Generating Hard Random Boolean Formulas and Disjunctive Logic Programs

Giovanni Amendola
University of Edinburgh, UK
gamendol@exseed.ed.ac.uk

Francesco Ricca
University of Calabria, Italy
ricca@mat.unical.it

Miroslaw Truszczynski
University of Kentucky, USA
mirek@cs.uky.edu

Abstract

We propose a model of random quantified boolean formulas and their natural random disjunctive logic program counterparts. The model extends the standard models for random SAT and 2QBF. We provide theoretical bounds for the phase transition region in the new model, and show experimentally the presence of the easy-hard-easy pattern. Importantly, we show that the model is well suited for assessing solvers tuned to real-world instances. Moreover, to the best of our knowledge, our model and results on random disjunctive logic programs are the first of their kind.

1 Introduction

Models for generating random instances of search problems have received much attention from the artificial intelligence community in the last twenty years. The results obtained for boolean satisfiability (SAT) [Achlioptas, 2009] and constraint satisfaction (CP) [Mitchell, 2002] have had major impact on the development of fast and robust solvers, significantly expanding their range of effectiveness as general purpose tools for solving hard search and optimization problems arising in AI, and scientific and engineering applications. They also revealed an intriguing phase-transition phenomenon often associated with the inherent hardness of instances, and provided theoretical and experimental basis for a good understanding of the phase-transition phenomenon.

Models of random propositional formulas and QBFs that can reliably generate large numbers of instances of a desired hardness are important [Gent and Walsh, 1999]. Inherently hard instances for SAT and QBF solvers are essential for designing and testing search methods employed by solvers [Achlioptas, 2009], and are used to assess their performance in solver competitions [Järvisalo *et al.* 2012; Narizzano *et al.* 2006; Calimeri *et al.* 2016]. On the flip side, large collections of *easy* instances support the so-called *fuzz* testing, used to reveal problems in solver implementation, as well as defects in solver design [Brummayer *et al.* 2010].

Previous work mainly focused on random CNF formulas and random prenex-form QBFs with the matrix in CNF or DNF (depending on the quantifier sequence). The fixed-length clause model of k-CNF formulas and its 2QBF extension have been especially well studied. Formulas in the fixed-length clause model consist of m clauses over a (fixed) set of n variables, each clause with k non-complementary literals. All formulas are assumed to be equally likely. For that model it is known that there are reals $\rho_l(k)$ and $\rho_u(k)$ such that if $m/n < \rho_l(k)$, a formula from the model is almost surely satisfiable (SAT), and if $m/n > \rho_u(k)$, almost surely unsatisfiable (UNSAT). It is conjectured that $\rho_l(k) = \rho_u(k)$. That conjecture is still open. However, it holds asymptotically, i.e., the two bounds converge to each other with $k \to \infty$ [Achlioptas and Moore, 2002]. For the best studied case of $k = 3$, we have $\rho_l(3) \geq 3.52$ and $\rho_u(3) \leq 4.49$ [Achlioptas, 2009], and experiments show that the phase transition ratio m/n is close to 4.26 [Crawford and Auton, 1996]. Important for the solver design and testing is that instances from the phase transition region are hard and those from regions on both sides of the phase transition are easy, a property called the *easy-hard-easy* pattern [Selman *et al.*, 1996] or, more accurately, the "easy-hard-less hard" pattern [Coarfa *et al.*, 2003]. Empirical studies suggest that SAT solvers devised for solving random formulas are usually not effective with real world instances; *vice versa* solvers for industrial instances are less efficient on random formulas [Järvisalo *et al.*, 2012]. This is often attributed to some form of (hidden) structure present in industrial problems that solvers designed for industrial applications can exploit [Ansótegui *et al.*, 2009]. Finding models to generate random formulas with "structure" that behave similarly to those arising in practice is an important challenge [Kautz and Selman, 2003]. Ansótegui *et al.* [2009] presented the first model that may have this property: despite the "randomness" of its instances, they are better solved by solvers tuned to industrial applications. More recently, Giráldez-Cru and Levy [2016] proposed a model of random SAT based on the notion of *modularity*, and showed that formulas with high modularity behave similarly to industrial ones.

The fixed-length clause model was extended to QBFs by Chen and Interian [2005]. In addition to n and m (understood as above), their model includes parameters controlling the structure of formulas. Once these parameters are fixed, similar properties as in the case of the k-CNF model emerge. There is a phase transition region associated with a specific value of the ratio m/n (that does not depend on n) and the easy-hard-easy pattern can be experimentally verified.

These two models are based on formulas in normal forms.

However, many applications give rise to formulas in non-normal forms motivating studies of solvers of non-normal form formulas and QBFs, and raising the need of models of random non-normal form formulas. The *fixed-shape* model proposed by Navarro and Voronkov [2005], and studied by Creignou *et al.* [2012], is a response to that challenge.

Motivated by the work on random SAT and QBF models, researchers proposed models of random logic programs, and obtained empirical and theoretical results concerning their properties [Zhao and Lin 2003; Namasivayam and Truszczynski 2009; Wang et al. 2015; Wen *et al.* 2016]. Those results are limited to *non-disjunctive* logic programs. No models for *disjunctive* logic programs have been proposed so far. Such results would be of substantial interest to answer set programming (ASP) [Brewka *et al.*, 2011], a popular computational formalism based on disjunctive logic programs.

