special cases, some variables occur much more frequently than others. In all the previous models the length of clauses is fixed. Since this is not the common

*Research partially supported by the research projects TIN2007-68005-C04-{01,03,04} and TIN2006-15662-C02-02.

structure in industrial instances we introduce a fifth model. This last model iteratively uses a probability distribution to select a variable and another probability distribution to select a clause where to put the variable. We also compare the frequency of number of occurrences of variables and lengths of clauses that we obtain, with the frequencies observed in industrial instances. We show that the power-law distribution fits very well the frequencies observed in the instances of the SAT Race 2008.

We have observed that for all our models we can experimentally identify a phase transition point characterized by a constant ratio on the number of clauses over the number of variables. This is not trivial, since there are variations of our models where the phase transition point would not happen at a fixed ratio. 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 the industrial ones as possible, but not trivial.

Finally, as a test of how real our random instances look like, we have compared the time required by solvers specialized in industrial instances with the time required by solvers specialized in random instances. We observe that, when we move away from the classical case, solvers specialized in industrial instances become faster than solvers specialized in random instances.

2 Description of the Models

We generalize the uniform and the regular random k-CNF generators by selecting variables following two distinct non-uniform distributions, one geometric and the other one power-law.

2.1 Geometric and Powerlaw k-CNF Models

The distributions we use 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 $P(X = i; n) = 1/n$.

Given a continuous probability distribution ϕ with domain $\big[0, 1\big]$, we can easily generate a family of probability distributions $P(X = i; n)$, with discrete domain $i = \{1, \ldots, n\}$, as follows. We break the interval $\big[0, 1\big]$ into n pieces, obtaining the points $1/n, 2/n, \ldots, 1$. Then $P(X = i; n)$ is defined to be $\phi(i/n)$ with the appropriated normalization. This results into:

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

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

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

We should say that we don't use in our models the previous approximation even though the error it gives is sufficiently small. We calculate the exact normalization constant.

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**
      p = rand( ); v = 1;
      while $p > P(v; b, n)$ **do**
        $p = p - P(v; b, n)$; $v = v + 1$;
      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

Figure 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.

We have observed that defining our probability distributions this way, allows us to generate formulas that have a phase transition phenomena. Not all distributions to select variables will give us formulas with this property.

To define the geometric k-CNF model we use the following probability density function

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

where $b > 1$. Notice that, since $\lim_{b\to 1} \phi^{geo}(x; b) = 1$, at the limit $b = 1$ we obtain the uniform distribution. Hence, the uniform k-CNF model is a particular case of the geometric k-CNF model with $b = 1$.

The continuous probability function $\phi^{geo}(x; b)$ results into the following family of discrete probability distributions:

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

In the powerlaw model we would have to use the continuous probability distribution $\phi^{pow}(x; \beta) = (1-\beta)\, x^{-\beta}$. However, this function is not defined in $x = 0$, so a small change is necessary. We use the interval $\big[0 + \epsilon, 1 + \epsilon\big]$, for a small value of ϵ, or equivalently we use the function

$$\phi^{pow}(x; \beta) = \frac{1 - \beta}{(1 + \epsilon)^{1-\beta} - \epsilon^{1-\beta}}\, (x + \epsilon)^{-\beta}$$

Using ϕ^{pow} and normalizing we obtain the following family of discrete probability distributions:

$$P(X = i; \beta, n) = \frac{(i + \epsilon \cdot n)^{-\beta}}{\sum_{j=1}^{n} (j + \epsilon \cdot n)^{-\beta}}$$

Notice that, since $\phi^{pow}(x; 0) = 1$, the powerlaw k-CNF model is a generalization of the uniform k-CNF model where $\beta = 0$. Geometric k-CNF formulas may be generated with the algorithm in Figure 1. To generate powerlaw formulas we just substitute $P(v; b, n)$ by $P(v; \beta, n)$ in the algorithm.

