A Modularity-based Random SAT Instances Generator *

Jesús Giráldez-Cru and **Jordi Levy**
IIIA - CSIC
Campus UAB, Bellaterra, Spain
{jgiraldez,levy}@iiia.csic.es

Abstract

Nowadays, many industrial SAT instances can be solved efficiently by modern SAT solvers. However, the number of real-world instances is finite. Therefore, the process of development and test of SAT solving techniques can benefit of new models of random formulas that capture more realistically the features of real-world problems. In many works, the structure of industrial instances has been analyzed representing them as graphs and studying some of their properties, like modularity.

In this paper, we use modularity, or community structure, to define a new model of pseudo-industrial random SAT instances, called *Community Attachment*. We prove that the phase transition point, if exists, is independent on the modularity. We evaluate the adequacy of this model to real industrial problems in terms of SAT solvers performance, and show that modern solvers do actually exploit this community structure.

1 Introduction

The Boolean Satisfiability Problem (SAT) is one of the most studied problems in Computer Science. It was the first known NP-complete problem. However, many application problems (such as cryptography, hardware and software verification, planning, scheduling, among others) may be encoded into SAT, and efficiently solved by modern SAT solvers.

It is accepted that random k-CNF and industrial SAT instances have a distinct nature. While random formulas can be easily generated on demand, the set of industrial instances, which encode real-world problems, is limited. The problem of generating realistic pseudo-industrial random instances is stated in [Selman *et al.*, 1997; Kautz and Selman, 2003; Dechter, 2003] as one of the most important *challenges* for the next few years. The main motivation of this challenge is improving the process of development and test of SAT solvers, and their possible specialization.

There exist a wide variety of works on the analysis of the nature of industrial SAT instances. The intuition is that these formulas have some kind of *structure*, which is exploited by SAT solvers. In many of these works, SAT formulas are represented as graphs, and some (graph) features are studied. The classical Erdös-Rényi model has been extensively used for generating random graphs. In this model, the arity of nodes follows a binomial distribution, with small variability. This is exactly the case of random k-CNF formulas. However, the structure of most real-world problems cannot be described with this classical model, and therefore, new models have been defined. For instance, Preferential Attachment [Barabási and Albert, 1999] is used to explain the scale-free structure of some real networks, where the arity of nodes follows a power-law distribution, with big variability. In the context of SAT, some notions of structure have also been studied, such as the small-world property [Walsh, 1999], the scale-free structure [Ansótegui *et al.*, 2009a], or the centrality [Katsirelos and Simon, 2012], or the self-similarity [Ansótegui *et al.*, 2014], among others.

There also exist many methods for generating industrial-like random instances. In [Slater, 2002], it is described a method based on the *characteristic path length* and *clustering coefficient*. In [Burg *et al.*, 2012], it is presented a generator that combines subparts of real-world instances to create new ones. In [Järvisalo *et al.*, 2012], it is proposed an instance generator based on finding optimal circuits of Boolean functions. In [Ansótegui *et al.*, 2009b], it is used the notion of scale-free graph to generate formulas where the number of variable occurrences follows a power-law distribution.

The inspiration of this work is [Ansótegui *et al.*, 2012]. In that paper, it is stated that industrial SAT formulas exhibit a clear community structure (i.e. high modularity Q). This means that, representing formulas as graphs, we can find a partition of the formula into communities with many edges between nodes of the same community, and few edges connecting distinct communities. This property is very characteristic in real-world problems in contrast to randomly generated instances, where modularity is very low. In the context of SAT, it has been shown that the community structure is correlated with the runtime of CDCL SAT solvers, and the modularity is the best predictor of this runtime to date [Newsham *et al.*, 2014]. Moreover, it has also been used to improve the performance of some solvers [Martins *et al.*, 2013; Sonobe *et al.*, 2014]. To the best of our knowledge, there exists no SAT instance generator considering the community

*This work is partially supported by the CSIC project 201450E045.

structure.

In this paper, we present a new model of generation of random CNF problems based on the community structure[1]. With this new model, we can generate formulas for any given value of modularity. For a high modularity, the resulting formula is more adequate to model industrial problems than classical random k-CNF. This generator allows us to *isolate* the effect of modularity on the performance and behavior of SAT solvers. For instance, we show that CDCL solvers concentrate their decisions on variables of the same (or few) module along time.

