Analysis and Generation of Pseudo-Industrial MaxSAT Instances[1]

Carlos ANSÓTEGUI [a] María Luisa BONET [b] Jordi LEVY [c] Chu Min LI [d]
[a] *Universitat de Lleida (DIEI, UdL)*
[b] *Universitat Politècnica de Catalunya (LSI, UPC)*
[c] *Artificial Intelligence Research Institute (IIIA, CSIC)*
[d] *Université de Picardie Jules Verne (MIS)*

Abstract. We propose two random generation models for MaxSAT and Partial MaxSAT in order to produce instances more similar to the industrial benchmarks used in the MaxSAT evaluation. Following the work of [4] and [2], we analyze properties of industrial instances and use a non-uniform (powerlaw) distribution to select the variables.

We also study empirically the optimum (minim number of unsatisfiable clauses) that we obtain with these models, and the relative performance of some MaxSAT solvers. We observe that industrial specialized MaxSAT solvers are better on these random formulas than random specialized solvers. We conclude that instances generated with these new models are more similar to industrial instances than the generated with the classical random models based on uniform probability distributions.

Keywords. Satisfiability, Maximum Satisfiability, Boolean Optimization

1. Introduction

The MaxSAT and Partial MaxSAT problems are the optimization versions of the satisfiability problem. We work typically with unsatisfiable instances, and the idea is to find an assignment that satisfies the maximum number of clauses. In the Partial MaxSAT setting, the instance has two types of clauses: the ones that the assignment must satisfy (hard), and the ones that may or not be satisfied (soft). Then, given a formula with hard and soft clauses, we are looking for an assignment that satisfies all the hard clauses and satisfies a maximum number of soft clauses. For the MaxSAT setting all the clauses are soft. In the SAT community there are mainly two kinds of solvers that can solve MaxSAT and Partial MaxSAT problems: branch and bound solvers, like WMaxSatz(wmsz) [11], MiniMaxSat(mmax) [10], IncWMaxSatz(inc. wmsz) [12], and solvers based on satisfiability testing, like SAT4J [7], wbo and msuncore [13] and WPM1, PM2 [1,3,5]. The latest essentially make use of successive calls to a SAT solver. In general, the branch and bound solvers are more competitive on random problems, while solvers based on calls to a SAT solver are better for industrial or real problems.

[1]This research has been partially founded by the CICYT research projects TASSAT (TIN2010-20967-C04-01/03/04) and ARINF (TIN2009-14704-C03-01).

The MaxSAT and Partial MaxSAT problem are both natural combinatorial problems, that occur in practical areas like: combinatorial auctions, scheduling and timetabling problems, FPGA routing, software package installation, etc. To encourage the work on good algorithms for the MaxSAT and Partial MaxSAT problems, an evaluation (competition) [6] has been organized. For the last four years, a MaxSAT evaluation has taken place associated to the SAT conference. An important part of this competition is to gather enough test instances. As in the SAT competition, the types of instances are: random, crafted and industrial. Of course it is important for a solver to be good in the industrial category. However the number of benchmarks in this category is small. Also, we do not have an instance for every number of variables. And finally, they do not have a parametrized 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.

Following the work of [4], we present a model of generation of random instances that in some aspects matches the properties of the industrial instances. In [4] the generator was defined for SAT instances, and in the present work we use it for MaxSAT and Partial MaxSAT. This time though, there are added difficulties, since the set of industrial instances is smaller. This means that it is impossible to study some parameters of the industrial instances, simply because we don't have a big enough sample. By simply analyzing the quotient m/n (num. clauses / num. variables) we see that it is quite smaller than the value one could expect. This suggest that the frequency of variables is very different, and that the use of non-uniform probability distributions to chose these variables can generate instances more similar to industrial ones. Also, we analyze the optimums found in industrial instances, and they are relatively small.

We have done an extensive experimentation where we study the increment in the optimum with the addition of new clauses, and the dependence of this optimum with the percentage of hard/soft clauses. We have also analyzed the time needed by distinct MaxSAT solvers to solve theses instances, specially by the PM2 and the wmsz solvers, that were two of the fastest solvers in the last MaxSAT evaluation. The analysis includes the performance of the solvers on the classical random instances, to be able to see the effect of changing the variable selection distribution.