Input: n, m, k, b
Output: a k-SAT instance with n variables and m clauses
$bag = \emptyset$;
for $v = 1$ **to** n **do**
  $bag = bag \cup \{\lfloor P(v; b, n)\frac{k\,m}{2} \rfloor$ copies of $v\}$;
  $bag = bag \cup \{\lfloor P(v; b, n)\frac{k\,m}{2} \rfloor$ copies of $\neg v\}$;
endfor
$S =$ subset of $k\,m - |bag|$ literals from $\{1, \ldots, n, \neg 1, \ldots, \neg n\}$
  maximizing $P(v; b, n)\frac{k\,m}{2} - \lfloor P(v; b, n)\frac{k\,m}{2} \rfloor$
$bag = bag \cup S$;
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

Figure 2: Geo-regular k-CNF generator

2.2 Geo-regular and Pow-regular k-CNF Models

In regular k-CNF formulas all literals occur nearly the same number of times, i.e. $\lfloor \frac{k \cdot m}{2n} \rfloor$ or $\lfloor \frac{k \cdot m}{2n} \rfloor + 1$ times. In geo-regular we want the variables to occur with a frequency given by $P(X = i; b, n)$. Geo-regular k-CNF formulas are generated as follows. We construct a multiset bag with approximately $P(X = v; b, n)\frac{k\,m}{2}$ copies of the literals v and $\neg v$, for each variable $v = 1, \ldots, n$. 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 in Figure 2 describes this procedure. Notice that when a tautology or simplifiable clause is generated (i.e. a clause with repeated variables), we discard all generated clauses, not just the last one. To see that the algorithm terminates, we need to prove that a formula without tautologies and simplifiable clauses exists. For this it is sufficient to show that the most frequent variable has at most m copies in the bag. We choose the parameter b in a range that ensures this.

The Pow-regular k-CNF formulas are generated identically, but substituting everywhere $P(X = i; b, n)$ by $P(X = i; \beta, n)$ in the algorithm.

2.3 Models with Clauses of Variable Size

The double-powerlaw model (Figure 3) constructs a formula by repeating the following process. It chooses a variable and a clause following two (not necessarily equal) distributions, and includes the variable with an arbitrary sign in the clause.

3 Industrial SAT Instances

In the previous section we have described generalized models of random SAT instances. We want this models to generate formulas as close as possible to industrial ones. Therefore, to choose the distribution ϕ (geometric or powerlaw), and the best value of the parameter (b for the geometric and β for the powerlaw), we have analyzed some industrial instances. We have studied the 100 benchmarks (all industrial) used in the SAT Race 2008. All together, they contain $n = 25693792$

Input: $n, m, k, \beta_v, \beta_c$
Output: a SAT instance with n variables, m clauses
for $i = 1$ **to** m **do**
  $C_i := \square$;
for $i = 1$ **to** $k * m$ **do**
  repeat
    $p := rand()$; $v := 1$;
    while $p > P(v; \beta_v, n)$ **do**
      $p := p - P(v; b, n)$; $v := v + 1$;
    endwhile
    $p := rand()$; $c := 1$;
    while $p > P(c; \beta_c, m)$ **do**
      $p := p - P(v; b, n)$; $c := c + 1$;
    endwhile
  while $v \in C_c$
  $C_c := C_c \vee (-1)^{rand(2)} \cdot v$;
endfor

Figure 3: Double-powerlaw CNF generator.

variables, with a total of $\sum_{i=1}^{n} N(i) = 349760681$ occurrences, where $N(i)$ is the number of occurrences of the variable i. Therefore, the average number of occurrences per variable is $E[N(i)] = \sum_{i=1}^{n} N(i)/n = 13.6$. If we used the classical (uniform) random model to generate instances with this average number of occurrences, most of the variables would have a number of occurrences very close to 13.6. However, in the analyzed industrial instances, close to 90% of the variables have less than this number of occurrences, and more than 60% have 6 or less occurrences. The big value of the average is produced by a small fraction of the variables that have a huge number of occurrences. This indicates that the number of occurrences could be better modeled with a power-law distribution as suggested by [3].