In this paper, we propose and study models of random *non-normal* form formulas and 2QBFs. Specifically, we consider disjunctions of t k-CNF formulas (in the case of QBFs, using them as matrices). We call models generating such formulas *multi-component*. Instances we obtain in this way are not formulas from the fixed-shape model of Navarro and Voronkov, as their building blocks (k-CNF formulas) do not have a fixed size. Importantly, via the encoding by Eiter and Gottlob [1995], the multi-component models of QBFs give rise to a model of random *disjunctive* logic programs. Our models exhibit the phase transition and, aligned with it, easy-hard-easy pattern. We obtain theoretical bounds on the location of the phase transition, and conduct a comprehensive experimental study. In experiments, we considered several ASP, SAT and QBF solvers to exclude a possibility of a bias being an artifact of committing to a specific solver. For a fixed t, we study hardness as a function of the ratio m/n and show that the regions of hardness for the models lie within their phase transition regions. Importantly, we study hardness as a function of t, the number of components, and show that as t grows generated instances get orders of magnitude harder. As Ansótegui *et al.* [2009], we compare SAT solvers designed for random instances with those designed for real-world ones. We find that for $t \geq 2$ our models generate instances better solved by SAT solvers for real-world instances, and that the difference grows as t grows. We measure the effect of t on processing disjunctive programs, and show that t allows us to control the amount of computation dedicated to stable model checking [Koch *et al.*, 2003].

Our results provide new ways to generate hard and easy instances of propositional formulas, QBFs and disjunctive programs. Our models can generate instances of increasing hardness with properties affecting solver performance in a similar way real-world instances do. The results are particularly important to the development of disjunctive ASP solvers, as no models for generating random disjunctive programs of desired hardness have been known before.

2 Preliminaries

A *clause* is a set of literals that contains no pair of complementary literals. A *CNF formula* is a set of clauses. Disjunctions of CNF formulas are also understood as sets of their constituents. The dual concepts (e.g., DNF formulas) are defined similarly. By $C(k, n, m)$ we denote the set of all k-CNF formulas with m clauses over (some fixed) set of n propositional variables. Similarly, $D(k, n, m)$ is the set of all k-DNF formulas, i.e., disjunctions of m conjunctions of k non-complementary literals over an n-element set of variables.

The fixed-length clause model. The model is given by the set $C(k, n, m)$ of CNF formulas, with all formulas assumed equally likely. Formulas from the model can be generated by selecting m k-literal clauses over a set of n variables uniformly, independently and with replacement. As we noted, the model is well understood. In particular, let us denote by $p(k, n, m)$ the probability that a random formula in $C(k, n, m)$ is SAT. We define $\rho_l(k)$ to be the supremum over all real numbers x such that for every $\rho < x$, $\lim_{n\to\infty} p(k, n, \lfloor \rho n \rfloor) = 1$. Similarly, we define $\rho_u(k)$ to be the infimum over all real numbers x such that for every $\rho > x$, $\lim_{n\to\infty} p(k, n, \lfloor \rho n \rfloor) = 0$. As we mentioned, $\rho_l(k)$ and $\rho_u(k)$ are well defined. Moreover, $\rho_l(k) \leq \rho_u(k)$ and, it is conjectured that $\rho_l(k) = \rho_u(k)$. Experimental results agree with these theoretical predictions.

The Chen-Interian model. The model generates QBFs of the form $\forall X \exists Y F$. Sets X and Y are disjoint and contain all propositional variables that may appear in F. The sizes of X and Y are prescribed to some specific integers A and E, respectively. Moreover, each clause in F contains a literals over X and e literals over Y for some specific values a and e. We denote the set of all such CNF formulas F with m clauses by $C(a, e; A, E; m)$. Clearly, $C(a, e; A, E; m) \subseteq C(a + e, A + E, m)$. We write $Q(a, e; A, E; m)$ for the set of all QBFs $\forall X \exists Y F$, where $F \in C(a, e; A, E; m)$. The Chen-Interian model generates QBFs from $Q(a, e; A, E; m)$, with all formulas equally likely.

Chen and Interian [2005] presented a comprehensive experimental study of the model. Let $q(a, e; A, E; m)$ be the probability that a random QBF from $Q(a, e; A, E; m)$ is true. Let $r > 0$ be fixed real. We set $\nu_l(a, e; r)$ to be the supremum over all real numbers x such that for every $\nu < x$, $\lim_{n\to\infty} q(a, e; A, E; \lfloor \nu n \rfloor) = 1$, where $A = \lfloor rE \rfloor$ and $n = A + E$. Similarly, we set $\nu_u(a, e; r)$ to be the infimum over all real numbers x such that for every $\nu > x$, $\lim_{n\to\infty} q(a, e; A, E; \lfloor \nu n \rfloor) = 0$, again with $A = \lfloor rE \rfloor$ and $n = A + E$. Chen and Interian proved that $\nu_l(a, e; r)$ and $\nu_u(a, e; r)$ are well defined. Clearly, $\nu_l(a, e; r) \leq \nu_u(a, e; r)$. Whether $\nu_l(a, e; r) = \nu_u(a, e; r)$ is an open problem. The quantities $\nu_l(a, e; r)$ and $\nu_u(a, e; r)$ delineate the phase-transition region. For QBFs generated from the model $Q(a, e; \lfloor rE \rfloor, E; \lfloor \nu n \rfloor)$ (with fixed r), Chen and Interian experimentally observed the easy-hard-easy pattern as ν grows, showed that the hard region is aligned with the phase transition, and that the same behavior emerges no matter what concrete r is fixed as the ratio A/E.

