New Models for Generating Hard Random Boolean Formulas and Disjunctive Logic Programs☆

Giovanni Amendola[a], Francesco Ricca[a,*], Miroslaw Truszczynski[b]

[a]*Department of Mathematics and Computer Science, University of Calabria, Italy*
[b]*Department of Computer Science, University of Kentucky, USA*

Abstract

We propose two models of random quantified boolean formulas and their natural random disjunctive logic program counterparts. The models extend the standard models of random *k*-CNF formulas and the Chen-Interian model of random 2QBFs. The first model controls the generation of programs and QSAT formulas by imposing a specific structure on rules and clauses, respectively. The second model is based on a family of QSAT formulas in a non-clausal form. We provide theoretical bounds for the phase transition region in our models, and show experimentally the presence of the easy-hard-easy pattern and its alignment with the location of the phase transition. We show that boolean formulas and logic programs from our models are significantly harder than those obtained from the standard *k*-CNF and Chen-Interian models, and that their combination yields formulas and programs that are "super-hard" to evaluate. We also provide evidence suggesting that formulas from one of our models are well suited for assessing solvers tuned to real-world instances. Finally, it is noteworthy that, to the best of our knowledge, our models and results on random disjunctive logic programs are the first of their kind.

Keywords: Answer Set Programming, Random Boolean Formulas, Phase Transition, Random Logic Programs

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) [2, 42] and constraint satisfaction (CP) [35] have had a 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 "region" where the phase-transition occurs.

Models of random propositional formulas and QBFs that can reliably generate large numbers of instances of a *desired* hardness are important [26]. Inherently hard instances for SAT and QBF solvers are essential for designing and testing search methods employed by solvers [2], and are used to assess their performance in solver competitions [30, 38, 12]. 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 [11].

Previous work on models of random formulas 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*

☆Some of the results were presented in preliminary form at IJCAI 2017 [5].
*Corresponding author
URL: amendola@mat.unical.it (Giovanni Amendola), ricca@mat.unical.it (Francesco Ricca), mirek@cs.uky.edu (Miroslaw Truszczynski)

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).[1] 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 $n \to \infty$ [3], and also it is known that $\rho_l(k) = \rho_u(k)$ for sufficiently large k [19]. For the best studied case of $k = 3$, we have $\rho_l(3) \geq 3.52$ [31] and $\rho_u(3) \leq 4.49$ [18], and experiments show that the phase transition ratio m/n is close to 4.26 [15]. 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 [36] or, more accurately, the "easy-hard-less hard" pattern [14]. 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 [30]. This is often attributed to some form of (hidden) structure present in industrial problems that solvers designed for industrial applications can exploit [7]. Finding models to generate random formulas with "structure" that behave similarly to those arising in practice is an important challenge [32]. Ansotegui et al. [6] 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 [27] 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 [13]. 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 [39], and studied by Creignou et al. [16], is a response to that challenge. The model is similar to that of the k-CNF one (or its extensions to QBFs), but *fixed shape* (and size) non-normal form formulas are used in place of k-clauses as the key building blocks. Experimental studies again show the phase-transition and the easy-hard-easy pattern.

One of the most extensively studied and most successful computational knowledge representation formalisms since late 1990s has been answer set programming or ASP, for short [10]. This formalism is based on the language of disjunctive logic programming with the semantics of *answer sets* (for some fragments of the language also called *stable models*) [25]. The formalism has now a well-understood theory, effective implementations and exciting applications [10, 1]. The key computational task behind it is to find answer sets of propositional programs. This task requires search and shares strong commonalities with the SAT and QBF solving.

To advance the development and testing of ASP solvers, and 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 [47, 46, 37, 44, 45]. Zhao and Lin [47] proposed fixed rule-length and mixed rule-length models. For the former, they observed the existence of the phase transition and showed that programs within the phase transition were hard for available solvers. Namasivayam and Truszczynski [37] studied programs consisting of two-literal rules. In that case, rules can be classified into five types, and several interesting classes of programs can be defined by restricting the types of two-literal rules they contain. Namasivayam and Truszczynski studied changes in the hardness of such programs as numbers of rules of a particular type varied, and found several specific combinations of parameters that ensured hardness. Wang et al. [44] proposed a *linear* model of random logic programs and studied the average number and the distribution of sizes of answer sets in programs from that model. Wen et al., [45] proposed the *quadratic* model of logic programs. They showed the model to have the phase transition property and an associated easy-hard-easy computational pattern.

So far, those results have been limited to *non-disjunctive* logic programs. No models for *disjunctive*

[1] We give a precise statement of these properties in Section 2.1.

logic programs have been proposed so far. In this paper we propose two models of random QBF formulas and the corresponding models of disjunctive logic programs. First, we propose a *controlled* version of the Chen-Interian model in which CNF formulas that are used as matrices are subject to additional conditions restricting their structure. Second, we propose *multi-component* versions of the earlier models.[2] In the multi-component models, propositional formulas and matrices of QBFs are disjunctions of t k-CNF formulas (either standard or "controlled"), for some integer t. They are not formulas from the fixed-shape model of Navarro and Voronkov, as their building blocks (CNF or DNF formulas) do not have a fixed size. In each case, the standard translation from QBFs to disjunctive programs suggests random models for the latter.

For the new models, we present theoretical bounds on the region where the phase transition is located, and study experimentally their behavior. In our experiments, we consider several ASP, SAT and QBF solvers to exclude any possible bias that could be an artifact of a particular solver. We study the regions of hardness for the models and show empirically that they lie within their phase transition regions. We compare the hardness of the controlled model with the corresponding Chen-Interian model and find that the former can generate formulas that are significantly harder. For the multi-component versions of the standard random CNF and the Chen-Interian models we study hardness as a function of the ratio m/n and the number t of components. The results show that the multi-component model allows us to control hardness of formulas and programs by changing t. Even when the number of variables is fixed, raising t may result in exponential growth in solving time. The results also show that the combination of controlled and multi-component models allows us to generate instances that are "super-hard" to evaluate.

As Ansótegui et al. [6], we compare SAT/QBF 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 solvers for real-world instances, and that the difference becomes more pronounced as t grows. For disjunctive logic programs, we measure the effect of t on processing them and show that t allows us to control the amount of computation dedicated to stable model checking [34].

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. By a *CNF formula* we mean an (ordered) tuple of clauses with repetitions of clauses allowed. Disjunctions of CNF formulas are also assumed to be (ordered) tuples and they also allow repetitions. The dual concepts (such as DNF formulas) are defined similarly. In other models, CNF formulas are viewed as *sets* of clauses, and disjunctions of CNF formulas are viewed as sets of CNF formulas. However, assuming some reasonable limit on the number of clauses in a formula, and assuming in each case the uniform distribution, the two probabilistic models are asymptotically equivalent for properties that do not depend on the order (such as satisfiability). Specifically, as the number of atoms tends to infinity, the probability that such a property holds in one model and the corresponding probability for the other model converge to each other. (We offer a technical justification for this claim in Appendix B.) Thus, there is no essential difference between the two models and we use them interchangeably.

By $C(k,n,m)$ we denote the set of all k-CNF formulas consisting of m clauses over (some fixed) set of n propositional variables. Similarly, $D(k,n,m)$ stands for the set of all k-DNF formulas of m *products* (conjunctions of non-complementary literals) over an n-element set of atoms.

2.1. The fixed-length clause model

The fixed-length clause 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

[2]When we use the term "multi-component model," we understand that the notion is parameterized by the underlying "standard," or single-component, model.

n variables uniformly, independently and with repetition. 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 ρ such that $\lim_{n\to\infty} p(k,n,\lfloor \rho n \rfloor) = 1$. Similarly, we define $\rho_u(k)$ to be the infimum over all real numbers ρ such that $\lim_{n\to\infty} p(k,n,\lfloor \rho n \rfloor) = 0$. As we mentioned, $\rho_l(k)$ and $\rho_u(k)$ are well defined. Moreover, $\rho_l(k) \le \rho_u(k)$ and it is conjectured that $\rho_l(k) = \rho_u(k)$. Experimental results agree with these theoretical predictions [36, 15, 14, 31, 18].

2.2. The Chen-Interian model

The Chen-Interian 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 [13] presented a comprehensive experimental study of the model. Let us denote by $q(a,e;A,E;m)$ the probability that a random QBF from $Q(a,e;A,E;m)$ is true. Given a real number $r > 0$, we set $\nu_l(a,e;r)$ to be the supremum over all real numbers ν such that $\lim_{n\to\infty} q(a,e;A,E;\lfloor \nu n \rfloor) = 1$, 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 ν such that $\lim_{n\to\infty} q(a,e;A,E;\lfloor \nu n \rfloor) = 0$, again with $A = \lfloor rE \rfloor$ and $n = A+E$. Chen and Interian [13] proved the following result.

Theorem 1. *For every positive integers a and e, and for every real number $r > 0$, the values $\nu_l(a,e;r)$ and $\nu_u(a,e;r)$ are well defined (that is, the sets of values ν in the definitions of those quantities are non-empty and bounded from above, in the first case, and from below, in the second one).*

Clearly, $\nu_l(a,e;r) \le \nu_u(a,e;r)$. 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 n and r), Chen and Interian experimentally observed the easy-hard-easy pattern as ν grows. They 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. New models of random formulas and QBFs

We propose several variations of the models described above. They are based on two ideas. First, we impose an additional structure on clauses in CNF formulas that serve as matrices of QBFs. Second, we consider disjunctions of CNF formulas both in the SAT and QBF setting.