We can estimate the form of the function ϕ that best fits these industrial instances, as follows. Compute the number of occurrences $N(i)$ of each variable i of the set of variables of the industrial instance. Sort them in decreasing order in the number of occurrences, i.e. assume $N(i) \geq N(i+1)$, for $i = 1, \ldots, n-1$. Then, approximating ϕ as $N(i)$ and normalizing it conveniently, we obtain

$$\phi^{ind}(i/n) = \frac{n}{\sum_{j=1}^{n} N(j)}\, N(i)$$

for $i = 1, \ldots, n$. The resulting function is shown in Figure 4 in red.

Notice that $\log \phi^{geo}(x;\, b)$ is linear on x, whereas $\log \phi^{pow}(x;\, \beta)$ is linear on $\log x$. Therefore, if we plot ϕ^{ind} on double logarithmic axes and on semi-logarithmic axes, we can see which one of the theoretical functions, ϕ^{geo} or ϕ^{pow}, best fits ϕ^{ind}. As shown in Figure 4 (right) $\phi^{ind}(x) \approx \phi^{pow}(x;\, 0.82)$. The coincidence between both functions has two reasons. First, as suggested in [3], the distribution of frequencies on the number of occurrences of variables follows a power-law distribution, in industrial SAT instances. This coincidence validates the hypothesis of that paper. Second, the use of a (continuous) power-law distribution (with domain $[1, 0]$) to assign probabilities to variables, has as a consequence, a (discrete) power-law distribution on the frequencies

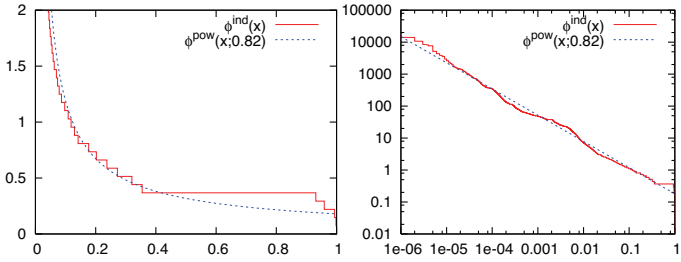

Figure 4: Estimated industrial function $\phi^{ind}(x)$ (in red) and powerlaw function $\phi^{pow}(x; 0.82)$ (in blue), with normal axes (left) and double-logarithmic axes (right).

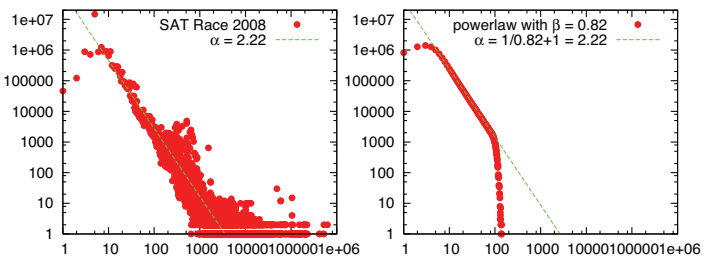

Figure 5: Comparison of the frequencies of variable occurrences obtained for the SAT Race 2008, and the random model powerlaw with $\beta = 0.82$. It is also shown the line with slope $\alpha = 1/0.82 + 1 = 2.22$.

of occurrences (with a different exponent). This second result is proved in the following theorem.

Theorem 1 *In the powerlaw model, with* $\phi^{pow}(x; \beta) = (1 - \beta)\, x^{-\beta}$*, when* n *tends to* ∞*, the probability that a variable has* k *occurrences follows a powerlaw distribution* $P(k) \sim k^{-\alpha}$*, where* $\alpha = 1/\beta + 1$.

PROOF: Let $P(X = i, \beta) = C \cdot (i/n)^{-\beta}$ be the probability of choosing variable i in the powerlaw generation model. Let $N(i)$ be the number of occurrences of variable i in a randomly generated formula F. We have $E[N(i)] = C \cdot (i/n)^{-\beta} \cdot |F|$. Chernoff's or Hoeffding's bounds ensure that, for big n's, $N(i)$ is approximately $E[N(i)]$. Hence, $N(1) > N(2) > \cdots > N(n)$ with high probability.