3 Multi-component Models

Let $\mathcal{F}$ be a class of propositional formulas (or a model of random formula). By t-$\mathcal{F}$ we denote the class of all disjunctions of t formulas from $\mathcal{F}$ (or a model generating disjunctions of random formulas from $\mathcal{F}$). Similarly, if $\mathcal{Q}$ is a class (model)

of QBFs of the form $\forall X \exists Y F$, where $F \in \mathcal{F}$, we write t-$\mathcal{Q}$ for the class (model) of all QBFs of the form $\forall X \exists Y F$, where $F \in t$-$\mathcal{F}$. We refer to models t-$\mathcal{F}$ and t-$\mathcal{Q}$ as *multi-component*. For QBFs we also consider the dual model to t-$\mathcal{Q}$, based on conjunctions of t DNF formulas. It gives rise to the multi-component model of disjunctive logic programs via the Eiter-Gottlob translation. In all cases, we assume that formulas (QBFs, respectively) are equally likely.

In our work we focus on models based on the model $C(k, n, m)$. First, we note that the corresponding multi-component model t-$C(k, n, m)$ has similar satisfiability properties and that the phase transition regions in the two models are closely related. Let $p_t(k, n, m)$ be the probability that a random formula in t-$C(k, n, m)$ is SAT.

Theorem 1 *Let $t \geq 1$ be a fixed integer. Then, for every $\rho < \rho_l(k)$, $\lim_{n\to\infty} p_t(k, n, \lfloor \rho n \rfloor) = 1$, and for every $\rho > \rho_u(k)$, $\lim_{n\to\infty} p_t(k, n, \lfloor \rho n \rfloor) = 0$.*

The proof follows from the identity $p_t(k, n, m) = 1-(1-p(k, n, m))^t$.Thus, if the phase transition conjecture holds for the single component model $C(k, n, m)$, it also holds for the multi-component model t-$C(k, n, m)$, and the threshold value is the same for every t.

We also considered the multi-component model t-$Q(a, e; A, E; m)$ of QBFs, with the Chen-Interian model as its single-component specialization. Let $q_t(a, e; A, E; m)$ be the probability that a random QBF from t-$Q(a, e; A, E; m)$ is true (in particular, $q_1(a, e; A, E; m) = q(a, e; A, E; m)$). Extending Chen and Interian's work, we can prove that the phase transition for different values of t coincide (and coincide with the phase transition in the Chen-Interian model).

Theorem 2 *For every integer $t \geq 1$ and real $r > 0$, if $\nu < \nu_l(a, e; r)$, then $\lim_{n\to\infty} q_t(a, e; A, E; \lfloor \nu n \rfloor) = 1$, and if $\nu > \nu_u(a, e; r)$, $\lim_{n\to\infty} q_t(a, e; A, E; \lfloor \nu n \rfloor) = 0$ (where $A = \lfloor rE \rfloor$ and $n = A + E$).*

The theorems above describe the situation when t is fixed and n is large. When n is fixed and t grows, the analysis of $p_t(k, n, m)$ and $q_t(a, e; A, E; m)$ shows that the region of the transition from SAT to UNSAT shifts to the right. Of course, once we stop growing t and start increasing n again, the phase transition region will move back to the left.

The experimental results on satisfiability we present later agree with this theoretical analysis; we will also see the easy-hard-easy pattern and a strong dependence of hardness on t.

4 Random Disjunctive Programs

Our results on QBFs imply a model of random disjunctive logic programs. This is important as disjunction increases the expressive power of answer set programming posing, at the same time, a computational challenge [Brewka *et al.*, 2011].

Our model of random disjunctive programs is based on the translation from QBFs to programs due to Eiter and Gottlob [1995]. It works on the model dual to the one we discussed above. The model consists of QBFs $\Phi = \exists X \forall Y F$, where $F \in D(e, a; E, A; m)$, the set of formulas dual to $C(e, a; E, A; m)$. The model clearly has the same properties (modulo a switch between true and false). To describe the translation, let us assume that $X = \{x_1, \ldots, x_E\}$, $Y = \{y_1, \ldots, y_A\}$ and $F = D_1 \vee \ldots \vee D_m$, where $D_i = L_{i,1} \wedge \ldots \wedge L_{i,a+e}$ and $L_{i,j}$ are literals over $X \cup Y$. For every atom $z \in X \cup Y$ we introduce a fresh atom z'. For every $z \in X \cup Y$, we set $\sigma(z) = z$ and $\sigma(\neg z) = z'$. Finally, we introduce one more fresh atom, say w, and define a disjunctive logic program P_Φ to consist of the following rules:

$z \vee z'$	for each $z \in X \cup Y$
$y \leftarrow w$ and $y' \leftarrow w$	for each $y \in Y$
$w \leftarrow \sigma(L_{i,1}), \ldots, \sigma(L_{i,a+e})$	for each D_i, $i = 1, \ldots, m$
$w \leftarrow not\ w$	

Eiter and Gottlob proved that Φ is true if and only if P_Φ has an answer set. Thanks to this correspondence, the model $D(a, e; A, E; m)$ gives rise to a model of disjunctive logic programs. We denote it (and, with some abuse of notation, the set of programs it consists of) by $D_{dlp}(e, a; E, A; m)$.