The rest of the paper proceeds as follows. After some preliminaries on modularity of graphs in Section 2, we describe the generation model in Section 3. In Section 4, we show that this model works appropriately for different input values of number of variables n and clauses m. In Section 5, we show that, if the formulas exhibit a phase transition region SAT-UNSAT when the ratio clause/variable is increased, then it does not depend on the modularity. Finally, in Section 6, we give empirical evidence that the performance of SAT solvers is consistent with the nature of the generated formulas, i.e. SAT solvers *specialized* in industrial formulas work better in high modular instances than SAT solvers *specialized* in random k-CNF, and vice versa.

2 Preliminaries

SAT is the problem of deciding if the variables of a propositional formula can be assigned in such a way that the formula is evaluated as `true`. In this context, some concepts are natural: a *literal* is either a variable or its negation, a *clause* is a disjunction of literals, a conjunctive normal form (*CNF*) formula is a conjunction of clauses, and a *k-CNF* is a CNF in which all clauses have exactly k literals.

An undirected weighted graph G is a pair $G = (V, w)$, where V is the set of nodes, and $w : V \times V \to \mathbb{R}^+$ is the edge-weight function that satisfies $w(x, y) = w(y, x)$.

The Variable Incidence Graph (VIG) of a SAT instance Γ is the graph whose nodes represent the variables of Γ, and there exists an edge between two variables if they both appear in a clause cl. A clause with l literals results into $\binom{l}{2}$ edges. Thus, to give the same relevance to all clauses, edges have a weight $w(x, y) = \sum_{\substack{cl\in\Gamma \\ x,y\in cl}} 1/\binom{|cl|}{2}$, where $|cl| = l$ is the length of the clause cl.

The *community structure* of a graph is usually measured using the notion of *modularity* [Newman and Girvan, 2004]. Defined for a graph G and a partition C of its vertexes into communities, the modularity Q (see Eq. 1) measures the fraction of internal edges (edges connecting vertexes of the same community) w.r.t. a random graph with same number of vertexes and same degree. This avoids that the best partition is the one made up by an only community containing all vertexes.

[1]This modularity-based SAT generator is available in `http://www.iiia.csic.es/~jgiraldez/software`

$$Q(G, C) = \sum_{C_i \in C} \frac{\sum_{x,y \in C_i} w(x, y)}{\sum_{x,y \in V} w(x, y)} - \left(\frac{\sum_{x \in C_i} deg(x))}{\sum_{x \in V} deg(x)} \right)^2 \quad (1)$$

The modularity of a graph is the maximal modularity for any possible partition: $Q(G) = \max\{Q(G, C) \mid C\}$.

Computing the modularity of a graph is NP-hard [Brandes *et al.*, 2008]. Due to its complexity, instead of computing the (exact) modularity, most of methods in the literature approximate a lower-bound in the value of Q.

3 The Model

In the classical random k-SAT model, a random formula $F_k(n, m)$ is a set of m clauses over n variables, where clauses are chosen uniformly and independently among all $2^k\binom{n}{k}$ non-trivial clauses of length k.[2] In this paper we present a new model of random formulas: the **Community Attachment** (**CA**) model. This is parametric in a probability P and a partition C of the set of variables, and it allows the generation of highly modular formulas.

Definition 1 Community Attachment. *Let N be a set of n variables, a partition C of N into c communities of the same size $s = n/c$, with $k \leq c \leq n/k$, and a real value $0 \leq P \leq 1$. A random formula $F_k(n, m, c, P)$ is a set of m non-trivial clauses with k literals over the n variables, selected independently as follows. With probability P, choose a clause uniformly among all $c\,2^k\binom{n/c}{k}$ clauses with all literals in the same community; and with probability $1 - P$, a clause uniformly among all $\frac{n^k 2^k}{c^k}\binom{c}{k}$ clauses with all literals in distinct communities.*

Notice that in the previous definition we need to impose the restriction $k \leq c \leq n/k$ to ensure that there always exists at least one possible clause to select. Notice also that for $k = 2$ and $P = \frac{n/c-1}{n-1}$ we have the classical 2-SAT model: $F_2(n, m) = F_2(n, m, c, \frac{n/c-1}{n-1})$, but for bigger k the community attachment model does not subsume the classical model.

Given a SAT formula Γ with n variables and m clauses, consider the VIG G of Γ. Our model ensures a lower-bound for the modularity of this graph.

Theorem 2 *Given a formula $\Gamma \in F_k(n, m, c, P)$, let G be its VIG. The average modularity of G is bounded as:*

$$E[Q] \geq P - \frac{1}{c}$$

PROOF: Recall that modularity is defined as the maximal modularity for all possible partitions of the nodes into communities. Here we consider the partition used to generate the formula. For this particular partition, when we select a clause with all variables in the same community (with probability P), we get $\binom{k}{2}$ internal edges. The sum of the weights of the

[2]A non-trivial clause of length k contains k distinct, non-complementary literals.