3.1. The controlled model

To describe the model, we define first a version of a model of a random CNF formula. In this model, clauses are built of variables in a set $X \cup Y$, where $X \cap Y = \emptyset$; we set $|X| = A$ and $|Y| = E$. A formula in the model consists of $2A$ k-literal clauses. Each clause consists of a single literal over X and $k-1$ literals over Y, and for each literal over X there is a single clause in the formula that contains it. A formula in this model is generated taking $2A$ $(k-1)$-literal clauses over Y and extending each of them by a literal over X (following some fixed one-to-one mapping between the clauses and the literals over X). We denote this model (and the corresponding set of formulas) by $C^{ctd}(k,A,E)$. We write $Q^{ctd}(k,A,E)$ for the model (and the set) of QBFs whose matrix is a formula from $C^{ctd}(k,A,E)$. We refer to both models as *controlled*. In our work we are primarily interested in the controlled model for QBFs.

Clearly, $Q^{ctd}(k,A,E) \subseteq Q(1,k-1;A,E;2A)$. Thus, the controlled model is related to the Chen-Interian model. The main difference is that the clauses, while random with respect to existential variables are not random with respect to universal variables. For each $x \in X$ there is exactly one clause involving x and exactly one clause involving $\neg x$. Consequently, the number of clauses is $2A$ and, moreover, for every truth assignment to X, once we simplify the matrix accordingly, we are left with *exactly* (hence, the term "controlled") A $(k-1)$-literal clauses over E variables. In contrast, in the case of the Chen-Interian model

$Q(1,k-1;A,E;2A)$, similar simplifications leave us with $(k-1)$-CNF formulas with *varying* number of clauses, with the *average* number being A.

Let $q^{ctd}(k,A,E)$ denote the probability that a random formula in $Q^{ctd}(k,A,E)$ is true. As before, we define $\mu_l^{ctd}(k)$ to be the supremum over all positive real numbers ρ such that $\lim_{E\to\infty} q^{ctd}(k,\lfloor\rho E\rfloor,E)=1$, and $\mu_u^{ctd}(k)$ to be the infimum over all positive real numbers ρ such that $\lim_{E\to\infty} q^{ctd}(k,\lfloor\rho E\rfloor,E)=0$.

We will now derive bounds on $\mu_l^{ctd}(k)$ and $\mu_u^{ctd}(k)$ by exploiting results on random $(k-1)$-CNF formulas.

Theorem 2. *For every $k\geq 2$, $\mu_l^{ctd}(k)\geq \frac{\rho_l(k-1)}{2}$ and $\mu_u^{ctd}(k)\leq \rho_u(k-1)$.*

Proof. Let $\Phi\in Q^{ctd}(k,A,E)$. By definition, we can write Φ as $\Phi=\forall X\exists YF$, where $X=\{x_1,...,x_A\}$, $Y=\{y_1,...,y_E\}$ are two disjoint sets of atoms and F is a k-CNF formula of a special type. Namely, F is a conjunction of $2A$ clauses, $F=C_1\wedge\ldots\wedge C_{2A}$, and for each i, $1\leq i\leq 2A$, $C_i=l_{i1}\vee\ldots\vee l_{ik}$, where l_{i1} is a literal over X and $l_{i2},\ldots,l_{ik}$ are literals over Y.

We now define $C_i^Y=l_{i2}\vee\ldots\vee l_{ik}$ and $F^Y=C_1^Y\wedge\ldots\wedge C_{2A}^Y$. Moreover, for every interpretation I of X we define $F|_I=\bigwedge\{C_i^Y\mid C_i\in F \text{ and } I\not\models l_{i1}\}$.

Let us assume that Φ is selected from $Q^{ctd}(k,A,E)$ uniformly at random. By the definition of the model $Q^{ctd}(k,A,E)$, F^Y can be regarded as selected from $C(k-1,2A,E)$ uniformly at random and, for each interpretation I of X, $F|_I$ can be regarded as selected uniformly at random from $C(k-1,A,E)$.

To derive an upper bound on $\mu_u^{ctd}(k)$, let us fix an interpretation I of X. Clearly, if $F|_I$ is unsatisfiable, then Φ is false. Let us choose any real $\rho>\rho_u(k-1)$. If $A/E\geq\rho$, the probability that $F|_I$ is unsatisfiable converges to 1 as E approaches infinity and, consequently, the probability that Φ is false converges to 0 as E approaches infinity. It follows that if $\rho>\rho_u(k-1)$ and $A/E\geq\rho$, the probability that Φ is true converges to 0 as E approaches to infinity. Since ρ is an arbitrary real such that $\rho>\rho_u(k-1)$, $\mu_u^{ctd}(k)\leq\rho_u(k-1)$ follows.

To prove the lower bound, we observe that if the formula F^Y is satisfiable, then for every interpretation I of X, the formula $F|_I$ is satisfiable or, equivalently, Φ is true. Let ρ be a positive real number such that $\rho<\frac{\rho_l(k-1)}{2}$. By the definition of $\rho_l(k-1)$, if we assume that $A/E\leq\rho$, that is, $2A/E\leq 2\rho<\rho_l(k-1)$, the probability that F^Y is satisfiable converges to 1 as E approaches infinity. Thus, the probability that Φ is true converges to 1 as E approaches to infinity. It follows that $\mu_l^{ctd}(k)\geq\frac{\rho_l(k-1)}{2}$. □

It follows that as ρ grows, the properties of $Q^{ctd}(k,\lfloor\rho E\rfloor,E)$ change. For small values of ρ, randomly selected QBFs are almost surely true. As ρ grows beyond $\mu_l^{ctd}(k)$ the proportion of false formulas grows until, eventually, when ρ grows beyond $\mu_u^{ctd}(k)$, the formulas in the model are almost surely false. Clearly, $\mu_l^{ctd}(k)\leq\mu_u^{ctd}(k)$. As in the other cases, the question whether $\mu_l^{ctd}(k)=\mu_u^{ctd}(k)$ is open.

3.2. The multi-component models

Let $\mathscr{F}$ be a class of propositional formulas (or a model of a random formula). By t-$\mathscr{F}$ we denote the class of all disjunctions of t formulas from $\mathscr{F}$ (or a model generating disjunctions of random formulas from $\mathscr{F}$). Similarly, if $\mathscr{Q}$ is a class (model) of QBFs of the form $\forall X\exists YF$, where $F\in\mathscr{F}$, we write t-$\mathscr{Q}$ for the class (model) of all QBFs of the form $\forall X\exists YF$, where $F\in t$-$\mathscr{F}$. We refer to models t-$\mathscr{F}$ and t-$\mathscr{Q}$ as *multi-component*. For QBFs we also consider the dual model to t-$\mathscr{Q}$, based on conjunctions of t DNF formulas. That dual multi-component model of QBFs gives rise to a multi-component model of disjunctive logic programs via the Eiter-Gottlob translation. In all cases, when we define multi-component models we assume that the underlying propositional formulas and QBFs are equally likely.

We first observe that the multi-component model t-$C(k,n,m)$ has similar satisfiability properties as $C(k,n,m)$, and that the phase transition regions in the two models are closely related as explained by the theorem below. To state this theorem we define $p_t(k,n,m)$ to be the probability that a random formula in t-$C(k,n,m)$ is SAT (we note that, in particular, $p_1(k,n,m)=p(k,n,m)$).

Theorem 3. *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 $\rho>\rho_u(k)$, $\lim_{n\to\infty} p_t(k,n,\lfloor\rho n\rfloor)=0$.*

Proof. As we discussed earlier, we can assume that our model actually generates ordered t-tuples of $C(k,n,m)$ formulas (they represent disjunctions of t formulas from the model $C(k,n,m)$, where repetitions of disjuncts are allowed, and disjunctions differing in the order of disjuncts are viewed as different). Thus, it is clear that

$$p_t(k,n,m) = 1-(1-p(k,n,m))^t. \tag{1}$$

It follows that for every fixed t, and every ρ,

$$\lim_{n\to\infty} p_t(k,n,\lfloor\rho n\rfloor) = 0 \quad \text{if and only if} \quad \lim_{n\to\infty} p(k,n,\lfloor\rho n\rfloor) = 0$$

and

$$\lim_{n\to\infty} p_t(k,n,\lfloor\rho n\rfloor) = 1 \quad \text{if and only if} \quad \lim_{n\to\infty} p(k,n,\lfloor\rho n\rfloor) = 1.$$

Thus, the assertion follows. □

Theorem 3 implies that 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.

Theorem 3 describes the situation when t is fixed and n is large. Specifically, when n is fixed and t grows, the identity (1) shows that the region of the transition from SAT to UNSAT shifts to the right. (In contrast, we recall that by Theorem 3, if t is fixed and n grows, the phase transition region shifts to the left.) Our experimental study discussed later provides results consistent with this theoretical analysis. Moreover, our experiments also show that the phase transition region is where the hard formulas are located, and that hardness depends significantly on 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, $q_1(a,e;A,E;m) = q(a,e;A,E;m)$). Using Theorem 1 and reasoning as above, we can prove that the phase transition regions for different values of t coincide (and coincide with the phase transition region in the Chen-Interian model).

Theorem 4. *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 $n = A+E$).*

Proof. For the proof, we will assume that the model t-$Q(a,e;A,E;m)$ generates QBFs with matrices that are ordered t-tuples of formulas generated from the model $C(a,e;A,E;n)$. As before, we have that for each fixed positive integer t,

$$q_t(a,e;A,E;\lfloor\nu n\rfloor) = 1-(1-q(a,e;A,E;\lfloor\nu n\rfloor))^t. \tag{2}$$

This identity implies the theorem in the same way that (1) implies the assertion of Theorem 3. □

The experimental results on satisfiability of QBFs from t-$Q(a,e;A,E;m)$, which we present in Section 5, agree with our theoretical analysis; we will also see there the easy-hard-easy pattern and a strong dependence of hardness on t.

Finally, we considered the multi-component model t-$Q^{ctd}(k,A,E)$, that is, a multi-component model whose t components are formulas from the controlled model $Q^{ctd}(k,A,E)$. Thus, the model t-$Q^{ctd}(k,A,E)$ incorporates both ideas we introduced in the paper. As in the other two cases, it is easy to derive the existence of the phase transition region and its invariance with respect to t from the results on the underlying single-component model which, for the controlled model are given in Theorem 2. Let $q_t^{ctd}(k,A,E)$ denote the probability that a random formula in t-$Q^{ctd}(k,A,E)$ is true.