The set of atoms of programs in $D_{dlp}(e, a; E, A; m)$ is $\{w\} \cup Z \cup Z'$, where $Z = X \cup Y$ and $Z' = \{z' : z \in Z\}$. The structure of these programs is quite simple. Each program has a fixed part (the same in all programs)

$z \vee z'$	for each $z \in Z$
$y \leftarrow w$ and $y' \leftarrow w$	for each $y \in Y$
$w \leftarrow not\ w$	

and its unique *core* consisting of m Horn rules

$$w \leftarrow z_1, \ldots, z_\ell$$

where $\ell = e + a$ and the body of each rule has e atoms z and z' with $z \in X$ (and so, also a atoms z and z' with $z \in Y$).

This structure can be generalized. Assuming an additional set of t fresh atoms $w_1, \ldots, w_t$, we write t-$D_{dlp}(e, a; E, A; m)$ for the set of programs consisting of:

$z \vee z'$	for each $z \in Z$
$y \leftarrow w$ and $y' \leftarrow w$	for each $y \in Y$
$w \leftarrow w_1, \ldots, w_t$ and $w \leftarrow not\ w$	

as the fixed part, and the core of mt Horn rules of the form

$$w_h \leftarrow z_1, \ldots, z_\ell$$

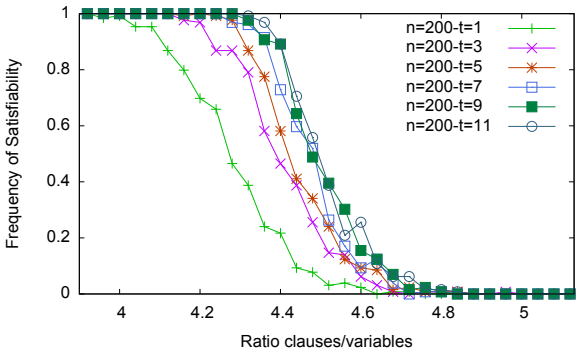
where $h = 1, \ldots, t$, each w_h is the head of exactly m rules, $\ell = e + a$, and the body of each rule has e atoms z and z' with $z \in X$ (and so, also a atoms of the form z and z' with $z \in Y$). Note that programs in 1-$D_{dlp}(e, a; E, A; m)$ coincide with those in $D_{dlp}(e, a; E, A; m)$ modulo a rewriting, where the rule $w \leftarrow w_1$ is removed and w_1 is replaced by w.

Programs in t-$D_{dlp}(e, a; E, A; m)$ correspond to $\exists\forall$ QBFs whose matrix belongs to t-$D(e, a; E, A; m)$. They can be seen as the results of translating QBFs Φ into logic programs Φ_{dlp}, where we use variables w_i to represent component DNF formulas in the matrix of Φ. The correspondence $\Phi \mapsto \Phi_{dlp}$ preserves the semantics in the following sense.

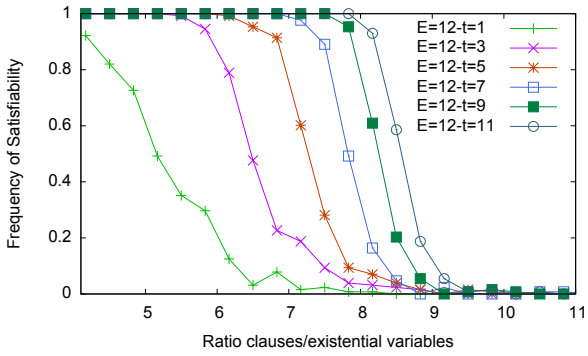
Theorem 3 *Let $\Phi = \exists X \forall Y F$, for $F \in t$-$D(e, a; E, A; , m)$. Then Φ is true if and only if Φ_{dlp} has an answer set, where Φ_{dlp} is the disjunctive logic program in t-$D_{dlp}(e, a; E, A; m)$ corresponding to Φ.*

5 Empirical Analysis

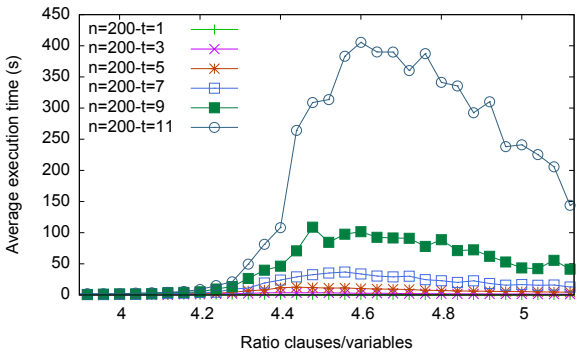
We now present an experimental analysis of the behavior of our models and discuss their properties.