Algorithm 1: Community Attachment

Input: **int** n, m, c, k; **real** Q;
Output: k-CNF **SAT Instance** Γ

$\Gamma := \emptyset$;
$P := Q + 1/c$;
for $j \in \{1, \ldots, m\}$ **do**
  if $rand([0,1]) \leq P$ **then** `// same community`
    $r := rand(\{1, \ldots, c\})$;
    for $i \in \{1, \ldots, k\}$ **do**
      $C_i := r$;
  else `// distinct communities`
    for $i \in \{1, \ldots, k\}$ **do**
      repeat $C_i := rand(\{1, \ldots, c\})$;
      until $\forall i' < i (C_{i'} \neq C_i)$;
  repeat
    $Cl := \emptyset$;
    for $i \in \{1, \ldots, k\}$ **do**
      $X_i := rand(\{\lfloor (C_i - 1)\frac{n}{c} \rfloor + 1, \ldots, \lfloor C_i \frac{n}{c} \rfloor\})$;
      $Cl := Cl \bigvee rand(\{-1, 1\}) \cdot X_i$;
  until $\forall i' < i (X_{i'} \neq X_i)$;
  $\Gamma := \Gamma \bigwedge Cl$;
return Γ;

edges generated by a single clause is always 1. Therefore, the fraction of internal edges is, on average, $\frac{P\,m}{m}$. The sum of nodes degrees is $2m$, thus $2m/n$ is the expected node degree. Since n/c is the number of nodes per community, the sum of nodes degrees in one community is $\frac{n}{c}\frac{2m}{n}$.

Summarizing, for this partition C, we get

$$E[Q(G,C)] = \frac{Pm}{m} - c\left(\frac{\frac{n}{c}\frac{2m}{n}}{n\frac{2m}{n}}\right)^2 = P - \frac{1}{c}$$

that is a lower-bound for the modularity. ■

When P is big enough, the modularity is very close to this lower-bound, because the partition used in the formula generation is highly modular. Therefore, we can use the previous theorem to generate formulas with a desired modularity Q. We simply take:

$$P = Q + \frac{1}{c} \tag{2}$$

which ensures at least a modularity Q. In practice, as we will see in Section 4, the formulas we obtain have a modularity $Q \approx P - \frac{1}{c}$, except when P and m/n are small.

In Alg. 1, it is described in detail an implementation of a generator of community attachment random formulas from $F_k(n, m, c, P)$. Using $P = Q + \frac{1}{c}$ these formulas will have an expected modularity close to Q.

4 Validation of the Model

In order to analyze the community structure of the SAT instances obtained with our model, we have generated some sets of random formulas for different values of $Q \in \{0.9, 0.8, 0.7, 0.5, 0.3\}$ (hence $P = Q + 1/c$). Remark that the modularity Q of (real) industrial SAT instances is usually greater than 0.7, while no modularity greater than 0.3 is found for classical random k-CNF formulas. Moreover, the number of communities c is usually in $(10, 100)$ [Ansótegui *et al.*, 2012]. In Fig. 1, we analyze their modularity Q and their number of communities c, varying the number of variables n and the clause/variable ratio m/n (we fix $m/n = 4$ and $n = 1000$ respectively). We use the algorithm described in [Ansótegui *et al.*, 2012] to compute an approximation for Q and c. In fact, this algorithm computes a lower-bound for Q. The dispersions of the approximated Q and c are very small, so they are not shown in the plots.

We observe that the modularity Q and the number of communities c are almost unaffected by these variations of n and m/n. In general, the approximation computed for Q is slightly smaller than expected, and the partition into communities is also very similar to the partition used in the generation. Recall that the algorithm used to approximate Q returns a lower-bound of it. For small values of m/n and P, the number of clauses relating variables of the same expected community is very small. This produces the existence of some unconnected sub-communities within each expected community. Hence, c and Q are much greater than expected, and Q cannot be estimated as $P - 1/c$. When we generate formulas with small values of P, e.g. $Q = 0.3$ and $c = 40$, we observe that, although the formulas have a guaranteed lower-bound of $Q \geq 0.3$, the computed approximation of Q is smaller (close to 0.2 when $n \approx 20000$). The number of communities is also smaller than expected. In this case, this error is not produced by our model. It is due to the greediness of the algorithm used to approximate Q, which is not able to find a similar partition to the one used in the generation.