Theorem 5. *For every integer $t \geq 1$, if $\rho < \mu_l^{ctd}(k)$, then $\lim_{E\to\infty} q_t^{ctd}(k,\lfloor\rho E\rfloor,E) = 1$, and if $\rho > \mu_u^{ctd}(k)$, $\lim_{E\to\infty} q_t^{ctd}(k,\lfloor\rho E\rfloor,E) = 0$,*

Proof. For the proof, we will assume that the model t-$Q^{ctd}(k,\lfloor\rho E\rfloor,E)$ generates QBFs with matrices that are ordered t-tuples of formulas generated from the model $C^{ctd}(k,\lfloor\rho E\rfloor,E)$ (disjunctions of t formulas from the model, where repetitions of disjuncts are allowed and the order matters). As in the two classes of multi-component models we considered above, we have that for each fixed positive integer t,

$$q_t^{ctd}(k,\lfloor\rho E\rfloor,E) = 1-(1-q^{ctd}(k,\lfloor\rho E\rfloor,E))^t. \tag{3}$$

This identity, when combined with Theorem 2, implies the assertion. □

We also studied the model $t\text{-}Q^{ctd}(k,A,E)$ experimentally. The results are reported in Section 5. As in other cases, they agree with the predictions of the theoretical anaylysis above. Importantly, they show that formulas from the model $t\text{-}Q^{ctd}(k,A,E)$ can be "super-hard." That is, with CNF formulas as components of a multicomponent model, we can generate formulas that are much harder than those generated from any other model considered before.

4. Random Disjunctive Programs

Models of random QBFs imply models of random disjunctive logic programs. This is important as disjunctive logic programs increase the expressive power of answer set programming, posing, at the same time, a computational challenge [10, 24].

We now review some basic notions and notation concerning logic programs. Let Σ be a set of propositional atoms. A *disjunctive rule* (over Σ) is an expression of the form

$$a_1 \vee \ldots \vee a_l \leftarrow b_1, \ldots, b_m,\ not\ c_1, \ldots,\ not\ c_n, \tag{4}$$

where all a_i, b_j, and c_k are atoms from Σ, l,m,n are non-negative integers such that $l+m+n>0$, and *not* represents *negation-as-failure*, also known as *default negation*. Let r stand for the rule (4). Then, the set $H(r) = \{a_1, \ldots, a_l\}$ is the *head* of r, while $B^+(r) = \{b_1, \ldots, b_m\}$ and $B^-(r) = \{c_1, \ldots, c_n\}$ are the *positive body* and the *negative body* of r, respectively. A *(disjunctive logic) program* P is a finite set of disjunctive rules. Any set $I \subseteq \Sigma$ is an *interpretation*; it is a *model* of a program P if for each rule $r \in P$, $I \cap H(r) \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. A model M of P is *minimal* if there is no model M' of P such that $M' \subset M$. A model M is an *answer set* (or a *stable model*) of P if M is also a minimal model of the *Gelfond-Lifschitz reduct* [25] of P with respect to M, denoted by P^M. The reduct P^M is the program consisting of rules $a_1 \vee ... \vee a_l \leftarrow b_1, ..., b_m$, obtained from rules $r \in P$ of form (4), such that $B^-(r) \cap M = \emptyset$.

Our approach to design models of random disjunctive programs is based on the translation from QBFs to programs due to Eiter and Gottlob [20]. The Eiter-Gottlob translation works on QBFs $\Phi = \exists X \forall Y G$, where G is a DNF formula.

To describe the translation, let us assume that $X = \{x_1, \ldots, x_E\}$, $Y = \{y_1, \ldots, y_A\}$ and $G = D_1 \vee \ldots \vee D_m$, where $D_i = L_{i,1} \wedge \ldots \wedge L_{i,k_i}$ 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 $y \in Y$
$w \leftarrow \sigma(L_{i,1}), \ldots, \sigma(L_{i,k_i})$	for each D_i, $i = 1, \ldots, m$
$w \leftarrow not\ w$	

Theorem 6 (Eiter and Gottlob [20]). *Let Φ be a QBF $\exists X \forall Y G$, where G is a DNF formula over $X \cup Y$. Then Φ is true if and only if P_Φ has an answer set.*

We will use this result to derive models of disjunctive logic programs from the models of QBFs that we considered above. We recall that these models consist of formulas of the form $\forall X \exists Y F$, where F is a CNF formula. Before we can apply the Eiter-Gottlob translation, we have to transform these models (their formulas) into their dual counterparts.

To this end, for a CNF formula F, we denote by $\overline{F}$ the formula obtained from $\neg F$ by applying the De Morgan laws (thus, transforming $\neg F$ into DNF). Extending the notation, for each QBF $\Phi = \forall X \exists Y F$, where F is a CNF formula, we write $\overline{\Phi}$ for the QBF $\exists X \forall Y \overline{F}$. Clearly, Φ is true if and only if $\overline{\Phi}$ is false (or equivalently, Φ is false if and only if $\overline{\Phi}$ is true).

Corollary 1. *Let Φ be a QBF $\forall X \exists Y G$, where G is a CNF formula over $X \cup Y$. Then Φ is false if and only if $P_{\overline{\Phi}}$ has an answer set.*

Given a model (set) of QBFs of the form $\forall X \exists Y F$, where F is a CNF formula, the mapping $\Phi \mapsto \overline{\Phi}$ transforms the model into its dual, consisting of QBFs with a DNF formula in the matrix. To these formulas we can apply the Eiter-Gottlob translation, thus obtaining a model (set) of disjunctive logic programs. By Corollary 1, this model has the same satisfiability properties as the original QBF model modulo the switch between true and false.

We now define $\overline{Q}(e,a;E,A;m) = \{\overline{\Phi}\colon \Phi \in Q(e,a;E,A;m)\}$. The model (set) $\overline{Q}(e,a;E,A;m)$ is the dual to the Chen-Interian model $Q(e,a;E,A;m)$. Applying the Eiter-Gottlob translation $\Psi \mapsto P_\Psi$ to QBFs $\Psi \in \overline{Q}(e,a;E,A;m)$, yields a model (set) of disjunctive logic programs, which we denote by $D_{dlp}(e,a;E,A;m)$. It follows from our comments after Corollary 1 that the theoretical results we obtained for the Chen-Interian model $Q(e,a;E,A;m)$ apply directly to the model $D_{dlp}(e,a;E,A;m)$ (modulo the switch between true and false).

Next, we define $\overline{Q^{ctd}}(k,E,A) = \{\overline{\Phi}\colon \Phi \in Q^{ctd}(k,E,A)\}$. The model $\overline{Q^{ctd}}(k,E,A)$ is dual to our controlled model of QBFs. By applying the Gottlob-Eiter translation to QBFs in $\overline{Q^{ctd}}(k,E,A)$, we obtain the model (set) of disjunctive logic programs, which we denote by $D^{ctd}_{dlp}(k,E,A)$. As before, by our comments following Corollary 1, the models $Q^{ctd}(k,E,A)$ and $D^{ctd}_{dlp}(k,E,A)$ have the same satisfiability properties (modulo the switch between true and false).

4.1. Multi-component models of disjunctive logic programs

The translation proposed by Eiter and Gottlob can be extended to QBFs of the form $\Phi = \exists X \forall Y G$, where $G = G_1 \wedge \ldots \wedge G_t$ and each G_i is a DNF formula. The translation is similar, except that we need t additional variables $w_1,\ldots,w_t$ to represent DNF formulas G_i. The translation consists of rules

$z \vee z'$	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 $w \leftarrow \mathit{not}\ w$	

that form the *fixed* part of the translation, and its *core* consisting of Horn rules

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

where $h = 1,\ldots,t$, and the rules with the head w_h are obtained from the formula G_h just as in the original Eiter-Gottlob translation (except that w_h is now used as the head and not w). In fact, in the case when $t = 1$ the program above coincides with the result of the Eiter-Gottlob translation modulo a rewriting, in which we eliminate the rule $w \leftarrow w_1$ and replace w_1 in the head of each rule in the core with w.

Extending the earlier notation, we denote the program described above by P_Φ.

Theorem 7. *Let $\Phi = \exists X \forall Y (G_1 \wedge \ldots \wedge G_t)$, where each G_i is a DNF formula. Then Φ is true if and only if P_Φ has an answer set.*

Proof. The proof follows by observing that w is derived if and only if $w_1,\ldots,w_t$ are together in the same answer set. For each $i = 1,\ldots,t$, w_i occurs in an answer set if and only if the formula corresponding to the i-th component is satisfiable. Thus, the thesis follows by a trivial adaptation of the argument used in Theorem 3 of [20] by Eiter and Gottlob. □

We can now derive multi-component models of disjunctive logic programs from the multicomponent models of QBFs. The basic idea is the same as before. A multi-component model of QBFs gives rise to its dual via a transformation $\Phi \mapsto \overline{\Phi}$ (it consists of negating Φ and applying De Morgan laws). Next, the translation above transforms QBFs from that dual model into disjunctive programs, yielding the corresponding multi-component model of programs. We apply this approach to two multi-component models of QBFs we considered in this paper: t-$Q(e,a;E,A;m)$ and t-$Q^{ctd}(k,E,A)$. We denote the corresponding models of disjunctive logic programs by t-$D_{dlp}(e,a;E,A;m)$ and t-$D^{ctd}_{dlp}(k,E,A)$.

Corollary 2. *Let $\Phi = \exists X \forall Y F$, where $F \in t$-$Q(e,a;E,A;,m)$ or $F \in t$-$Q^{ctd}(k,E,A)$. Then, Φ is false ($\overline{\Phi}$ is true) if and only if $P_{\overline{\Phi}}$ has an answer set.*

By Corollary 2, the models t-$Q(e,a;E,A;,m)$ (t-$Q^{ctd}(k,E,A)$, respectively) and t-$D_{dlp}(e,a;E,A;m)$ (t-$D^{ctd}_{dlp}(k,E,A)$, respectively) have the same satisfiability properties (modulo the switch between true and false).