Now we want to approximate the provability $F(k) = P(K \geq k)$ that a variable occurs at least k times. Let i be the index of a variable satisfying $E[N(i)] = k$. For big n, all variables with index smaller that i have more than k occurrences, and those with indexes between $i + 1$ and n have less than k occurrences. Therefore, $P(K \geq k) = i/n$, for the particular i defined above. From $E[N(i)] = k$ and $E[N(i)] = C \cdot (i/n)^{-\beta} \cdot |F|$ we obtain

$$F(k) = P(K \geq k) = i/n = \left(\frac{k}{C\,|F|}\right)^{-1/\beta}$$

We can approximate the probability $f(k) = P(K = k)$ as the derivative of $F(k)$,

$$f(k) \approx -\frac{\partial}{\partial k}\left(\frac{k}{C\,|F|}\right)^{-1/\beta} = \frac{\beta}{(C\,|F|)^{-1/\beta}}\, k^{-1/\beta - 1}$$

Hence we obtain a discrete power-law (zeta) distribution with exponent $\alpha = 1/\beta + 1$. ■

The previous theorem can be generalized only assuming that $P(X = i)$, i.e. ϕ, decreases monotonously. The approximations used in this theorem are validated by the following experiment. We have generated a random formula with the powerlaw model with 10^7 variables, $2.5\,10^7$ clauses and $\beta = 0.82$. The frequencies of occurrences of variables are shown in Figure 5 and are compared with those obtained for the SAT Race 2008, and the line with slope $\alpha = 1/0.82 + 1 = 2.22$.

4 Experimental Results

First, we want to locate the phase transition points, and second, we want to study the performance of some solvers on the instances generated with our models.

In order to choose the SAT solvers we took the best ones from the industrial and random categories at the SAT 2007 Competition. Then, we selected from each category the best performing one on the instances we have generated. The SAT solver minisat (v.2) [6] was selected as the specialized solver in industrial instances. Solvers kcnfs (v.2004) [5] and march_ks [7] were selected as specialized solvers in random instances. gnovelty+ [10] was selected as an incomplete local search solver, that usually performs well on satisfiable random instances. We also provide results with the SAT solver satz (v.2004) [9] that has shown a good average performance.

All the experiments were run on a 1Ghz machine with 1Gbyte of RAM.

We have identified the existence of a phase transition point for all the generation models we have introduced. We provide results for geometric and geo-regular with $b \in \{1, 2, 4, 6, 16\}$, and for powerlaw and pow-regular with $\beta \in \{0, 0.25, 0.5, 0.75, 1\}$. Figure 6 shows the clause/variable ratio at the phase transition points (the exact points are in Table 1). Each data point represents the results on the computation of 200 instances. The length of the clauses, k, was set to 3. Recall that for $b = 1$ ($\beta = 0$), the geometric (powerlaw) and geo-regular (pow-regular) are the uniform and regular random k-CNF generators, respectively. Their phase transition point is located around the clause/variable ratio 4.25 and 3.55, respectively.

We can observe that as we increase the base b or the exponent β the phase transition ratios become lower. Also we see that the regular models have always a lower ratio, and the regular and non regular models seem to converge as we increase b or β.

As we discussed in the introduction, our goal is to generate random instances with properties similar to industrial instances. One way of checking that we are going in the right direction is to show that SAT solvers specialized in industrial instances have better performance than the ones specialized in random instances, contrarily to what we could expect since these instances are indeed random.

We also want to conduct such experimental investigation

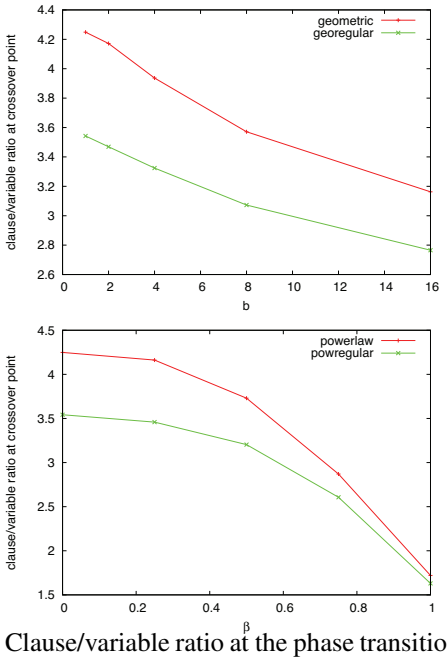

Figure 6: Clause/variable ratio at the phase transition point as a function of b. We use 200 instances per point.