5 Phase Transition

In classical random k-CNF instances, some interesting properties, as the satisfiability or the hardness, are correlated to the clause/variable ratio m/n [Mitchell *et al.*, 1992]. The Satisfiability Threshold Conjecture, which remains open, suggests that it may exist a critical ratio r, such that below this point all formulas are SAT (under-constrained) and above it they are UNSAT (over-constrained) with uniformly positive probability, when n tends to infinity. Experimentally, this phase transition point has been shown to be around $r \approx 4.26$ for $k = 3$. Moreover, the hardness of these instances is also characterized by this parameter: closer to this ratio, harder the instance.

In this section we check if this phenomenon also exists in the random SAT instances generated with our model, and if the new transition point, noted r', differs from the classical $r' \neq r$. In Fig. 2, we represent the fraction of UNSAT instances for some sets of random formulas with distinct Q, varying the clause/variable ratio m/n. We observe that the fraction of UNSAT formulas increases with m/n. Therefore, for small (big) values of m/n, nearly all formulas are SAT (UNSAT). When Q is small, the value r' is close to the classical $r \approx 4.26$. Recall that when $P \approx 1/c$, our model is quite

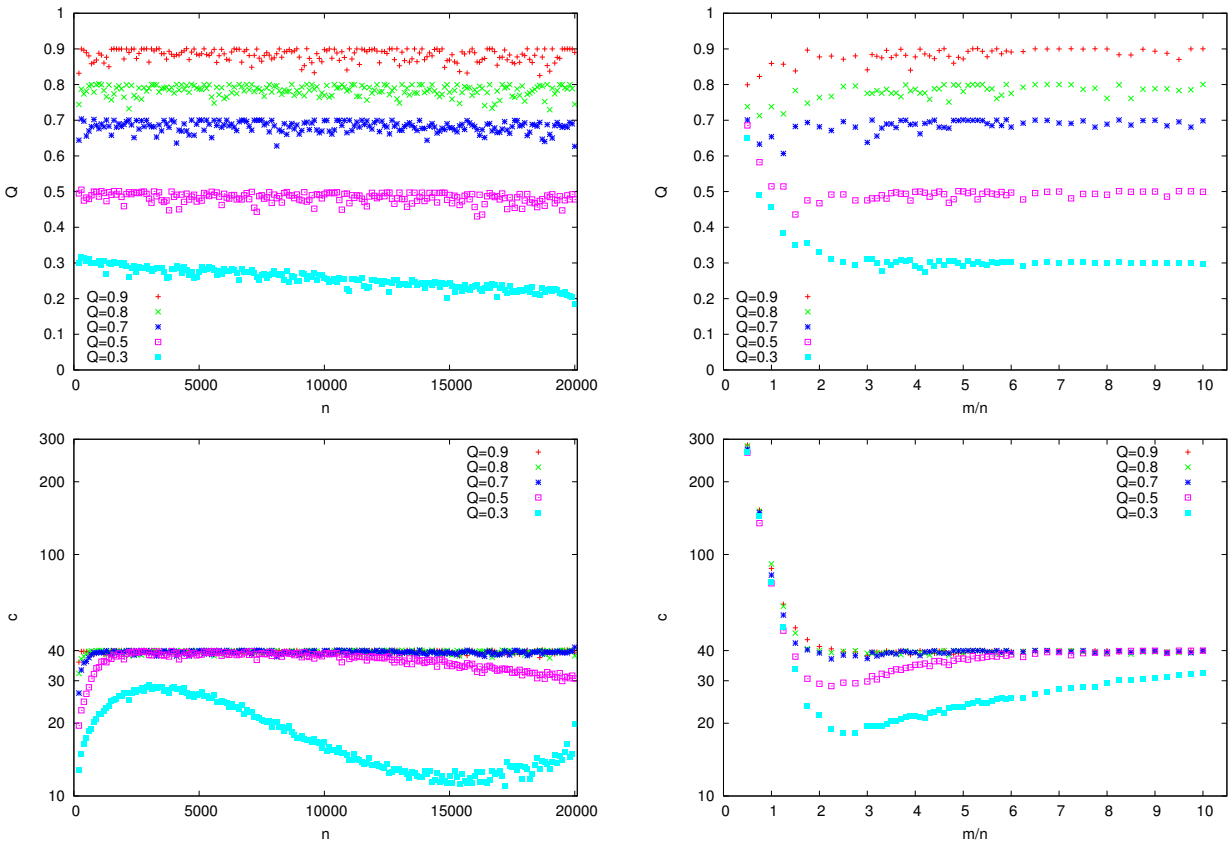

Figure 1: Approximations of modularity Q (top) and no. of communities c (bottom) of some sets of random SAT formulas, varying the number of variables n with $m/n = 4$ (left), and varying the clause/variable ratio m/n with $n = 1000$ (right). Each plotted data is the mean of 50 instances. The input number of communities c is fixed to 40.