5. Empirical analysis

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

5.1. 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 included GLUCOSE 4.0 [8]; LINGELING, version of 2015 [9]; and KCNFS, version of SAT'07 competition [17]. The QBF solvers included BQ-CEGAR (a combination of *bloqqer* preprocessor [28] and *ghostq* [33] solver from QBF gallery 2014); AIGSOLVE [41]; RAREQS [29], version 1.2 from QBF competition 2016; and AQUA-S2V[3]. Finally, the two ASP solvers we used in experiments were CLASP 3.1.3 [22] and WASP 2.1 [4], both paired with *gringo* 4.5.3 [21]. 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*.

To support experiments, we developed a tool in Java to generate random CNF formulas from $C(k,n,m)$, QBFs from $Q(a,e;A,E;m)$ and $Q^{ctd}(k,A,E)$, and programs from $D_{dlp}(e,a;E,A;m)$ and $D_{dlp}(k,E,A)$ ("dual" to QBFs from $Q(e,a;E,A;m)$ and $Q^{ctd}(k,E,A)$). For each class $\mathscr{C}$ of formulas and programs listed, our tool generates also formulas (programs) from the corresponding multicomponent model t-$\mathscr{C}$.

Formulas and QBFs generated according to the multi-component models t-$C(k,n,m)$, t-$Q(a,e;A,E;m)$ and t-$Q^{ctd}(k,A,E)$, where $t > 1$, are non-clausal or have non-clausal matrices (in the case of QBFs). As they do not adhere to the (Q)DIMACS format required by SAT/QBF solvers, the generator transforms non-clausal formulas to CNF using the Tseitin transformation [43]. That transformation introduces fresh auxiliary variables (while replacing binary subformulas) and new clauses (modeling the equivalence of each replacement) to obtain a CNF formula that is equisatisfiable to the original one. The Tseitin transformation is efficient, since it only causes a linear growth in size (whereas doing the same normalization via distributivity laws may lead to an exponential blow-up).[4] Interestingly, the logic programs in the models t-$D_{dlp}(e,a;E,A;m)$ and t-$D_{dlp}(k,E,A)$ have a much simpler structure than the corresponding Tseitin-transformed formulas from the "dual" models t-$Q(e,a;E,A;m)$ and t-$Q^{ctd}(k,E,A)$). As can be seen from the translation, these programs need new variables only to represent each of the t components (disjuncts) of the matrix formula.

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 [30, 38], and the other one with the disjunctive logic program corresponding to Φ in the ASPCore 2.0 syntax [12]. As discussed in the previous section, since the programs are generated from the negations of the QBFs in our random QBF models, 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 (the plots are symmetric to each other). In all the experiments the results are averaged over 128 samples of the same size.

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. The generator used in the experiments is publicly available at https:/www.mat.unical.it/ricca/RandomLogicProgramGenerator.

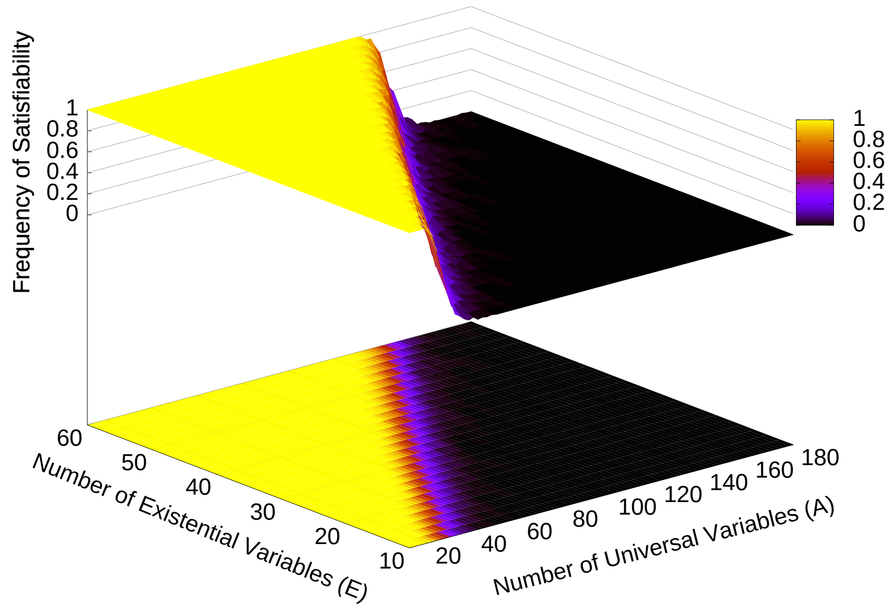
5.2. Behavior of the controlled model

We first study the satisfiability and hardness of formulas and corresponding programs generated according to the controlled model. We generated QBF instances from the model $Q^{ctd}(4,A,E)$ and program instances from the dual model $D_{dlp}(4,A,E)$) for the parameters E and A ranging over [10..60] and [20..180], respectively (consequently, the number of clauses ranges from 40 to 360).

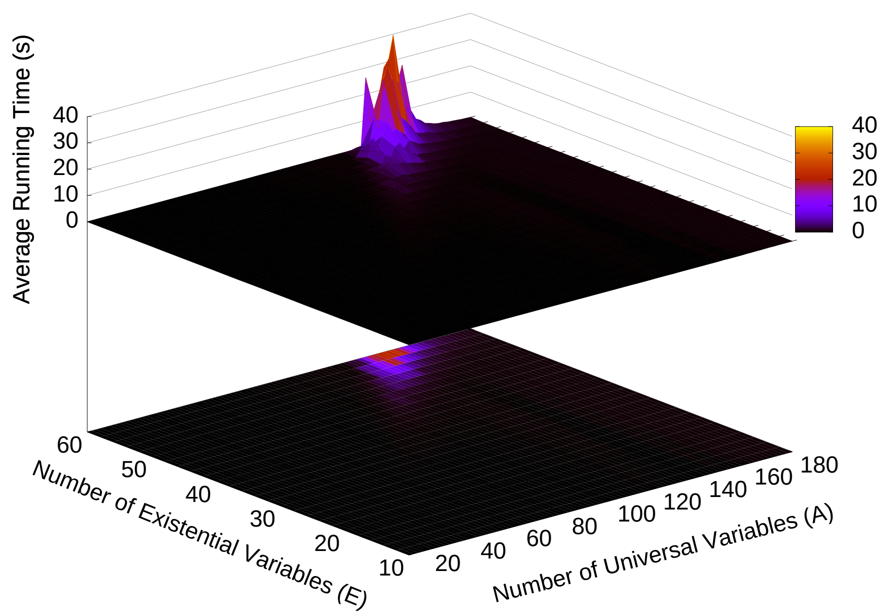
Figure 1a shows the satisfiability results for the model $Q^{ctd}(4,A,E)$. The picture for $D_{dlp}(4,A,E)$ is dual (symmetric with respect to the plane given by the frequency of satisfiability equal to $1/2$); the results

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

[4] For this reason the Tseitin transformation is employed very often in real-world applications of SAT/QBF. Actually, many formulas used in SAT and QBF competitions [30, 38] come from applying it to non-normal form inputs suggested by problem statements.

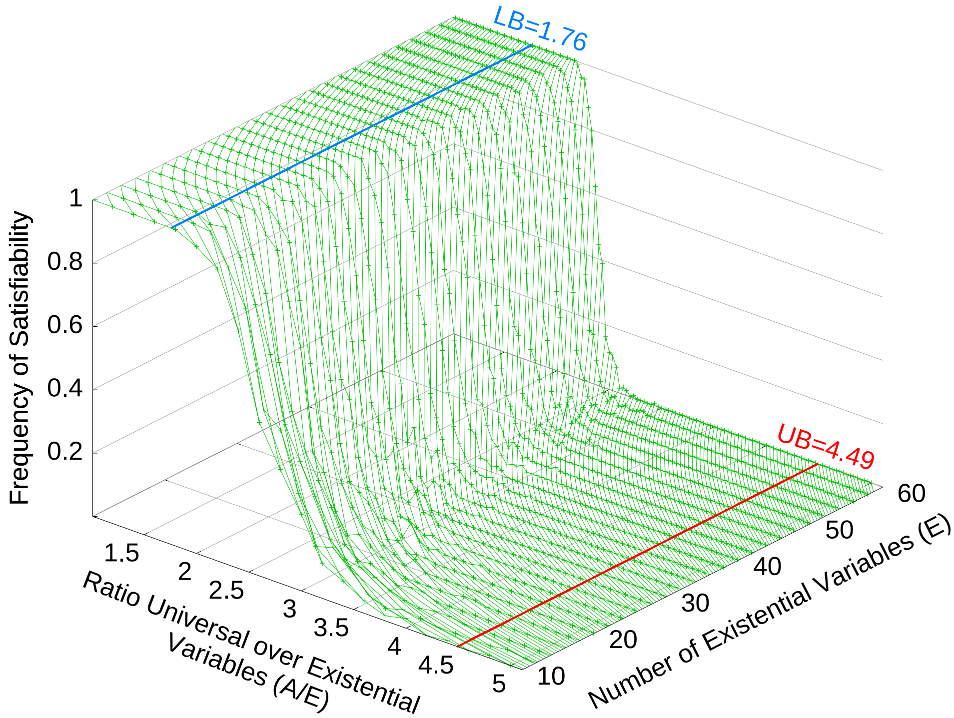


(a) Phase transition (Controlled)