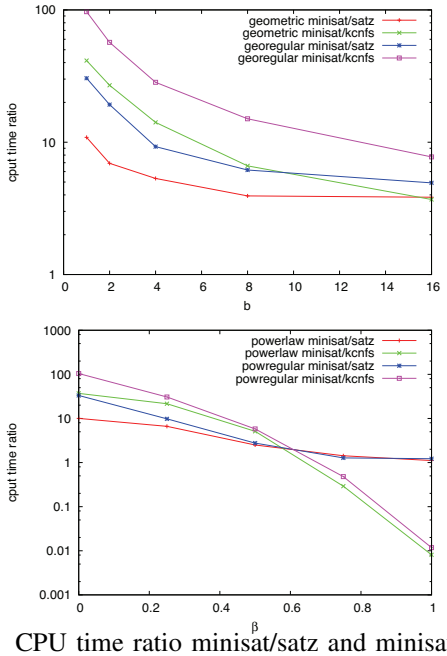

Figure 7: CPU time ratio minisat/satz and minisat/kcnfs as a function of b or β for the models: geometric, geo-regular, powerlaw and pow-regular. We use 200 instances per point.

on the hardest instances our model can produce for a particular number of variables. The hardest instances are usually located at the phase transition region, since these instances are (almost) minimally unsatisfiable or have very few models. So the instances we use always have clause/variable ratio around the previously computed phase transition point. We increase the number of variables until we achieve a median time of around 100 seconds to solve the instances using the SAT solver satz, which we use as a reference of the difficulty of the instances.

Since we wanted to compare how good SAT solvers specialized in industrial instances are on these random instances, we present the ratio of minisat versus satz and kcnfs. Figure 7 shows the results of the comparisons (the numerical data can be found at Table 1).

As we expected, the regular models are always harder for all solvers. For the geometric and geo-regular models, minisat is always worse than the other two solvers, although the difference becomes smaller as we increase b. However, for the powerlaw and pow-regular models, minisat outperforms kcnfs as we increase β and becomes as good as satz. In particular, with the pow-regular model, for $\beta = 0$ kcnfs is two orders of magnitude better than minisat, but for $\beta = 1$, the situation becomes the opposite.

Although we have shown that for high values of β we may be achieving the properties that we were looking for, we still have some unresolved issues: (i) minisat is not outperforming satz, which is worse than minisat on several industrial instances; (ii) our instances have a fixed clause length contrary to what happens for industrial instances; and (iii) the instances we are generating are really small (though challenging) compared to the size of some industrial instances, which have hundreds of thousands of variables and can be solved in seconds by an specialized industrial SAT solver.

Therefore, we still need to improve our best generation models so far, powerlaw and pow-regular. The solution also comes from our previous study at section 2 on industrial instances, were apart form observing that the occurrences of variables may follow a powerlaw distribution, also the occurrences of the length of clauses may follow a powerlaw distribution.

Table 1 also shows the results for the double-powerlaw model, for $\beta = 0.75$ and $k = 5$, where k is now the average size of the length of clauses. The β parameter is obtained from our study in Section 3, and the k parameter is set to 5 to ensure that the instances are not too easy. Our results are calculated at the phase transition point as in the previous models. As we can see, finally, we get bigger instances that can be solved easily by minisat, and take a longer time for the satz and march_ks solvers. We do not provide experimental results using kcnfs since this solver does not accept clauses of the length we have generated.