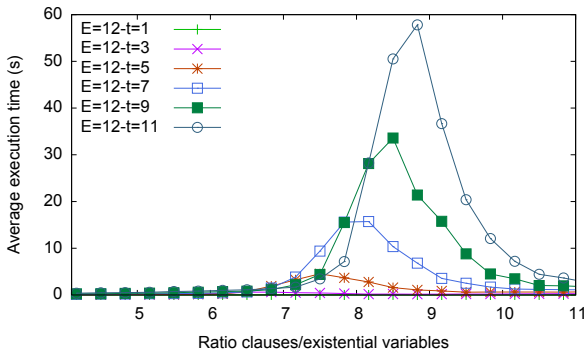
(a) Phase transition shift (SAT)



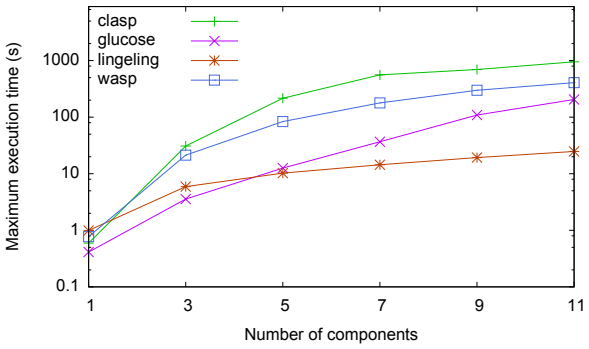
(b) Phase transition shift (QBF)



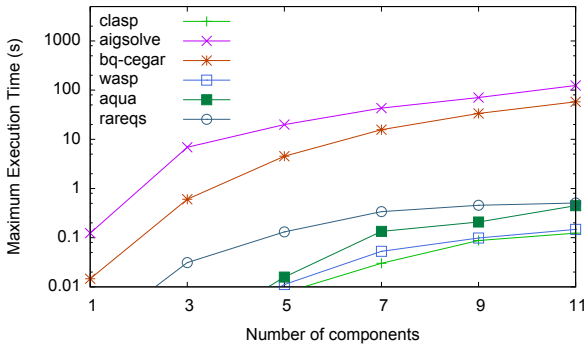
(c) Easy-hard-easy pattern (SAT)



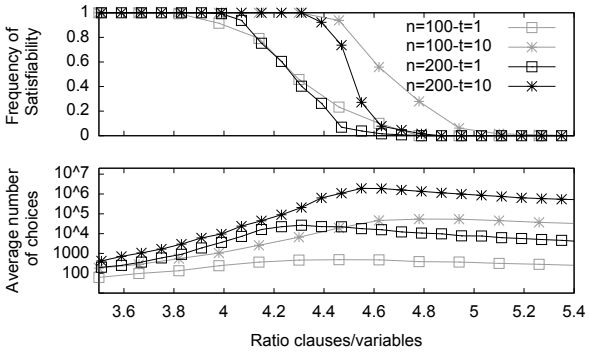
(d) Easy-hard-easy pattern (QBF)



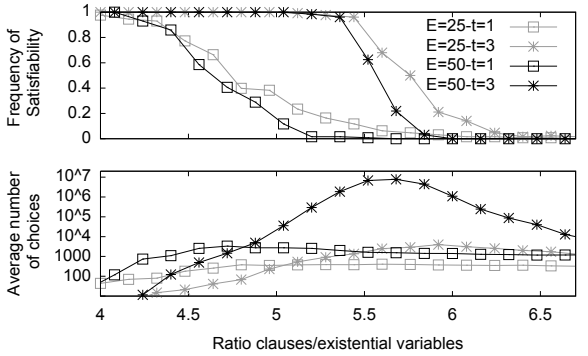
(e) Hardness of multi-component model (SAT)



(f) Hardness of multi-component model (QBF)



(g) Combined effect of variables and components (SAT)



(h) Combined effect of variables and components (QBF)

Figure 1: Behavior of multi-component model: phase transition and hardness.

Experiment Setup. To claim that properties and patterns are inherent to a model and not an artifact of a solver used, we performed our experiments with several well-known SAT, QBF and ASP solvers: the SAT solvers GLUCOSE 4.0 [Audemard *et al.*, 2013] LINGELING, version of 2015 [Biere, 2014], and KCNFS, version of SAT'07 competition [Dequen and Dubois, 2006]; the QBF solvers BQ-CEGAR[1], AIGSOLVE [Pigorsch and Scholl, 2010]; RAREQS [Janota *et al.*, 2016], and AQUA-S2V[2] and the ASP solvers CLASP 3.1.3 [Gebser *et al.*, 2007] and WASP 2.1 [Alviano *et al.*, 2015], both paired with *gringo* 4.5.3.All solvers were run in their default configurations. We stress that we did not aim at comparing solver performance, instead *our goal was to identify solver-independent properties inherent to a model*.

Formulas generated according to the multi-component model with $t > 1$ are non-clausal, whereas the (Q)DIMACS format requires the formulas to be in CNF. Therefore, the generator transforms non-clausal formulas to CNF using the standard Tseitin transformation. Once a formula Φ is generated, it is stored in two files: one with an encoding of Φ in the (Q)DIMACS numeric format of (Q)SAT solvers [Järvisalo *et al.*, 2012; Narizzano *et al.*, 2006], and the other one with the disjunctive logic program corresponding to Φ in the ASP-Core 2.0 syntax [Calimeri *et al.*, 2016]. Since the formulas we generate are of the form $\forall X \exists Y F$, the programs are obtained from their negations $\exists X \forall Y \neg F$. They have answer sets if and only if the original formulas are false. Thus, when we analyze satisfiability we plot only the curves obtained by evaluating either the formulas or the corresponding logic programs (plots are symmetric). Experiments were run on a Debian Linux with 2.30GHz Intel Xeon E5-4610 v2 CPUs and 128GB of RAM. Each execution was constrained to one single core by using the *taskset* command. Time measurements were performed by using the *runlim* tool. Formulas were generated with a tool we developed in Java.The results are averaged over 128 samples of the same size.