Q	r'	n	solver	$\bar{R}$	$S\,[R]$
0.9	**4.06**	5000	Glucose	80.86	125.28
0.8	**4.11**	2000	Glucose	291.87	1217.23
0.7	**4.13**	1200	Glucose	211.64	791.76
0.5	**4.18**	600	March	544.19	1051.81
0.3	**4.24**	600	March	3492.36	3117.23

Table 1: Phase transition point r' of some sets of 200 random SAT instances with $k = 3$, $c = 40$ and varying Q. We also report the number of variables n, the solver and runtime R (mean $\bar{R}$ and standard deviation $S\,[R]$) needed to solve them.

similar to the classical random k-SAT model. However, we also observe that, when Q increases, r' decreases. The natural question is if this decrease in r' is also valid for n tending to infinity. In our experimentation we use the biggest value of n allowing us to get a solution in less than 3 hours (see Table 1).

In order to explain this decrease in the phase transition point r', and predict the behavior when n tends to infinity, we will consider the extreme case with $P = 1$. In these formulas, clauses only contain variables of the same community. Therefore, the formula is composed by c unconnected sub-formulas, and the whole formula is UNSAT if, and only if, at least one of the sub-formulas is UNSAT. Moreover, in this extreme case, all sub-formulas follow the classical model $F_k(n/c, m')$, for some m'. On average, all sub-formulas contain $E[m'] = m/c$ clauses; and all of them contain $s = n/c$ variables. Hence, the average clause/variable fraction in sub-formulas is also $E[m'/s] = \frac{m/c}{n/c} = m/n$. However, even when the fraction m/n is smaller than the classical r (and so the expected clause/variable ratio of the formula), with some probability, some of the sub-formulas may get a large portion of clauses m' such that $\frac{m'}{n/c} > r$. This makes that sub-formula UNSAT with high probability. This has the effect of decreasing the phase transition point for finite n and c.

When n/c tends to infinity, the situation is completely different as the following theorem states.

Theorem 3 *The set of formulas $F_k(n, m, c, P)$ with $P = 1$ has a phase transition point r' at the same clause/variable ratio r of the classical $F_k(n, m)$.*

PROOF: Let $r' = m/n$ and r be the classical phase transition point. The minimal r' such that the probability that some of the sub-formulas has more than $r\,s$ clauses (hence it is UNSAT with high probability when s tends to infinity), will be

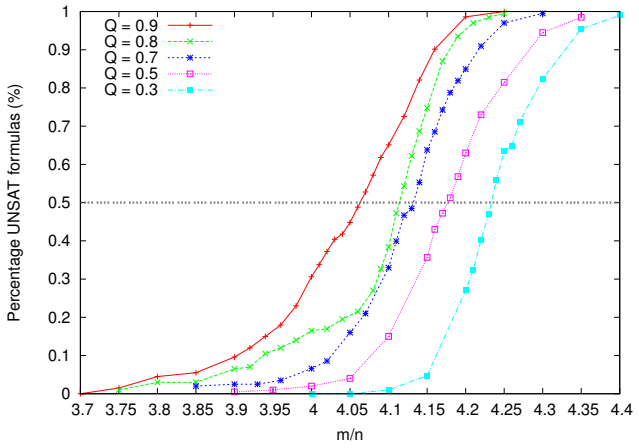

Figure 2: Fraction of UNSAT formulas for some sets of 200 random SAT formulas with $k = 3$, $c = 40$, and varying the clause/variable ratio m/n (see the number of variables n of each family in Table 1).

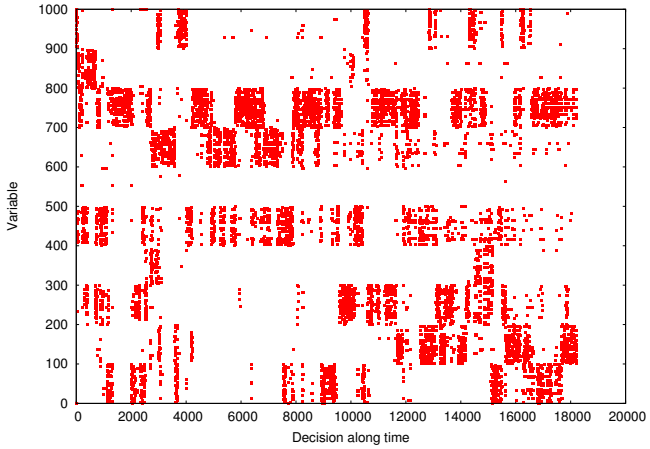

Figure 3: Decided variables along the execution of MiniSat on a random instance with $n = 1000$, $m = 4200$, $k = 3$, $c = 10$ and $Q = 0.8$.

the phase transition point for this special case $P = 1$ of our model.