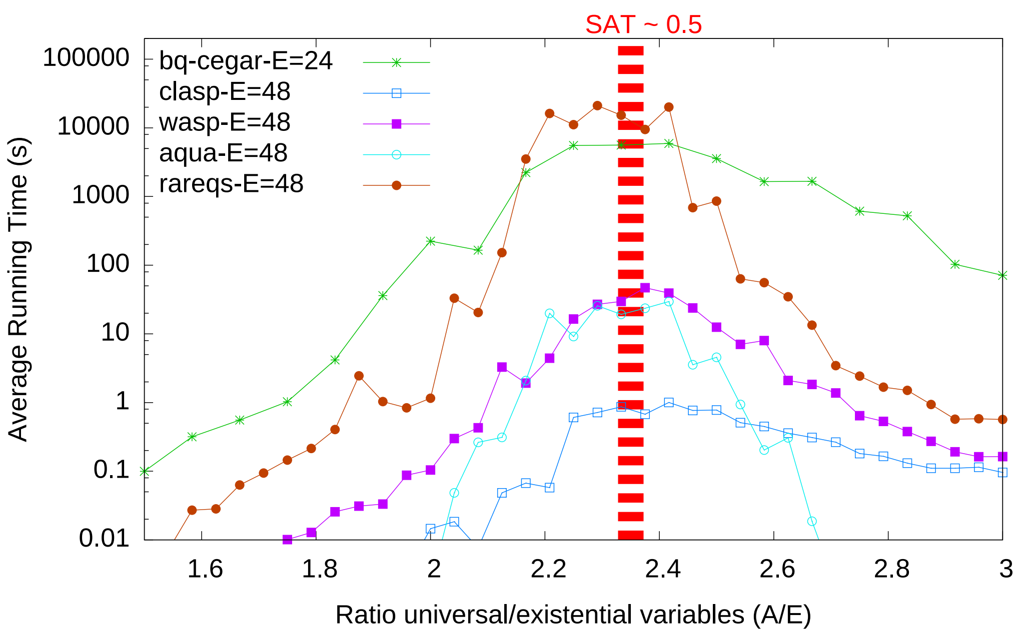


(b) Hardness (Controlled)

Figure 1: Behavior of controlled model: phase transition and hardness.



(a) Bounds on Satisfiability (Controlled)



(b) Solver Independence (Controlled)

Figure 2: Behavior of controlled model: bounds and solver independence.

we show were in fact obtained by running CLASP on programs from $D_{dlp}(4,A,E)$ and adapted to the case of $Q^{ctd}(4,A,E)$). The gradient of colors ranging from yellow (QBF true) to black (QBF false) helps to identify the phase transition region, which is also projected on the A-E plane below. We observe that phase transitions occur for a specific value of the ratio between universal and existential variables, specifically, for $A/E \simeq 2.37$. A different perspective on the same data is presented in Figure 2a, where the frequency of satisfiability is depicted with respect to the ratio A/E, and where the two straight lines show the bounds predicted by Theorem 2, assuming the bounds for satisfiability and unsatisfiability of 3-CNF formulas to be respectively 3.52 [31] and 4.49 [18], i.e., $\mu_l^{ctd}(4) \geq \frac{3.52}{2} = 1.76$ and $\mu_u^{ctd}(4) \leq 4.49$. We observe that the transition sharpens when the number of variables grows, and the transition occurs within the bounds predicted by the theoretical results.

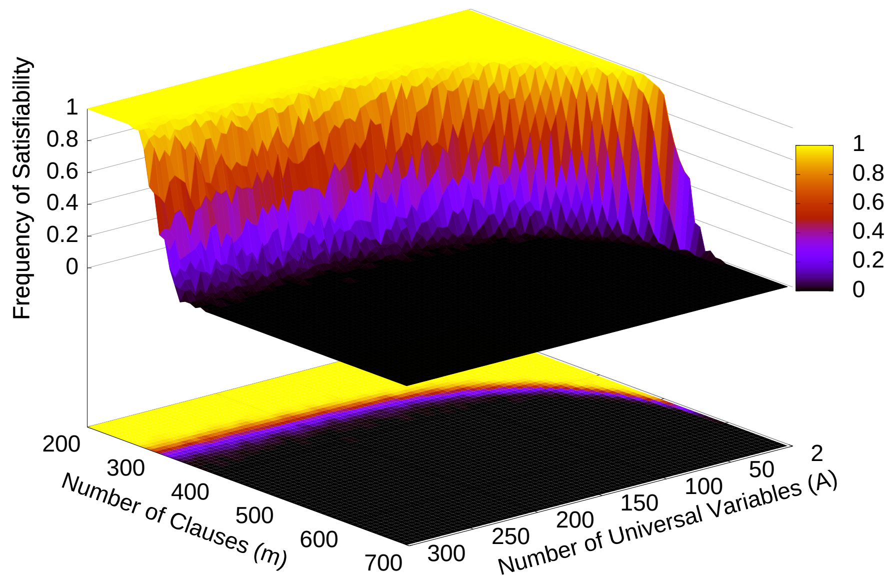
To study the hardness of formulas, the average running times are plotted in Figure 1b. Here the gradient of colors ranging from black (basically instantaneous execution) to yellow (the maximum average running time) helps to identify the hardness region. As before the region is also projected on the A-E plane below. As expected hardness arises around the phase transition region and grows with the number of variables. To provide evidence that the hardness of the controlled model is independent of the solver used, Figure 2b plots the average execution times when running two QBF solvers (RAREQS and AQUA-S2V) and two ASP solvers (CLASP and WASP) on formulas/programs implied by formulas from $Q^{ctd}(4,A,E)$ with 48 existential variables, and the QBF solver BQ-CEGAR on formulas with 24 existential variables (for that solver, we had to decrease the size of formulas to ensure termination within a reasonable time). We note that all solvers find hard formulas in the same region, and the maximum hardness coincides with the transition zone marked by the red vertical strip. No data is reported in Figure 2b for AIGSOLVE because it terminated abruptly in some instances (throwing *std::bad_alloc*) and in some other we had to kill the process after 15 days of execution. (This behavior is probably due to a memory access problem.)

5.3. Controlled vs Chen-Interian model

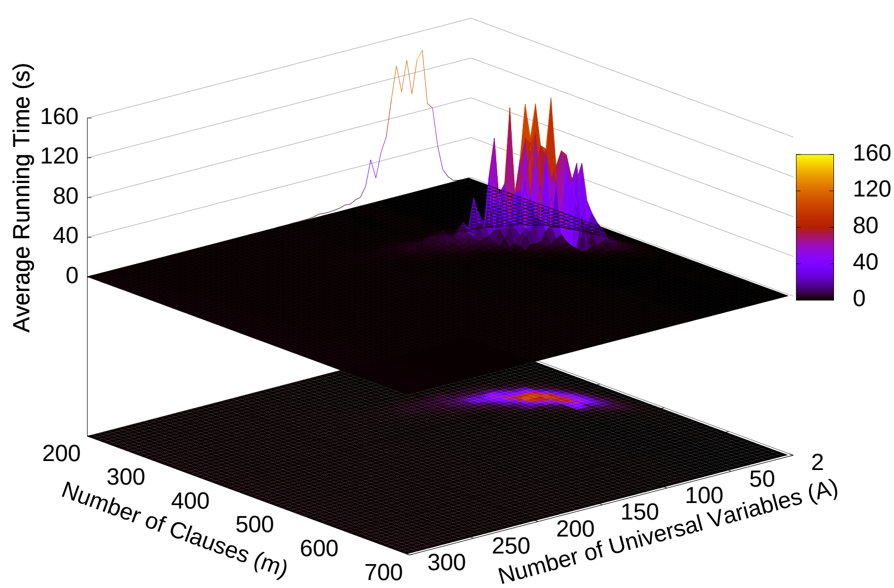
We now compare the controlled and the Chen-Interian models with respect to the hardness of formulas having the same number of variables.

We start by presenting results on the behavior of the Chen-Interian model $Q(a,e;A,E;m)$, where we set $a=1$, $e=3$, and $E=70$, and vary the number A of universal variables over the range $[2..300]$ and the number m of clauses over the range $[200..700]$. These results are shown in Figure 3. They confirm and extend the findings by Chen and Interian [13]. As before the gradient of colors in Figure 3, ranging from black to yellow, outlines the phase transition and the easy-hard-easy pattern. The surface is also projected onto the A-m plane for an alternative visualization. For every value of A (in fact, for every value of the ratio A/E, since we fixed the value of E to 70), as we grow m we observe the phase transition. The place where this phase transition occurs depends on A (more generally, on the ratio A/E; but in our experiments E is fixed). For each value of A (more precisely, for each value of A/E), the hardest formulas are located around the phase transition area, as evidenced by Figure 3(b). The behavior presents there only for the values of A of up to about 85; for higher values of A, the running times even on the formulas from the phase transition region are very small. Figure 3(b) also shows that the overall peak of hardness occurs in the phase transition region for a specific value of A or, as explained earlier, for a specific value of the ratio A/E.

Next, we compare the hardness properties of the controlled and the Chen-Interian models with the same number of existential variables, which can be viewed as a measure of the hardness of individual SAT instances that arise while solving a QBF of the form $\forall\exists F$. The graphs in Figure 4 capture the behavior of the hardness for the two models under this constraint. For the controlled model, for each value of A, the value on the corresponding hardness graph (the blue line) is obtained by averaging the solve times on formulas generated from the model $Q^{ctd}(4,A,70)$. The matrices of these formulas are 4-CNF formulas over $A+70$ variables and with $2A$ clauses. The corresponding point on the hardness graph for the Chen-Interian model is obtained by averaging the solve times on formulas generated from the model $Q(1,3;A,70;max)$, where for each A (and $E=70$), max is selected to maximize the solve times (in particular, max falls in the phase transition region for the combination of the values A and $E=70$). The matrices of these formulas are 4-CNF formulas over $A+70$ variables and max clauses.



(a) Phase transition (Chen-Interian)



(b) Hardness (Chen-Interian)

Figure 3: Chen-Interian: Phase transition and Hardness.