Behavior of Multi-component Model. To study the satisfiability of multi-component model instances (the location of the phase transition), we considered the setting with the number of variables (propositional atoms) fixed. Figure 1(a) shows the results for the t component model t-$C(3, 200, m)$, with $t \in \{1, 3, 5, 7, 9, 11\}$. The x-axis gives the ratio of the numbers of clauses and variables ($m/200$), the y-axis shows the frequency of SAT. Consistently with our theoretical results, the phase transition shifts from left to right, and it sharpens for growing values of t. The same can be observed in Figure 1(b), showing the frequency of QBFs from t-$Q(1, 3, 24, 12, m)$ that are true, for $t \in \{1, 3, 5, 7, 9, 11\}$.

To study the hardness of the multi-component model we computed the average solver running times. The results (on the same instances as before) for the GLUCOSE SAT solver and the BQ-CEGAR QBF solver are in Figures 1(c) and 1(d). The plots show a strong dependency of the hardness on the number of components: the peak of hardness moves right and grows visibly with t. In more detail, the CNF formulas (one component) are solved by GLUCOSE in less than 0.42s, whereas instances with 11 components require about 7 minutes, i.e, they are more than 3 orders of magnitude harder. Analogous behavior is observed when running BQ-CEGAR on QBF formulas. Those from the one-component model are solved instantaneously (average time ≤ 0.01s), those from the 11-component model requires about one minute. The experiments with other solvers gave similar results.

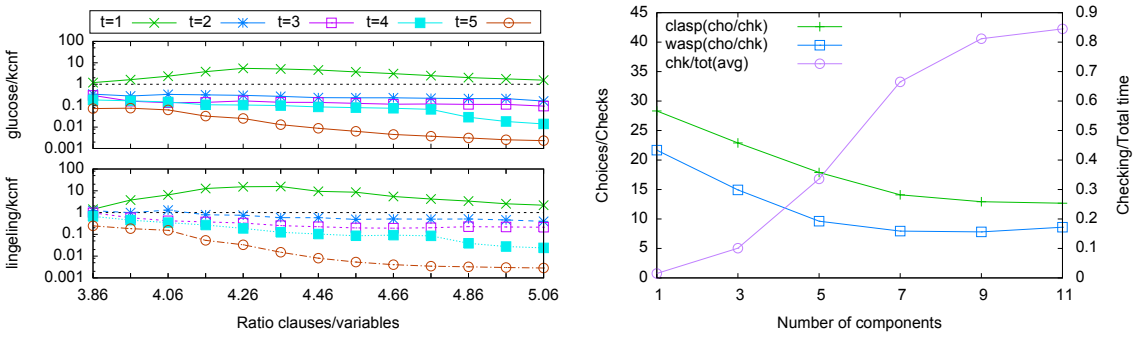
To underline the dependency of the hardness on the number of components, for each solver we compute the average time over samples of the same size and plot its maximum (for simplicity *maximum execution time*) for several values of t in Figures 1(e) (SAT) and 1(f) (QBF, programs). In particular, Figure 1(e) reports the results obtained by running GLUCOSE and LINGELING, and Figure 1(f) — the results obtained by running BQ-CEGAR, AIGSOLVE, AQUA-S2V, RAREQS and the results obtained by running CLASP and WASP on the corresponding programs. The picture shows that the peak of difficulty grows with the number of components no matter the implementation or the representation roughly, at a rate that is more than quadratic with t (y-axis in logarithmic scale).

Next, we discuss the behavior of formulas when both the number of variables and the number of components grow. Figure 1(g) reports on the behavior of CNF formulas with $n \in \{100, 200\}$ and $t \in \{1, 10\}$. Formulas with 100 variables are plotted in grey, and those with 200 variables in black. We use squares to identify graphs for formulas with one component and stars for graphs concerning formulas with ten components. Figure 1(g) shows that when the number of variables grows the phase transition moves to the left, and the transition becomes sharper. By Theorem 1, we expect that the bounds on (un)satisfiability do not depend on t, indeed when the number of variables grows the right shift due to an increase in the number of components is compensated, and becomes negligible. Our experiments also confirm that hardness grows with both the number of components and the number of variables. This is seen in Figure 1(g) in the bottom, which plots the average number of choices taken by CLASP (we consider choices since execution times are negligible). Note that CNF formulas with 100 variables and 10 components are already harder than formulas with 200 variables and one component. Figure 1(h) shows the same picture for QBFs in t-$Q(1, 3, 50, 25, m)$ (plotted in grey) and t-$Q(1, 3, 100, 50, m)$ (plotted in black), with $t \in \{1, 3\}$. These results were obtained by running CLASP on the corresponding programs.