The non-uniform random generation models are described in Section 2. In Section 3, we show the results on the dependence of the optimum on the number of clauses and the ratio hard/soft clauses. In Section 4 we extend this study to the performance of distinct solvers, on the classical and the non-uniform models. In Section 5 we make some observations on the industrial instances used as benchmarks in the last MaxSAT evaluation.

2. Description of the Models

We generalize the (uniform) random k-CNF model used in the MaxSAT evaluation to non-uniform distributions, using the ideas of [4]. We will describe two models, that we call powerlaw k-CNF model and double-powerlaw model.

The main idea in both models is to assign a different probability to each variable i of the formula, following a discrete and finite probability distribution $P(X = i; n)$,

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 $P(X = i; n) = 1/n$.

In [4], it is described how we can obtain a family of probability distributions $P(X = i; n)$, with discrete domain $i = \{1, \ldots, n\}$, from a continuous probability distribution ϕ with domain $\big[0, 1\big]$, taking

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

i.e. $P(X = i; n)$ is defined to be $\phi(i/n)$ with the appropriated normalization.

It has been observed that defining the probability distributions this way, allows us to generate formulas that have a phase transition phenomena [4].

Different continuous probability distributions can be used as a basis for the discrete distributions of the models. [2] observed that the powerlaw distribution is the one that best fits an important number of instances of the SAT competition. The powerlaw distribution is obtained from 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 in order to ensure that $\phi(x)$ is defined for all $x \in [0\ 1]$, and the existence of the phase-transition phenomena. 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:

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

Input: n, m, k, β
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**
            Choose a variable v with probability $P(X = v; \beta, n)$
            Choose a sign $s \in \{1, -1\}$ with $P(s) = 1/2$
            $C_i = C_i \vee s \cdot v$;
    until C_i is not a tautology or simplifiable
    $F = F \cup \{C_i\}$
return F

Figure 1. Powerlaw k-CNF generator.

Powerlaw k-CNF formulas may be generated with the algorithm in Figure 1. Notice that, since $\phi^{pow}(x; \beta = 0) = 1$, the powerlaw k-CNF model is a generalization of the uniform k-CNF model. In [4] (Theorem 1) it is proved that with this model we generate formulas where the frequency of occurrences of variables follows a powerlaw distribution with exponent $\alpha = 1/\beta + 1$.

Our second model, the double-powerlaw model, constructs a formula by repeating the following process. It chooses a variable and a clause following two (not necessarily equal) powerlaw distributions, $P(X = v; \beta_v, n)$ for variables, and $P(X = c; \beta_c, m)$ for clauses. Then, the selected variable v is included in the selected clause c, with an arbitrary sign, whenever the clause does not already contain the variable. This process is repeated $k\,m$ times to ensure that the mean size of clauses is k. As for the previous model, in [4] it is proved that this model generates formulas where the frequency of variables follows a powerlaw distribution with exponent $\alpha_v = 1/\beta_v + 1$, and the clauses sizes also follow a powerlaw distribution with exponent $\alpha_c = 1/\beta_c + 1$. The powerlaw distribution, in fact, refers to the *tail* of the distribution. Therefore, we still have some freedom to chose the first values of the distribution. If we use the random generation algorithm as has been described above, we tend to obtain variables with zero occurrences and clauses with very small size (even empty clauses). To avoid this problem, we can fix a minimal number of occurrences for each variable min_v and a minimal size for each clause min_c. Notice that this is compatible with a powerlaw distribution because it only affects the first values (not the tail) of the distribution. The double-powerlaw generation algorithm is described in Figure 2.

These models can be used to generate MaxSAT instances, if m is chosen greater than $c\,n$, where c is the phase transition point. In order to generate a *partial* MaxSAT instance with m_h hard clauses and m_s soft clauses, we generate (using one or the other model) a formula with $m = m_h + m_s$ clauses, and interpret the first m_h clauses as hard clauses, and the rest as soft clauses. In this case, we take $m_h < c\,n < m_h + m_s$ to ensure that the hard subformula is satisfiable, and the whole formula unsatisfiable.

Parameter settings. In all the experiments that follow, we have used either the uniform k-CNF generation model with $k = 3$, the powerlaw k-CNF generation model with $k = 3$ and $\beta = 1$, or the double-powerlaw generation model with $k = 5$, $\beta_v = \beta_c = 0.75$, $min_v = 1$ and $min_c = 2$.

We have decided not to work with 2-CNF partial MaxSAT formulas (although 2-CNF MaxSAT is NP-hard) because industrial instances tend to have larger clauses.