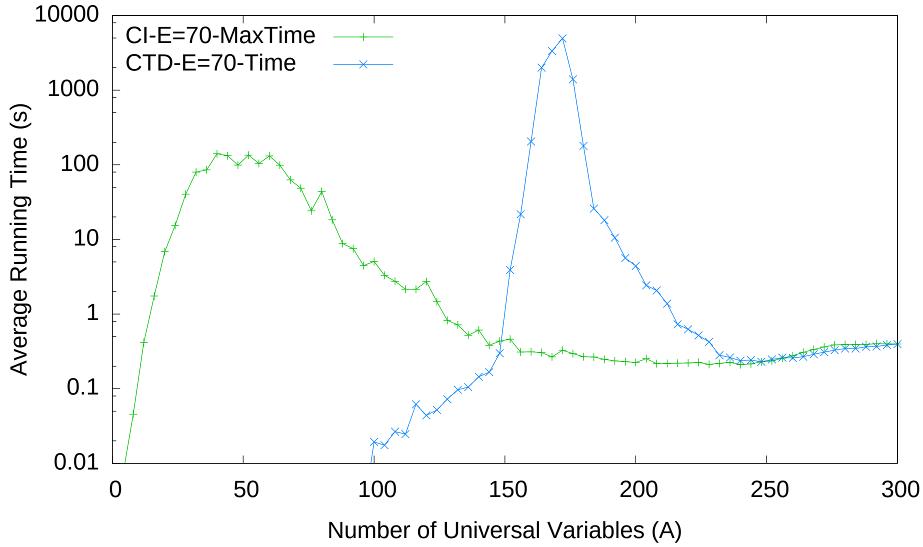


Figure 4: Comparing Chen-Interian and Controlled model: hardness comparison.

The results show that the peak hardness regions for the two models are not aligned. The hardest formulas over 70 existential variables from the Chen-Interian models have $A \approx$ 50-55 universal variables and $m = 350$ clauses. The hardest formulas over 70 existential variables from the controlled model have $A \approx 170$ and $m \approx 340$. Our results show that the hardest formulas from the controlled model are almost two orders of magnitude harder than the hardest formulas from the Chen-Interian model. On the other hand, while the hardest formulas (for a fixed value of E, here $E = 70$) in the two models have similar numbers of clauses (about 340-350), the Chen-Interian model formulas have fewer universal variables (about 50-55 versus 170 in the controlled model).

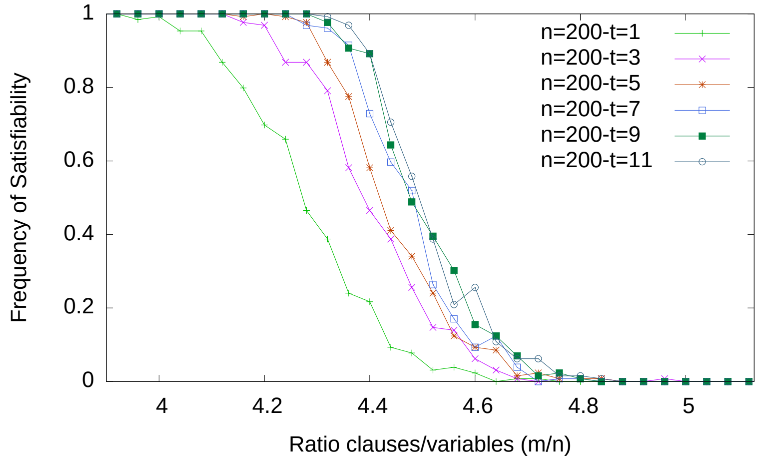
It is also useful to look at the point where the hardness of one model meets the other. It happens for $A \approx 150$. At this point, the CNF formulas that are the matrices of QBFs from the controlled model have 70 existential and about 150 universal variables, and about 300 clauses. The corresponding parameters for the formulas from the Chen-Interian model have very similar values. Indeed, the hardest formulas for the Chen-Interian model when $E = 70$ and $A = 150$ have about 300 clauses (cf. Figure 3).

To summarize, a direct comparison for the hardness of the two models is not clear cut. On the one hand, our results show that if we make the comparison for models with the same number of existential variables the points, in terms of A, in which the two model generate their hardest instances are very different. On the other hand, there is a setting (corresponding to the phase transition for the controlled model) in which the controlled model generates much harder formulas than any other setting (corresponding to a phase transition) for the Chen-Interian model.

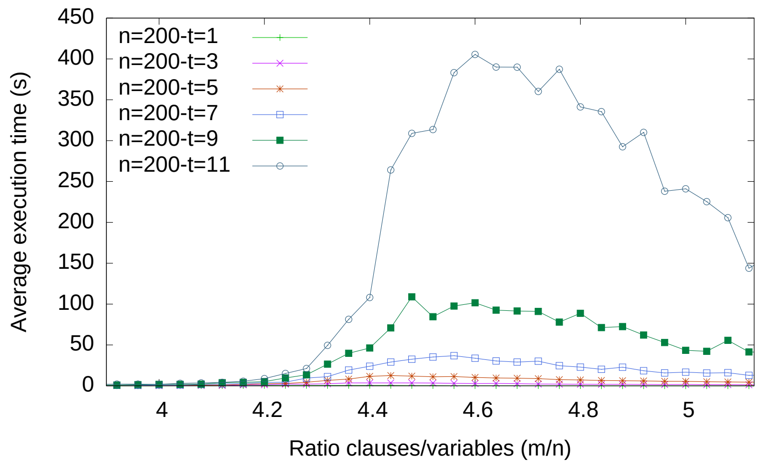
For the sake of completeness, we report that we obtained results consistent with those discussed above experimenting with other settings of existential variables and clause lengths.

5.4. Behavior of Multi-component Model

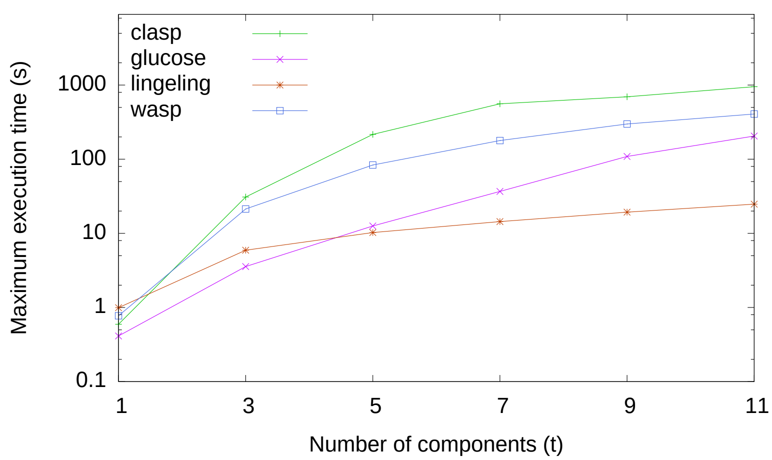
To study the satisfiability of multi-component model instances (the location of the phase transition), we considered the setting with the number of variables (propositional atoms) fixed. Figure 5a shows the results for the t component model t-$C(3,200,m)$, with $t \in \{1,3,5,7,9,11\}$. The x-axis gives the ratio of the numbers of clauses and variables ($m/200$); the y-axis shows the frequency of satisfiability, i.e., the ratio of the number of satisfiable instances to the total number of instances in each sample of the same size. 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 6a, showing the frequency of QBFs from



(a) Phase transition shift

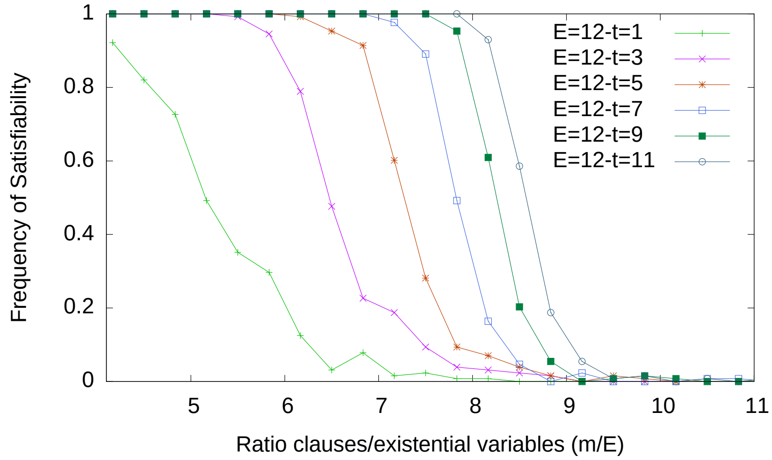


(b) Easy-hard-easy pattern - Measurements done with GLUCOSE.

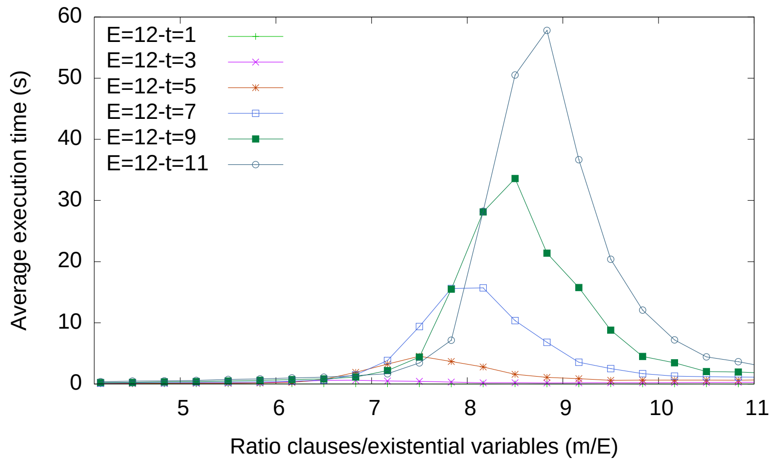


(c) Hardness

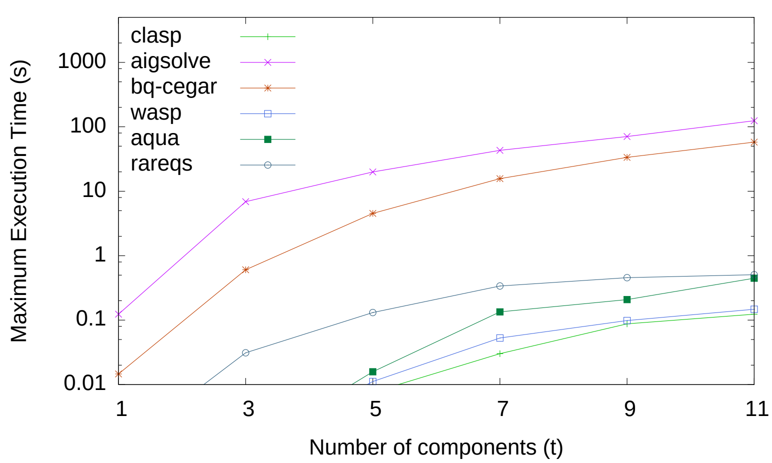
Figure 5: Behavior of the multi-component model $t\text{-}C(3,200,m)$.