We have to say though, that the great difference in the median time comes directly from solving the satisfiable instances, since for the unsatisfiable instances, satz and march_ks take around 10 seconds. We think that this is because in the double-powerlaw model, the powerlaw distribution gives the frequencies of big clauses (tail of the distribution), but the frequency of small clauses has to be adjusted with other methods. Notice that a big number of small clauses using very frequent variables results into easy to prove unsatisfiable formulas. We made the decision not to include hacks to avoid this, and to present clearly the experimental data. We also wanted to test if the satisfiable instances could be easily solved by an incomplete solver specialized in random

b, β	m	n	m/n	minisat	satz	kcnfs
geometric						
1	1657	390	4.248	1027	94	25
2	1773	425	4.171	649	94	24
4	2047	520	3.936	476	89	33
8	2357	660	3.571	372	94	56
16	2625	830	3.162	343	90	92
geo-regular						
1	921	260	3.542	2743	90	28
2	1006	290	3.468	1849	96	32
4	1230	370	3.324	1036	111	36
8	1475	480	3.072	633	102	42
16	1687	610	2.765	428	87	55
powerlaw						
0	1657	390	4.248	915	91	24
0.25	1873	450	4.162	669	101	31
0.5	2984	800	3.730	220	88	43
0.75	6027	2100	2.870	124	87	426
1	9460	5500	1.720	81	73	10^4
pow-regular						
0	921	260	3.542	2965	89	28
0.25	1124	325	3.458	1199	122	39
0.5	1858	580	3.203	236	85	40
0.75	4366	1675	2.606	117	91	244
1	7905	4850	1.629	118	96	10^4
double-powerlaw						
β	m	n	m/n	minisat	satz	march
0.75	1325201	$5 \cdot 10^5$	2.650	3.93	344	5121

Table 1: Median time (seconds) for 200 instances at the phase transition point. Timeout of 10000 seconds.

instances. The selected solver was, gnovelty+ [10], the winner of the random category for satisfiable instances at the SAT solver competition. Gnovelty+ was not able to solve any instances with a cutoff of 10000 seconds.

5 Conclusions

We have proposed a generalization of the uniform and the regular k-CNF random generation models, by generalizing the probability distribution used on the selection of variables to a geometric and a powerlaw distribution.

An important result is that all our models guarantee the existence of the phase transition phenomena. We generate instances at the phase transition point, of any given number of variables and computational hardness by adjusting the parameters of the distributions. 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.

We have shown that the instances generated with the powerlaw or pow-regular models present computational properties that are similar to industrial instances. In order to be able to generate bigger instances with variable clause length, we have provided a fifth model, double powerlaw, where we assign a different probability of being chosen to each variable and to each clause. This generates formulas were some variables occur very often and some clauses are very long.

We have conducted an experimental investigation with SAT solvers specialized in industrial instances and others specialized in random instances, to validate that our randomly generated instances have similar computational properties to real-world or industrial instances.

This is a first step in the development of generators for problem instances that have computational properties more similar to real-world (industrial) instances. Further research would require to identify other general properties of industrial instances and adapt our generation models to simulate them.

References

[1] C. Ansótegui, M. L. Bonet, J. Levy, and F. Manyà. Measuring the hardness of sat instances. In *Proc. of the 23th AAAI Conf. on Artificial Intelligence, AAAI'08*, 2008.

[2] R. Bayardo and R. Schrag. Using CSP look-back techniques to solve exceptionally hard SAT instances. In *Proc. of the 2nd Int. Conf. on Principles and Practice of Constraint Programming, CP'96*, pages 46–60, 1996.

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

[4] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003.

[5] O. Dubois and G. Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence, IJCAI'01*, pages 248–253, 2001.

[6] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. of the 6th Int. Conf on Theory and Applications of Satisfiability Testing, SAT'03*, pages 502–518, 2003.

[7] M. J. Heule and H. van Maaren. Whose side are you on? finding solutions in a biased search-tree. *Journal on Satisfiability, Boolean Modeling and Computation*, 4:117–148, 2008.

[8] H. A. Kautz and B. Selman. Ten challenges redux: Recent progress in propositional reasoning and search. In *Proc. of the 9th Int. Conf. on Principles and Practice of Constraint Programming, CP'03*, pages 1–18, 2003.

[9] C. M. 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*, pages 341–355, 1997.

[10] D. N. Pham, J. Thornton, C. Gretton, and A. Sattar. Advances in local search for satisfiability. In *Proc. of the 20th Australian Conf. on Artificial Intelligence*, pages 213–222, 2007.

[11] B. Selman, H. A. Kautz, and D. 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.

[12] R. Williams, C. P. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence, IJCAI'03*, pages 1173–1178, 2003.