In all the graphics that follow, all the points have been computed generating 100 instances with the corresponding models and different seeds. We compute means of optima and medians of times.[2]

3. Optimum

In this section we study how the minimum number of unsatisfiable clauses depends on the number of variables n, the number of clauses m and the fraction m_h/m_s of hard and soft clauses. We call the minimum number of unsatisfiable clauses, the *optimum* of the formula, and, from now on, we will abbreviate it as o.

[2] We use medians in the case of times because the variability in this case is very big.

Input: $n, m, k, \beta_v, \beta_c, min_v, min_c$
Output: a SAT instance with n variables, m clauses, and clause mean size k
for $i = 1$ **to** m **do** $C_i := \square$; $F := F \cup \{C_i\}$
for $v = 1$ **to** n **do**
  for $i = 1$ **to** min_v **do**
    repeat
      Choose a clause c with probability $P(X = c; \beta_c, m)$
      Choose a sign $s \in \{1, -1\}$ with $P(s) = 1/2$
    until $v \in C_c$
    $C_c := C_c \vee s \cdot v$
for $c = 1$ **to** m **do**
  for $i = 1$ **to** min_c **do**
    repeat
      Choose a variable v with probability $P(X = v; \beta_v, n)$
      Choose a sign $s \in \{1, -1\}$ with $P(s) = 1/2$
    until $v \in C_c$
    $C_c := C_c \vee s \cdot v$;
for $i = 1$ **to** $k * m - min_v * n - min_c * m$ **do**
  repeat
    Choose a variable v with probability $P(X = v; \beta_v, n)$
    Choose a clause c with probability $P(X = c; \beta_c, m)$
    Choose a sign $s \in \{1, -1\}$ with $P(s) = 1/2$
  until $v \in C_c$
  $C_c := C_c \vee s \cdot v$
return F

Figure 2. Double-powerlaw generator.

In the SAT context, the main question is whether a CNF formula is satisfiable or not. For random k-CNF generation models, it is well known the existence of a phase transition phenomena. There exits a value c such that most formulas with $m/n < c$ are satisfiable and most formulas with $m/n > c$ are unsatisfiable. In random generation models with a different probability distribution for variables, phase transition phenomena have also been observed experimentally [4]. In general, in these models the phase transition point c is smaller than in the classical (uniform) models.

In the MaxSAT context, the main question is what is the optimum assignment in terms of the minimal number of clauses that have to be falsified. Therefore, given a random generation model, it is natural to study how the optimum depends on the number of clauses and variables of the formula. This question was theoretically studied in [8] for Max-2-SAT and using the classical uniform distribution. [14] reproduce these results experimentally. They show that, for big values of m/n the optimum increases as

$$0.25m/n - 0.343859\sqrt{m/n} + \mathcal{O}(1) \gtrsim o/n \gtrsim 0.25\, m/n - 0.509833\sqrt{m/n}$$

The reason for the $0.25\, m/n$ term is that, for big values of m/n, a random assignment satisfies almost as many clauses as the optimal assignment; and a random assignment falsifies a randomly chosen clause of size k with probability $1/2^k$. For Max-2-SAT

we have $1/2^2 = 0.25$. Therefore, for k-CNF, we can expect an asymptotic behavior of o/n dominated by $1/2^k\, m/n$.

For values of m/n at the left of the phase transition point c, i.e. $m/n < c$, almost all formulas are satisfiable and the optimum is close zero. For $m/n = c$ half of the formulas are satisfiable, and half unsatisfiable (most of them with optimum one), so the average optimum is close to $\overline{o} \approx 0.5$.

For values of $m/n \gtrsim c$, [8] prove that, for Max-2-SAT and the classical uniform distribution, we have