Impact on SAT Solving. A desirable property of a random model is to generate instances that behave similarly to real-world ones [Kautz and Selman, 2003; Ansótegui *et al.*, 2009]. This similarity has been measured empirically by comparing the performance of solvers for random and industrial instances. Following [Ansótegui *et al.*, 2009], we measure the ratio of the execution times of solvers. We compared KCNFS (a well-known SAT solver specialized in random instances) with GLUCOSE and LINGELING (both specialized in real-world instances) to assess whether our model allows to generate instances that are better solved by solvers for real-world instances. In particular, Figure 2(a) shows the results for the model t-$C(3, 100, m)$, while varying the number of components $t \in \{1, 2, 3, 4, 5\}$. The x-axis gives the ratio

[1] A combination of *bloqqer* preprocessor [Heule *et al.*, 2015] and *ghostq* [Klieber *et al.*, 2013] solver from QBF gallery 2014.

[2] www.qbflib.org/DESCRIPTIONS/aqua16.pdf



(a) CPU time ratio glucose/kcnfs and lingeling/kcnfs

(b) Impact on answer set computation

Figure 2: Impact on solving technology.

of the numbers of clauses and variables ($m/100$), the y-axis shows GLUCOSE versus KCNFS (top) and LINGELING versus KCNFS (bottom). We observe that, KCNFS is faster (ratios > 1) than both GLUCOSE and LINGELING when $t = 1$, i.e., when our model coincides with the classical one for random formulas. Once we increase the number of components the result is reverted, GLUCOSE and LINGELING are faster than KCNFS (ratios < 1), and the difference grows significantly with t. This is independent of the clauses/variables ratio.

The difference between random and real-world instances is often attributed to the presence of some *hidden structure* in the latter [Ansótegui *et al.*, 2009]. We observed that multi-component models yield instances that are solved faster by solvers designed for real-world instances. We conjecture this is due to the component structure introduced by the model. This structure can be controlled by varying the number of components, yielding instances of varying hardness.

Impact on QBF and ASP Solving. An analysis distinguishing the behavior of random and industrial instances is not possible for ASP and QBF solvers. Indeed, no QBF/ASP solvers have ever been designated as (or known to be) specialized to random instances in ASP and QBF Evaluations so far (cf. [Calimeri *et al.*, 2016; Narizzano *et al.*, 2006] and `http://www.qbflib.org`). Nonetheless our model has other interesting implications for QBF and ASP solvers.

To assess the validity of our model for QBF, we submitted several instances to the QBF Evaluation 2016. All our instances (with $n = 100$ only, and $t \leq 6$) were classified as *hard* by the organizers, and helped identify a bug in one of the participating solvers, demonstrating the efficacy of our model in performance analysis and in correctness testing.

For ASP solvers, Figure 2(b) outlines the impact of our model on answer set search for programs corresponding to QBF formulas t-$Q(1, 3, 24, 12, m)$ with $t \in \{1, 3, 5, 7, 9, 11\}$. ASP solvers evaluate disjunctive programs by first computing a candidate model, and then checking its stability (the latter task is co-NP complete). Thus, we plot (i) the ratio between the number of choices made during the search phase and the number of stable model checks performed by WASP and CLASP, and (ii) the ratio between the time spent in stable model checking and the total execution time for the solver WASP (results for CLASP are analogous) both for growing t. The ratio between the numbers of choices and model checks decreases when the number of components grows, following a similar behavior for both solvers. This is a machine-independent measure of the impact of the two activities, and we observe that the role of the model checker grows with t. Specifically, the impact of the model checking on the total solving time grows from about about 3% ($t = 1$) to 88% ($t = 11$). It is known that, on available benchmarks, ASP solvers spend more time in the model search phase than in the model checking phase (this also happens when $t = 1$). However, our model allows to generate in a controllable way instances that put emphasis also the model checking phase.

6 Conclusions

In this paper we proposed the multi-component models for random propositional formulas, and disjunctive logic programs. Despite their simple structure the models have theoretical and empirical properties that make them important for further advancement of the SAT, QBF and ASP solvers. First, the hardness of formulas/programs can be controlled and, unlike in the earlier models, not only in terms of the ratio of clauses to variables. Our experiments showed that the hardness *strongly* depends on the number of components, and even a small number of components can lead to very hard instances. Further, the multi-component model generates formulas that in at least one aspect are similar to instances arising in practice: they are solved better by solvers specialized in industrial benchmarks than by solvers specialized in random ones. This makes them useful for development and testing of solvers intended for practical applications. Finally, our model of random disjunctive programs is the first model for that class of logic objects. This and the fact that it allows us to control the role of the stable model checking phase point to its potential for the development of ASP solvers.

Acknowledgments

This work was partially supported by the Italian Ministry MISE under projects "PIUCultura" (n. F/020016/01-02/X27), and "Smarter Solutions in the Big Data World" (PON 2014-2020), and by EPSRC grant N023056.

References

[Achlioptas and Moore, 2002] D. Achlioptas and C. Moore. The asymptotic order of the random k-sat threshold. In *FOCS 2002*, pages 779–788, 2002.

[Achlioptas, 2009] D. Achlioptas. Random satisfiability. In *Handbook of Satisfiability*, pages 245–270. 2009.

[Alviano *et al.*, 2015] M. Alviano, C. Dodaro, N. Leone, and F. Ricca. Advances in WASP. In *LPNMR 2015*, pages 40–54, 2015.

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

[Audemard *et al.*, 2013] G. Audemard, J-M. Lagniez, and L. Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *SAT 2013*, volume 7962 of *LNCS*, pages 309–317, 2013.

[Biere, 2014] A. Biere. Lingeling essentials, a tutorial on design and implementation aspects of the the SAT solver lingeling. In *POS-14.*, volume 27 of *EPiC*, page 88, 2014.