The probability that a given community C_i contains $r\,s$ clauses when the formulas has m clauses is

$$P(C_i \text{ is UNSAT}) = \frac{\binom{m}{r\,s}\,(c-1)^{m-r\,s}}{c^m}$$

Suppose that the formula has $m = r'\,n$ clauses. Recall also $n = c \cdot s$. We also assume that both the number of communities c and their size s tend to ∞.

When $m, n \to \infty$, and $m/n \to \infty$, the binomials $\binom{m}{n}$, may be approximated as:

$$\begin{aligned}\binom{m}{n} &= \frac{m\cdot(m-1)\cdots(m-n+1)}{n!} \approx \frac{(m-n/2)^n}{\sqrt{2\pi n}(n/e)^n} = \\ &= \frac{m^n\left(1-\frac{1}{2m/n}\right)^{\frac{2m}{n}\cdot\frac{n^2}{2m}}}{\sqrt{2\pi n}(n/e)^n} \approx \frac{m^n(1/e)^{n^2/2m}}{\sqrt{2\pi n}(n/e)^n} = \\ &= \frac{1}{\sqrt{2\pi n}}\left(\frac{m\,e}{n\,e^{n/2m}}\right)^n\end{aligned}$$

using the middle value in the numerator, and the Stirling approximation in the denominator.

When $c \to \infty$, we can also approximate

$$(c-1)^{m-r\,s} = c^{m-r\,s}\,(1-1/c)^{c\frac{m-rs}{c}} \approx \frac{c^{m-r\,s}}{e^{\frac{m-rs}{c}}}$$

Replacing these two approximations, and $m = r'\,c\,s$ we get

$$P(C_i \text{ UNSAT}) \approx \frac{1}{\sqrt{2\pi rs}}\left(\frac{r'}{r}\exp\left(1-\frac{r'}{r}+\frac{1}{c}(1-\frac{r}{2r'})\right)\right)^{rs}$$

For $s, c \to \infty$, this function is dominated by the exponential factor

$$P(C_i \text{ UNSAT}) = \mathcal{O}\left(\left(\frac{r'}{r}\exp\left(1-\frac{r'}{r}\right)\right)^{rs}\right)$$

The base of the exponentiation is strictly smaller than one except for $r = r'$. Therefore, when the number of communities and their size both tend to infinity, even in the extreme case $P = 1$, the probability that the formula is UNSAT is zero, for $r' < r$, i.e. the phase transition point is the same as for the classical random formulas.

In the case that c is finite, the approximation we have used for the binomial is not correct. When k is constant and $n \to \infty$, we may use

$$\binom{kn}{n} \approx \frac{1}{\sqrt{2\pi n\frac{k-1}{k}}}\left(\frac{k^k}{(k-1)^{k-1}}\right)^n$$

In this case we get

$$P(C_i \text{ UNSAT}) = \mathcal{O}\left(\left(\frac{(\frac{r'}{r}c)^{\frac{r'}{r}c}}{(\frac{r'}{r}c-1)^{\frac{r'}{r}c-1}}\frac{(c-1)^{\frac{r'}{r}c-1}}{c^{\frac{r'}{r}c}}\right)^{rs}\right)$$

Like in the previous case, the base of the exponentiation is one, only when $r' = r$. Therefore, the phase transition point is just the same. ■

6 SAT Solvers Performance

In this section we show that *industrial-specialized* SAT solvers exploit the community structure of the formula, whereas *random-specialized* solvers do not.

In Fig. 3 we show which variable is decided along the execution of MiniSat [Eén and Sörensson, 2003] on one of our random SAT instances. Notice that our generator assigns consecutive numbers to all variables of the same community. We observe that the solver is focused on only one community, only deciding variables of this community, and it changes to another, from time to time.

Finally, in Fig. 4 we compare the performance of the SAT solvers Glucose [Audemard and Simon, 2009] and March [Heule *et al.*, 2004] over some sets of SAT formulas generated with our model, with distinct modularity values.