$$o/n \lesssim 1/3\,(m/n - c)^3$$

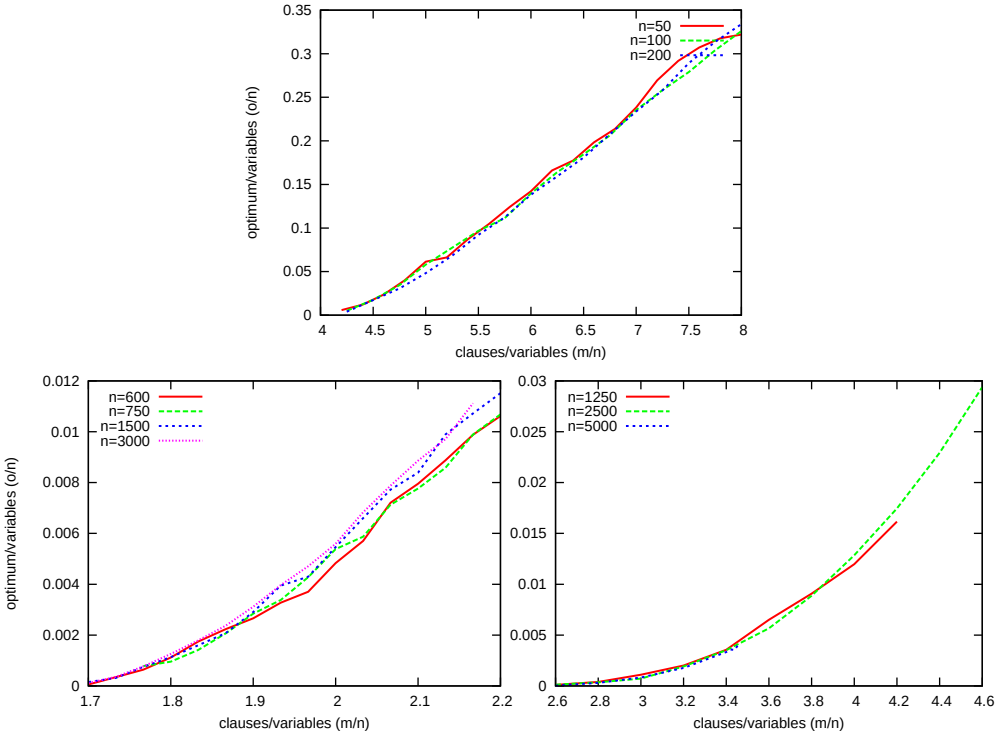

Figure 3. Mean optimum/num. variables (o/n) as a function of num. clauses/num. variables (m/n). **Above:** for the uniform 3-CNF model with $m_h = 4\,n$ hard clauses. **Left:** for the powerlaw 3-CNF model with $m_h = 1.66\,n$ hard clauses. **Right:** for the double-powerlaw model with $m_h = 1.5\,n$ hard clauses.

We have observed experimentally (see Figure 3) that o/n grows with m/n with a rate smaller than $1/2^k$. This growth rate increases with m/n, i.e. the dependence of o/n with m/n defines a convex function. This is compatible with the theoretical results proved for classical Max-2-SAT.

A natural question is how the fraction of hard/soft clauses affects the value of the optimum. Suppose that we have two partial MaxSAT problems with identical sets of clauses but one with higher percentage of hard clauses

$$F_1 = \{(\infty, C_1), \ldots, (\infty, C_{m_h}), (\infty, C_{m_h+1}), \ldots, (\infty, C_{m'_h}), (1, C_{m'_h+1}), \ldots, (1, C_m)\}$$
$$F_2 = \{(\infty, C_1), \ldots, (\infty, C_{m_h}), (1, C_{m_h+1}), \ldots, (1, C_{m'_h}), (1, C_{m'_h+1}), \ldots, (1, C_m)\}$$

where we indicate hard clauses with a *weight* ∞, and soft clauses with a weight 1.

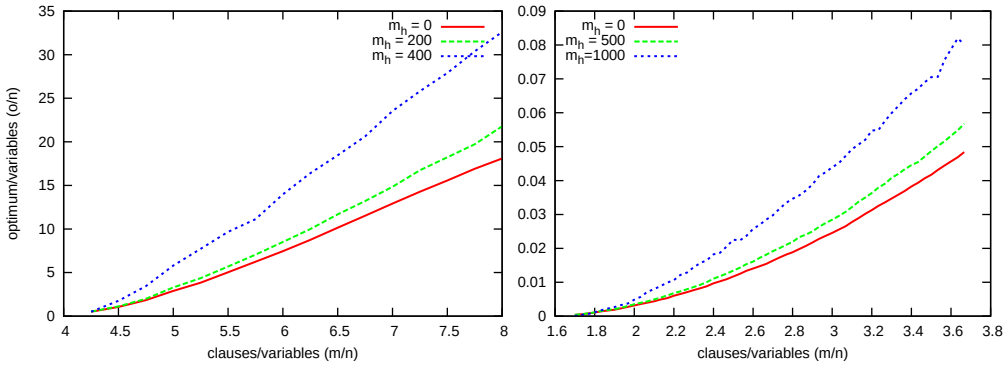

Figure 4. Mean optimum/num. variables (o/n) as a function of num. clauses/num. variables (m/n), with distinct number of hard clauses **Left:** for the uniform 3-CNF model. **Right:** for the powerlaw 3-CNF model.