(a) Phase transition shift

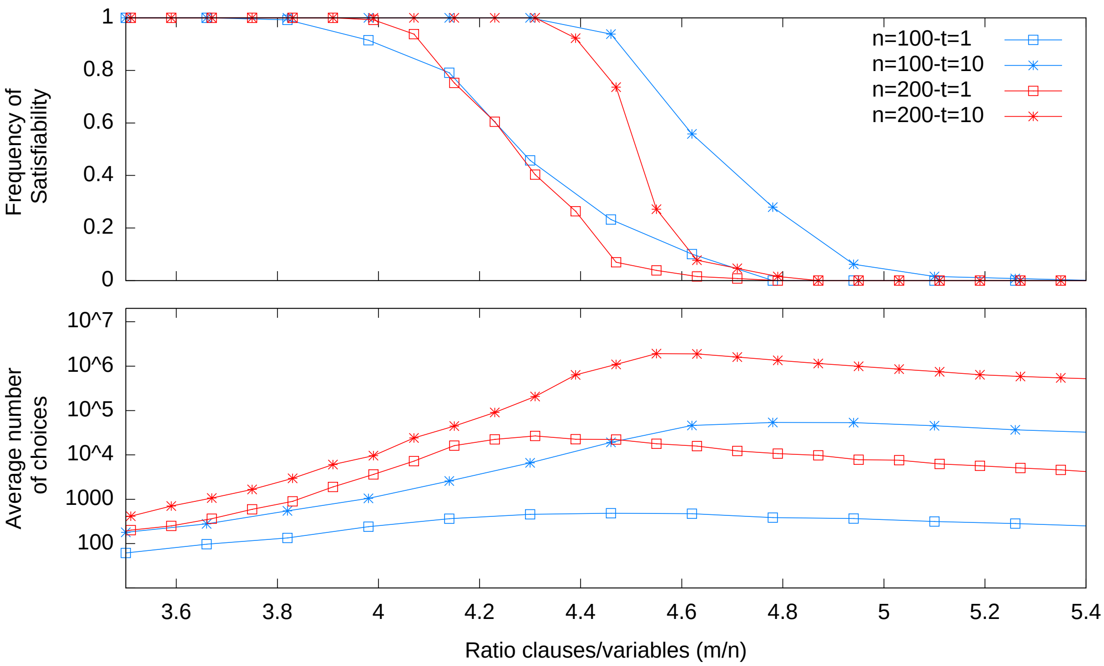


(b) Easy-hard-easy pattern - Measurements done with BQ-CEGAR.

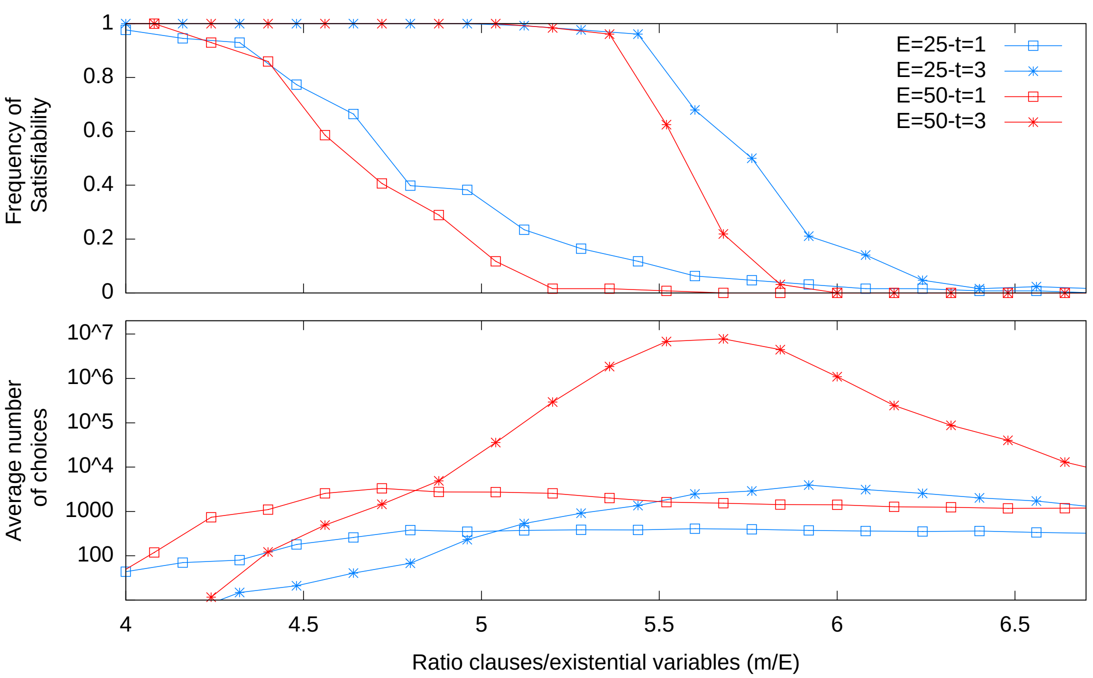


(c) Hardness

Figure 6: Behavior of the multi-component model t-$Q(1,3;24,12;m)$.



(a) Combined effect of variables and components (t-$C(3,n,m)$).



(b) Combined effect of variables and components (t-$Q(1,3;E,A;m)$).

Figure 7: Combined effect of variables and components: phase transition and hardness. Measurements done with CLASP.

t-$Q(1,3;24,12;m)$ that are true, for $t \in \{1,3,5,7,9,11\}$. The satisfiability plots obtained for logic programs from the corresponding models t-$D_{dlp}(1,3;24,12;m)$ are symmetric with respect to the line $y = 0.5$ and are not reported.

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 5b and 6b. 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 require 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 5c (for formulas) and 6c (for QBF formulas and programs). In particular, Figure 5c reports the results obtained by running GLUCOSE and LINGELING, and Figure 6c — 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 independent of the implementation or the representation. This growth is roughly at a rate that is more than quadratic with t (y-axis logarithmic).

Next, we discuss the behavior of formulas when both the number of variables and the number of components grow. Figure 7a reports on the behavior of CNF formulas with $n \in \{100,200\}$ and $t \in \{1,10\}$. Formulas with 100 variables are plotted in red, and those with 200 variables in blue. We use squares to identify graphs for formulas with one component and stars for graphs concerning formulas with ten components. Figure 7a shows that when the number of variables grows the phase transition moves to the left, and the transition becomes sharper. By Theorem 3, we expect that the bounds on (un)satisfiability do not depend on t; indeed, the right shift due to an increase in the number of components is compensated and becomes negligible when the number of variables grows. Our experiments also confirm that hardness grows with both the number of components and the number of variables. This is seen in Figure 7a, where the second (lower) plot reports the average number of choices taken by CLASP (we consider choices because 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 7b shows the same picture for QBFs in t-$Q(1,3;50,25;m)$ (plotted in red) and t-$Q(1,3;100,50;m)$ (plotted in blue), with $t \in \{1,3\}$. These results were obtained by running CLASP on the corresponding programs.

5.5. Combination of Controlled and Multi-component model

We now present results obtained by combining the two models introduced in this paper. These results focus on the effect that combining the models has on the hardness of formulas. The results are summarized in Figure 8 where a bar plot depicts the maximum average execution times (i.e., the average execution times measured evaluating the hardest instances at the phase transition) obtained by running ASP and QBF solvers on instances of models t-$Q(1,3;A,E;m)$ (multi-component with Chen-Interian) and t-$Q^{ctd}(4,A,E)$ (multi-component with controlled) varying the number of components $t \in \{1,3,5,7,9,11\}$. To obtain comparable execution times with both ASP and QBF solvers, CLASP and WASP were run on instances with $E = 24$, RAREQS and AQUA-S2V on instances with $E = 18$, whereas BQ-CEGAR and AIGSOLVE on instances with $E = 12$, and $A \in [2,120]$ and $m \in [2,300]$. Figure 8 shows histograms for each solver. The results obtained for each setting of t in t-$Q(1,3;A,E;m)$ and t-$Q^{ctd}(4,A,E)$ are reported side by side in blue and orange bars, respectively. The red horizontal line helps identifying a timeout of 24 hours, and a red bar ending with an arrow indicates that some execution required more than 24 hours. A red exclamation mark identifies abrupt termination of a solver.