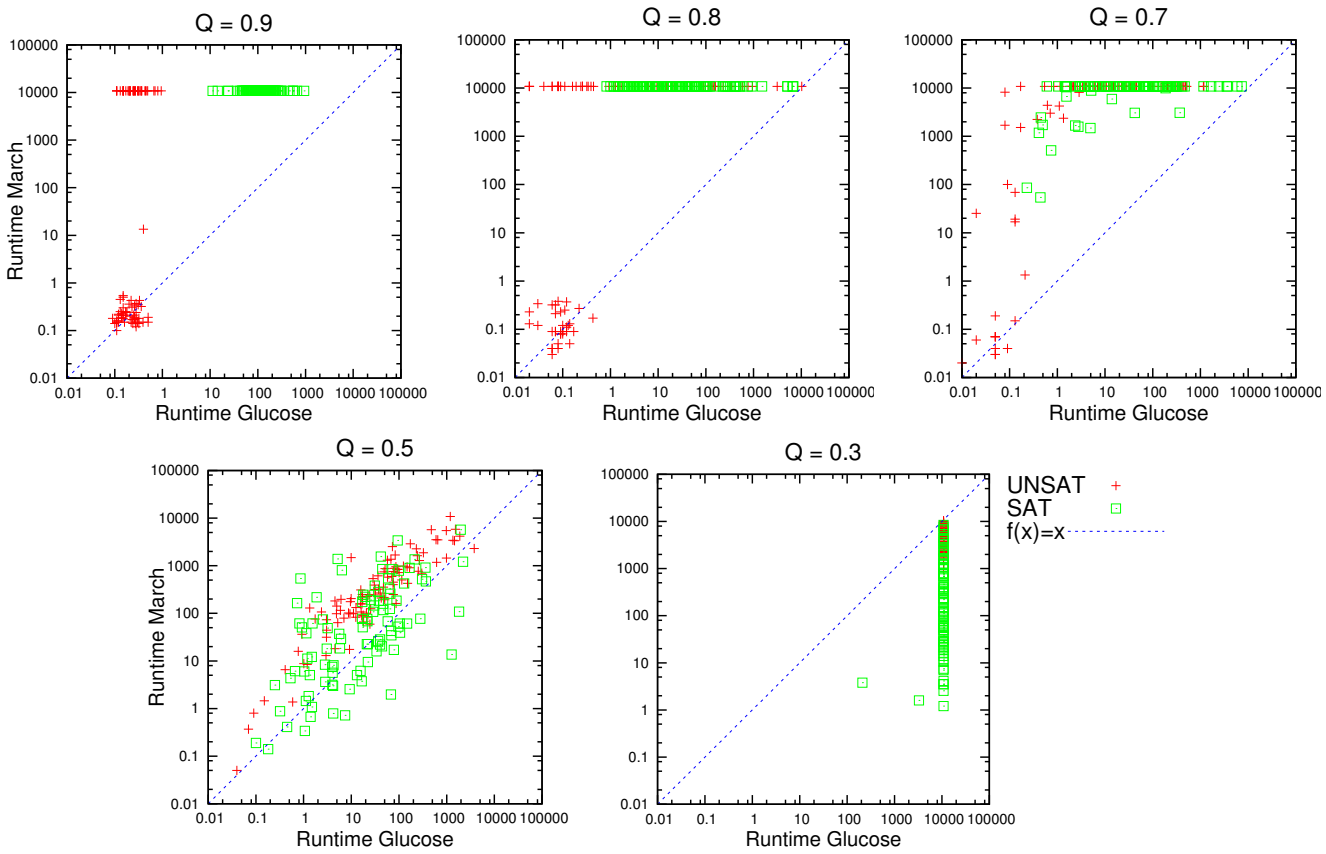

Figure 4: Relation between the runtimes (seconds) of Glucose and March, for some sets of 200 random SAT instances with $Q \in \{0.9, 0.8, 0.7, 0.5, 0.3\}$, $k = 3$ and $c = 40$ at the phase transition point. (i.e., using families of Table 1). The timeout is set to 3 hours.

While Glucose is a CDCL SAT solver which has been shown very good for solving industrial problems, March is a look-ahead SAT solver commonly used to solve random k-CNF instances. For high modularities (see $Q = 0.9$), Glucose solves all the instances, but March is only able to solve few UNSAT instances. More precisely, they are the ones in which there exists a very small unsatisfiability core, composed of variables of one or few communities. Notice that higher the modularity, more likely to find these instances with small refutations. It is also interesting to remark that Glucose also finds UNSAT formulas easier than SAT formulas, for high modularity. As Q decreases, March is able to solve more instances (see $Q = 0.7$), and it starts to be as fast as Glucose, if it is not faster, when the modularity is small enough (see $Q = 0.5$). Finally, when Q is very small (see $Q = 0.3$), March is able to solve all the instances but Glucose only solves few of them. The number of variables used in each set is not the same (see Table 1). We can also conclude that high modularity makes formulas easier to be solved by CDCL SAT solvers. This was also shown in [Newsham *et al.*, 2014].