An optimal assignment for F_1 will be a possible assignment for F_2, in the sense that it satisfies all hard clauses of F_2. This assignment will satisfy the same number of soft clauses of both formulas. Therefore, the minimum number of unsatisfiable soft clauses of F_2 (optimum of F_2) is less or equal than the optimum of F_1. The opposite is not true. In general, given two random partial MaxSAT formulas, with the same number of variables and total number of clauses, but different number of hard clauses, the one with a smaller number of hard clauses will have a smaller optimum. This is experimentally shown in Figure 4. On the other hand, for the previous example, we can expect that solving F_2 takes a longer time than solving F_1. This is because for F_2 we have to check a bigger number of possible assignments than for F_1.

4. Time

The MaxSAT solvers that participate in the MaxSAT evaluation [6] can be divided in two types: solvers based on a branch-and-bound schema, like wmsz [11], inc. wmsz [12] and mmax [10]; and those based on successive calls to a SAT solver, like WPM1 [3], PM2 [3], wbo [13], msuncore [13] and SAT4J [7]. With some exceptions, in the MaxSAT evaluation it has been observed that solvers based on branch-and-bound tend to perform better in the category of random formulas, and that the SAT-based solvers tend to be better in the industrial category. We have conducted a series of experiments with all these solvers and formulas generated with the uniform 3-CNF model, the powerlaw 3-CNF model and the double-powerlaw model.

Our experiments have been run on machines with the following specs; Operating System: Rocks Cluster 4.0.0 Linux 2.6.9, Processor: AMD Opteron 248 Processor, 2 GHz and compilers, Memory: 0.5 GB and Compilers GCC 3.4.3, javac JDK 1.5.0.

Figure 5 shows the results. The graphics start with values of m/n very close to the phase transition point, where the optimum is approximately zero, and finish for values

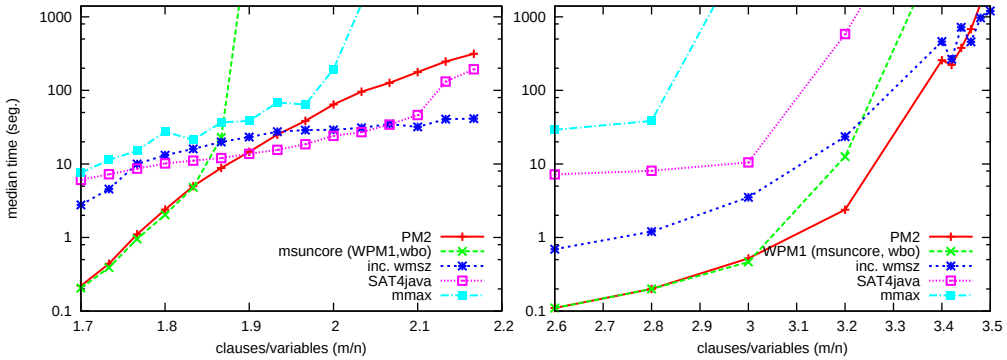

Figure 5. Median time as a function of number clauses/number variables, for distinct solvers. **Left:** for the powerlaw 3-CNF model with $n = 3000$, $m_h = 5000$ and $m_s = 100, 200, \ldots, 1500$. **Right:** for the double-powerlaw model with $n = 5000$, $m_h = 7500$ and $m_s = 5500, 6500, \ldots, 10500$.

of m/n farther away from the phase transition point, where most of the executions overpass a time cutoff of $1200s$. Observing Figure 3, this cutoff corresponds more or less to formulas with an optimum close to 30, for both models.

We have observed that WPM1, wbo and msuncore, have an almost identical behavior, because all three are implementations of the Fu&Malik [9] algorithm (with small differences). Therefore, we have decided to show the data for only one of them (the faster in each case). We also observe that wmsz and inc. wmsz have also almost identical behavior. Therefore, in this case we have decided to show the data only for the wmsz.