[Brewka *et al.*, 2011] G. Brewka, T. Eiter, and M. Truszczynski. Answer set programming at a glance. *Commun. ACM*, 54(12):92–103, 2011.

[Brummayer *et al.*, 2010] R. Brummayer, F. Lonsing, and A. Biere. Automated testing and debugging of SAT and QBF solvers. In *SAT 2010*, volume 6175 of *LNCS*, pages 44–57, 2010.

[Calimeri *et al.*, 2016] F. Calimeri, M. Gebser, M. Maratea, and F. Ricca. Design and results of the fifth answer set programming competition. *Artif. Intell.*, 231:151–181, 2016.

[Chen and Interian, 2005] H. Chen and Y. Interian. A model for generating random quantified boolean formulas. In *IJCAI 2005*, pages 66–71, 2005.

[Coarfa *et al.*, 2003] C. Coarfa, D.D. Demopoulos, A. San Miguel Aguirre, D. Subramanian, and M.Y. Vardi. Random 3-sat: The plot thickens. *Constraints*, 8(3):243–261, 2003.

[Crawford and Auton, 1996] J.M. Crawford and L.D. Auton. Experimental results on the crossover point in random 3-sat. *Artif. Intell.*, 81(1-2):31–57, 1996.

[Creignou *et al.*, 2012] N. Creignou, U. Egly, and M. Seidl. A framework for the specification of random SAT and QSAT formulas. In *TAP 2012*, volume 7305 of *LNCS*, pages 163–168, 2012.

[Dequen and Dubois, 2006] G. Dequen and O. Dubois. An efficient approach to solving random *k*-sat problems. *J. Autom. Reasoning*, 37(4):261–276, 2006.

[Eiter and Gottlob, 1995] T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *AMAI*, 15(3-4):289–323, 1995.

[Gebser *et al.*, 2007] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. *clasp* : A conflict-driven answer set solver. In *LPNMR 2007*, volume 4483 of *LNCS*, pages 260–265, 2007.

[Gent and Walsh, 1999] I. P. Gent and T. Walsh. Beyond NP: the QSAT phase transition. In *AAAI*, pages 648–653, 1999.

[Giráldez-Cru and Levy, 2016] J. Giráldez-Cru and J. Levy. Generating SAT instances with community structure. *Artif. Intell.*, 238:119–134, 2016.

[Heule *et al.*, 2015] M. Heule, M. Järvisalo, F. Lonsing, M. Seidl, and A. Biere. Clause elimination for SAT and QSAT. *J. Artif. Intell. Res. (JAIR)*, 53:127–168, 2015.

[Janota *et al.*, 2016] M. Janota, W. Klieber, J. Marques-Silva, and E.M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

[Järvisalo *et al.*, 2012] M. Järvisalo, D. Le Berre, O. Roussel, and L. Simon. The international SAT solver competitions. *AI Magazine*, 33(1), 2012.

[Kautz and Selman, 2003] H. A. Kautz and B. Selman. Ten challenges redux: Recent progress in propositional reasoning and search. In *CP 2003*, volume 2833 of *LNCS*, pages 1–18, 2003.

[Klieber *et al.*, 2013] W. Klieber, M. Janota, J. Marques-Silva, and E. M. Clarke. Solving QBF with free variables. In *CP 2013*, volume 8124 of *LNCS*, pages 415–431, 2013.

[Koch *et al.*, 2003] C. Koch, N. Leone, and G. Pfeifer. Enhancing disjunctive logic programming systems by SAT checkers. *Artif. Intell.*, 151(1-2):177–212, 2003.

[Mitchell, 2002] D. G. Mitchell. Resolution complexity of random constraints. In *CP 2002*, volume 2470 of *LNCS*, pages 295–309, 2002.

[Namasivayam and Truszczynski, 2009] G. Namasivayam and M. Truszczynski. Simple random logic programs. In *LPNMR 2009*, volume 5753 of *LNCS*, pages 223–235, 2009.

[Narizzano *et al.*, 2006] M. Narizzano, L. Pulina, and A. Tacchella. Report of the third QBF solvers evaluation. *JSAT*, 2(1-4):145–164, 2006.

[Navarro and Voronkov, 2005] J. Navarro and A. Voronkov. Generation of hard non-clausal random satisfiability problems. In *AAAI 2005*, pages 436–442, 2005.

[Pigorsch and Scholl, 2010] F. Pigorsch and C. Scholl. An AIG-based QBF-solver using SAT for preprocessing. In *DAC 2010*, pages 170–175, 2010.

[Selman *et al.*, 1996] B. Selman, D. G. Mitchell, and H. J. Levesque. Generating hard satisfiability problems. *Artif. Intell.*, 81(1-2):17–29, 1996.

[Wang *et al.*, 2015] K. Wang, L. Wen, and K. Mu. Random logic programs: linear model. *TPLP*, 15(6):818–853, 2015.

[Wen *et al.*, 2016] L. Wen, K. Wang, Y. Shen, and F. Lin. A model for phase transition of random answer-set programs. *ACM Trans. Comput. Log.*, 17(3):22, 2016.

[Zhao and Lin, 2003] Y. Zhao and F. Lin. Answer set programming phase transition: A study on randomly generated programs. In *ICLP 2003*, volume 2916 of *LNCS*, pages 239–253, 2003.