7 Conclusions

In the SAT community, it is accepted that industrial problems exhibit some kind of *structure* that is exploited by modern solvers. Nowadays, one of the most intriguing problems in this community is to better characterize this structure, with the aim of develop random SAT instances generation models that capture realistically the features of industrial problems, for SAT solving testing purposes.

Recently, the notions of community structure and modularity have been used with success to explain the structure of SAT instances [Ansótegui *et al.*, 2012], and their hardness [Newsham *et al.*, 2014]. Using these concepts, we present a modularity-based generator, which generates random k-CNF SAT instances of any desired modularity Q.

Industrial problems are characterized by a high modularity. Therefore, our model can generate more realistic pseudo-industrial random formulas on demand. We validate the adequacy of this model checking that (i) the community structure of the resulting formulas is the expected, (ii) if there exists a phase transition region SAT-UNSAT characterized by the clause/variable ratio, it is independent on the modularity, and (iii) the SAT solvers performances are consistent to the formulas of our modularity-based generator, i.e. SAT solvers

specialized in industrial (random) problems perform better in *high modular* (*low modular*) instances generated by our model, and vice versa.

References

[Ansótegui *et al.*, 2009a] C. Ansótegui, M. L. Bonet, and J. Levy. On the structure of industrial SAT instances. In *Proc. of CP'09*, pages 127–141, 2009.

[Ansótegui *et al.*, 2009b] C. Ansótegui, M. L. Bonet, and J. Levy. Towards industrial-like random SAT instances. In *Proc. of IJCAI'09*, pages 387–392, 2009.

[Ansótegui *et al.*, 2012] C. Ansótegui, J. Giráldez-Cru, and J. Levy. The community structure of SAT formulas. In *Proc. of SAT'12*, pages 410–423, 2012.

[Ansótegui *et al.*, 2014] C. Ansótegui, M. L. Bonet, J. Giráldez-Cru, and J. Levy. The fractal dimension of SAT formulas. In *Proc. of IJCAR'14*, pages 107–121, 2014.

[Audemard and Simon, 2009] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proc. of IJCAI'09*, pages 399–404, 2009.

[Barabási and Albert, 1999] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[Brandes *et al.*, 2008] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Trans. on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[Burg *et al.*, 2012] S. Burg, M. Kaufmann, and S. Kottler. Creating industrial-like SAT instances by clustering and reconstruction. In *Proc. of SAT'12*, pages 471–472, 2012.

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

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. In *Proc. of SAT'03*, pages 502–518, 2003.

[Heule *et al.*, 2004] M. J. H. Heule, J. E. Zwieten, M. Dufour, and H. Maaren. March_eq: Implementing additional reasoning into an efficient lookahead SAT solver. In *Proc. of SAT'04*, pages 345–359, 2004.

[Järvisalo *et al.*, 2012] M. Järvisalo, P. Kaski, M. Koivisto, and J. H. Korhonen. Finding efficient circuits for ensemble computation. In *Proc. of SAT'12*, pages 369–382, 2012.

[Katsirelos and Simon, 2012] G. Katsirelos and L. Simon. Eigenvector centrality in industrial SAT instances. In *Proc. of CP'12*, pages 348–356, 2012.

[Kautz and Selman, 2003] H. A. Kautz and B. Selman. Ten challenges redux: Recent progress in propositional reasoning and search. In *Proc. of CP'03*, pages 1–18, 2003.

[Martins *et al.*, 2013] R. Martins, V. M. Manquinho, and I. Lynce. Community-based partitioning for maxsat solving. In *Proc. of SAT'13*, pages 182–191, 2013.

[Mitchell *et al.*, 1992] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. of AAAI'92*, pages 459–465, 1992.

[Newman and Girvan, 2004] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69(2):026113, 2004.

[Newsham *et al.*, 2014] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, and L. Simon. Impact of community structure on SAT solver performance. In *Proc. of SAT'14*, pages 252–268, 2014.

[Selman *et al.*, 1997] B. Selman, H. A. Kautz, and D. A. McAllester. Ten challenges in propositional reasoning and search. In *Proc. of IJCAI'97*, pages 50–54, 1997.

[Slater, 2002] A. Slater. Modelling more realistic SAT problems. In *Proc. of AJCAI'02*, pages 591–602, 2002.

[Sonobe *et al.*, 2014] T. Sonobe, S. Kondoh, and M. Inaba. Community branching for parallel portfolio SAT solvers. In *Proc. of SAT'14*, pages 188–196, 2014.

[Walsh, 1999] T. Walsh. Search in a small world. In *Proc. of IJCAI'99*, pages 1172–1177, 1999.