As a general trend, we observe that SAT-based solvers are more sensitive to the value of the optimum than branch-and-bound solvers. Thus, whereas for optimums close to zero SAT-based solvers (except SAT4J) are 10 times faster than branch-and-bound solvers, when the optimum is close to 30 the situation reverses, and branch-and-bound solvers outperform SAT-based solvers. For the double-powerlaw model, the situation is similar. However, in this case although SAT-based solvers are again up to 10 times faster than branch-and-bound solvers in the area close to the phase transition point, when the optimum is close to 30, the times for both kinds of solvers are comparable and close to the 1200s cutoff. Observe also that, for points close to the phase transition point, the three solvers implementing versions of the Fu&Malik algorithm, and the PM2 have the same performance. But they diverge for points with bigger optima.

A natural question is if this pattern in the relative performance of the PM2 and the wmsz solvers is also observed for other values of n. Also, we have to show the differences between the uniform and non-uniform generation models looking at how the different solvers perform on them. In Figure 6 we show the results, only for PM2 and wmsz solvers and different values of n. The following table shows the optima at the points where both solvers have the same performance.

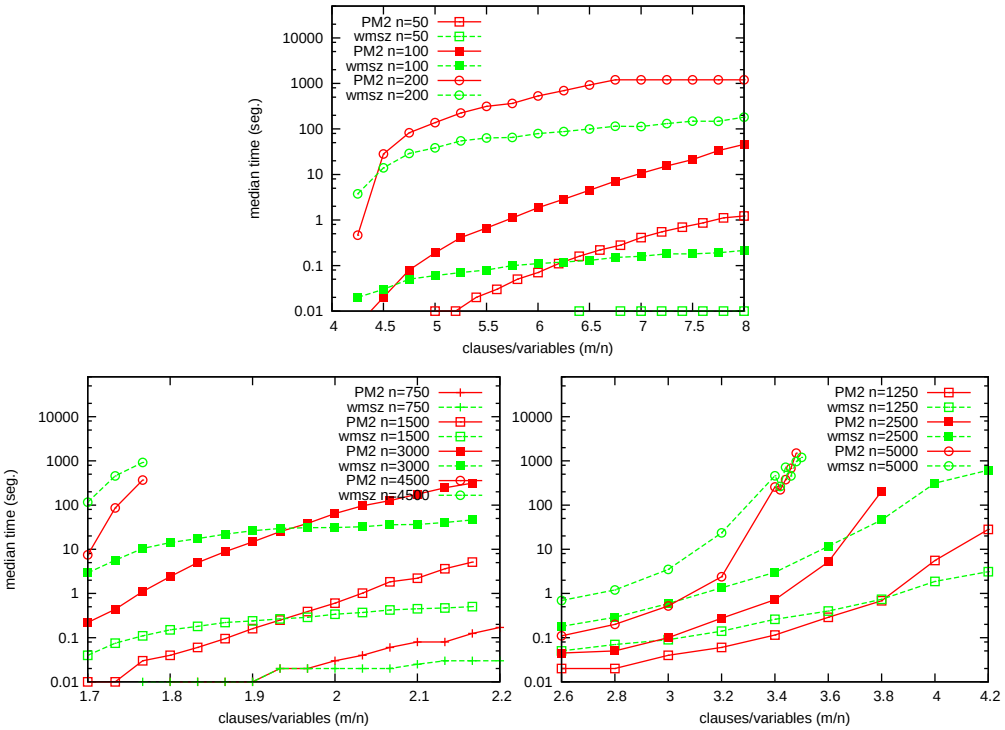

Figure 6. Mean time as a function of the number of variables and of the clause/variable ratio, for the PM2 and the wmsz solvers. **Above:** for the uniform 3-CNF model with $m_h = 4\,n$. **Left:** for the powerlaw 3-CNF model with $m_h = 1.66\,n$. **Right:** for the double-powerlaw model with $m_h = 1.5\,n$.

		m/n	o
Uniform	$n = 100$	4.61	2.4
	$n = 200$	4.43	2.4
Powerlaw	$n = 1500$	1.95	6
	$n = 3000$	1.94	12
Double-powerlaw	$n = 1250$	3.8	11
	$n = 2500$	3.7	17
	$n = 5000$	3.4	18

The reason for this behavior is the following. The PM2 and other SAT-based solvers make as many calls to the underlying SAT solver as the value of the optimum. Therefore, the bigger the optimum is, the longer the solvers needs to run. In the case of branch-and-bound solvers, they are not influenced by the value of the optimum. On the other hand, they only need a bit longer time for bigger values of m because they have to update more clauses. SAT-based solvers use SAT solvers very competitive in dealing with big (industrial) formulas. This makes also them competitive dealing with big industrial MaxSAT formulas, when the optimum is small. However, when the formulas have a uniform distribution of variable frequencies, like in the uniform model, the performance of the underlying SAT solver is poor and the SAT-based solvers are outperformed by branch-and-

bound solvers, even for small values of the optimum. Notice that industrial-specialized SAT solvers are also good dealing with random pseudo-industrial instances [4].

It is interesting to notice that SAT4J, even been based on a schema of successive calls to a SAT solver, shows a behavior more similar to branch-and-bound solvers like wmsz than to other SAT-based solvers like PM2. The reason for this is that SAT4J does not make as many calls to the SAT solver as the value of the optimum. On the contrary, it starts with an upper bound for the optimum and decreases this upper bound each time that the SAT solver returns satisfiable.

5. Industrial Benchmarks

The set of industrial instances in the MaxSAT evaluation is relatively small. Therefore, it is difficult to conclude if there exist a particular distribution in the frequencies of variables and clause sizes. All we can say is that the variability is big in most instances.

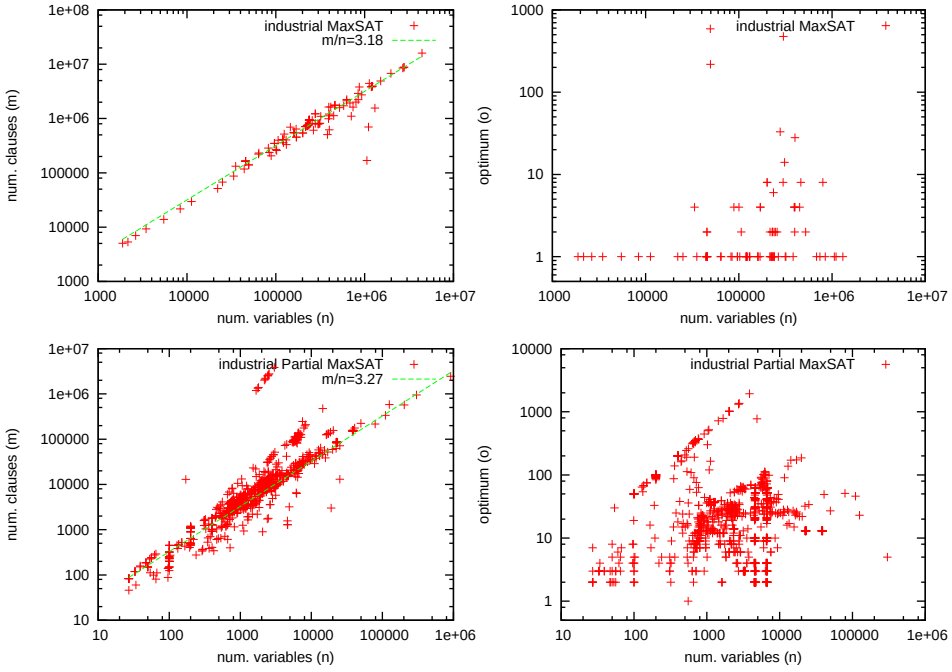

Figure 7. Plot of the relations num. variables — num. clauses ($n-m$) and num. variables — optimum ($n-o$) for the industrial MaxSAT and Partial MaxSAT benchmarks used in the MaxSAT evaluation.

In Figure 7, we show the relation between n and m in these industrial instances. It is remarkable that the quotient $m/n \approx 3.2$ is quite small. Notice that the average industrial clause size is 19, thus bigger than the size 3 of random 3-CNF formulas. However, most uniform 3-CNF formulas with $m/n < 4.25$ are satisfiable. The phase transition point of our non-uniform models is studied in [4]. There, it is shown that this point, for the powerlaw 3-CNF model, is smaller than the phase transition point of the uniform 3-CNF model. The same is observed for other models based in other distributions, like

the geometric distribution, and for the double-powerlaw model. We think that the higher the variance of the distribution is, the smaller the phase transition point is. This would explain why we can have a so small m/n ratio in an unsatisfiable formula with big clauses, if the variability in the frequency of variables and clause sizes is big.

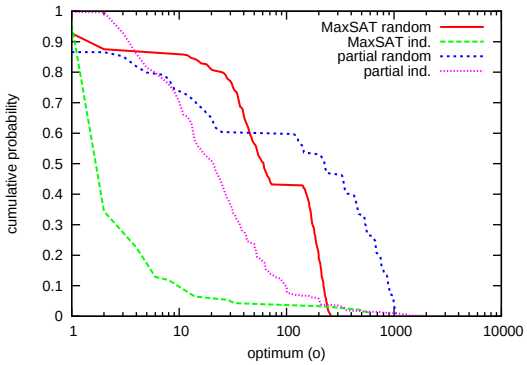

Figure 8. Experimental cumulative distributions for the optimum of distinct families of benchmarks.

We also study the value of the optima of the industrial instances. In Figure 7, we observe that these optima are obviously smaller than n, but it is difficult to observe any other correlation. There is not a correlation between o and m/n, either. In Figure 8, we observe that the optima values are typically around 20 (median) for the industrial partial MaxSAT instances, and around 1.5 for the industrial MaxSAT instances. Comparing these values with the optima obtained for the random formulas used in the random category of the MaxSAT evaluation, we observe that they are smaller.

6. Conclusions

We have studied two random generation models (powerlaw k-CNF and double-powerlaw) for the MaxSAT and Partial MaxSAT problems in order to produce instances more similar to the industrial benchmarks used in the MaxSAT evaluation. We have conducted an experimental investigation with the best performing branch-and-bound and SAT-based Partial MaxSAT solvers. The study shows that SAT-based solvers, that typically are the best at the industrial categories in the MaxSAT evaluation, can be better on these non-classical random formulas than branch-and-bound solvers, which typically are the best for the random categories at the MaxSAT evaluation.

This study will also allow us to fix the parameters of our models (n, m, m_h, m_s, k, o, β_v and β_m) to generate pseudo-industrial instances.

As a future work we plan to extend these generation models to produce Weighted MaxSAT and Weighted Partial MaxSAT instances more similar to the industrial ones.

References

[1] C. Ansótegui, M. L. Bonet, J. Gabàs, and J. Levy. Improving SAT-based Weighted MaxSAT solvers. In *Proc. of the 18th Int. Conf. on Principles and Practice of Constraint Programming (CP'07)*, 2012.

[2] C. Ansótegui, M. L. Bonet, and J. Levy. On the structure of industrial SAT instances. In *Proc. of the 15th Int. Conf. on Principles and Practice of Constraint Programming, CP'09*, pages 127–141, 2009.

[3] C. Ansótegui, M. L. Bonet, and J. Levy. Solving (weighted) partial MaxSAT through satisfiability testing. In *Proc. of the 12th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 427–440, 2009.

[4] C. Ansótegui, M. L. Bonet, and J. Levy. Towards industrial-like random SAT instances. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI'09*, pages 387–392, 2009.

[5] C. Ansótegui, M. L. Bonet, and J. Levy. A new algorithm for Weighted Partial MaxSAT. In *Proc. the 24th National Conference on Artificial Intelligence (AAAI'08)*, 2010.

[6] J. Argelich, C. M. Li, F. Manyà, and J. Planes. The first and second Max-SAT evaluations. *Journal on Satisfiability*, 4:251–278, 2008.

[7] D. L. Berre. SAT4JMaxSat. In *www.sat4j.org*.

[8] D. Coppersmith, D. Gamarnik, M. T. Hajiaghayi, and G. B. Sorkin. Random MAX SAT, random MAX CUT, and their phase transitions. In *Proc. of the 14th ACM-SIAM Symposium on Discrete Algorithms, SODA'03*, pages 364–373, 2003.

[9] Z. Fu and S. Malik. On solving the partial MAX-SAT problem. In *Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 252–265, 2006.

[10] F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: A new weighted Max-SAT solver. In *Proc. of the 10th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'07)*, pages 41–55, 2007.

[11] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Exploiting cycle structures in Max-SAT. In *Proc. of the 12th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'09)*, 2009.

[12] H. Lin, K. Su, and C. M. Li. Within-problem learning for efficient lower bound computation in Max-SAT solving. In *Proc. the 23th National Conference on Artificial Intelligence (AAAI'08)*, pages 351–356, 2008.

[13] V. M. Manquinho, J. P. M. Silva, and J. Planes. Algorithms for weighted boolean optimization. In *Proc. of the 12th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 495–508, 2009.

[14] H. Shen and H. Zhang. An empirical study of MAX-2-SAT phase transitions. *Electronic Notes in Discrete Mathematics*, 16:80–92, 2003.