We observe that, no matter the solver, the hardest instances of multi-component with controlled are at least one order of magnitude harder than the Chen-Interian-based counterparts for all settings of t. Notably, the combination of the two new models allows to generate instance that are "super-hard"; indeed instances with one component are solved in less than 0.9s and it was sufficient to set $t = 11$ to obtain instances that

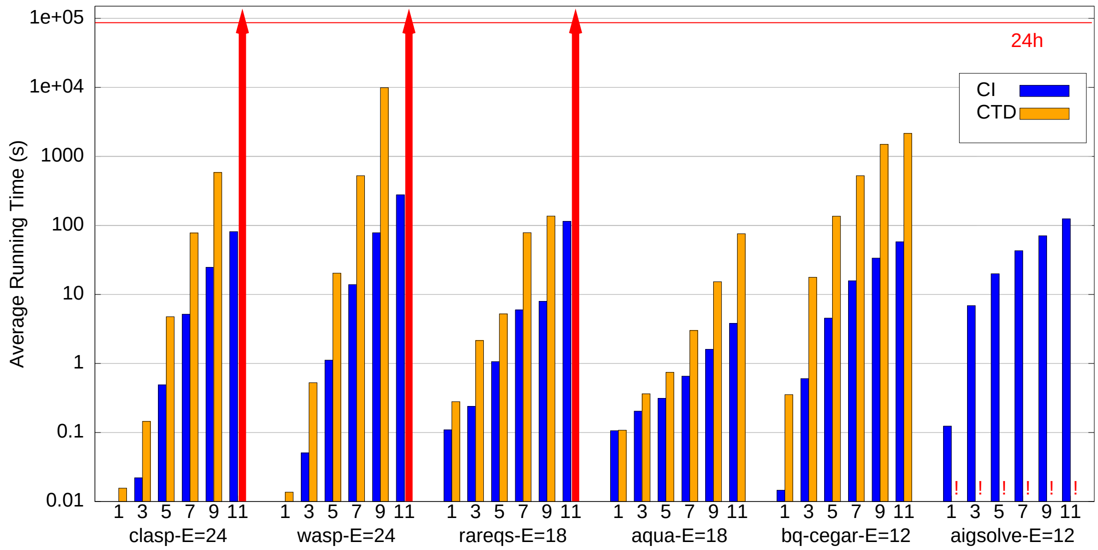


Figure 8: Comparing Chen-Interian and Controlled model: effect of components.

are more than six orders of magnitude harder to evaluate (some "controlled" instances with $E \geq 18$ could not even be solved in 24 hours).

5.6. Impact on SAT Solving

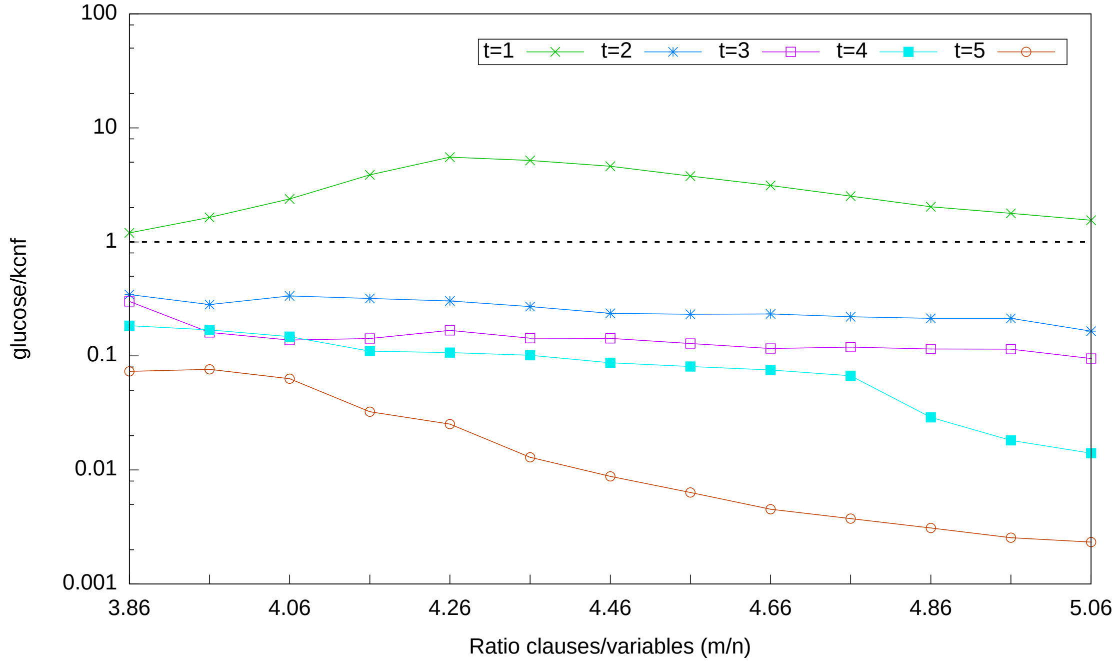
A desirable property of a random model is to generate instances that behave similarly to real-world ones [32, 6]. This similarity has been measured empirically by comparing the performance of solvers for random and industrial instances. Following Ansótegui et al. [6], we measure the ratio of the execution 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. Figure 9 shows the results for the model t-$C(3,100,m)$, while varying the number of components $t \in \{1,2,3,4,5\}$. In particular, the x-axis gives the ratio of the numbers of clauses and variables ($m/100$), and the y-axis shows GLUCOSE versus KCNFS (in Figure 9a) and LINGELING versus KCNFS (in Figure 9b).

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.

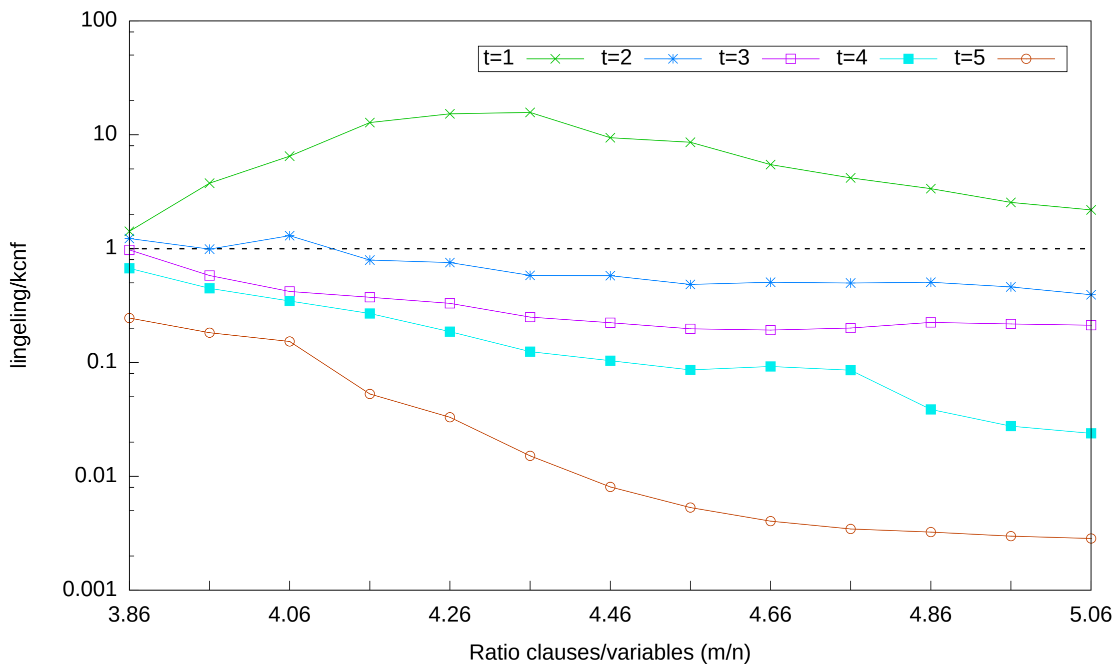
The difference between random and real-world instances is often attributed to the presence of some *hidden structure* in the latter [7]. We observed that multi-component models yield instances that are solved faster by solvers designed for real-world instances. We conjecture this is due to the component structure introduced by the model. This structure can be controlled by varying the number of components, yielding instances of varying hardness.

5.7. 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 (or known) as *specialized to random instances* in ASP and QBF Evaluations so far (cf. [12, 38] and http://www.qbflib.org). Nonetheless our models has other interesting implications for QBF and ASP solvers.

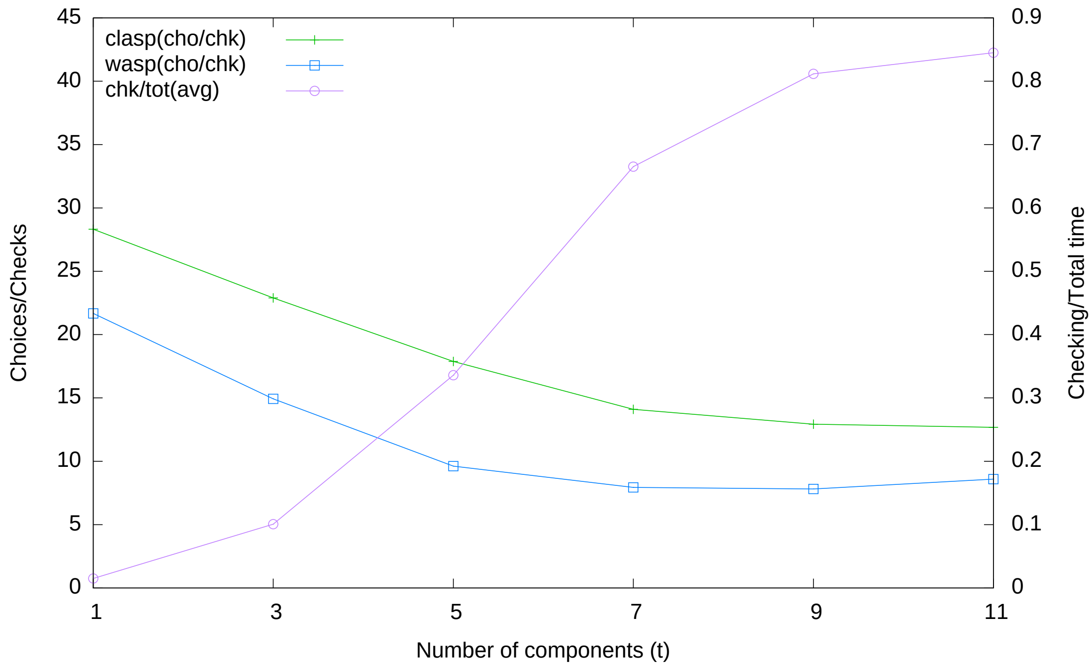


(a) CPU time ratio glucose/kcnfs

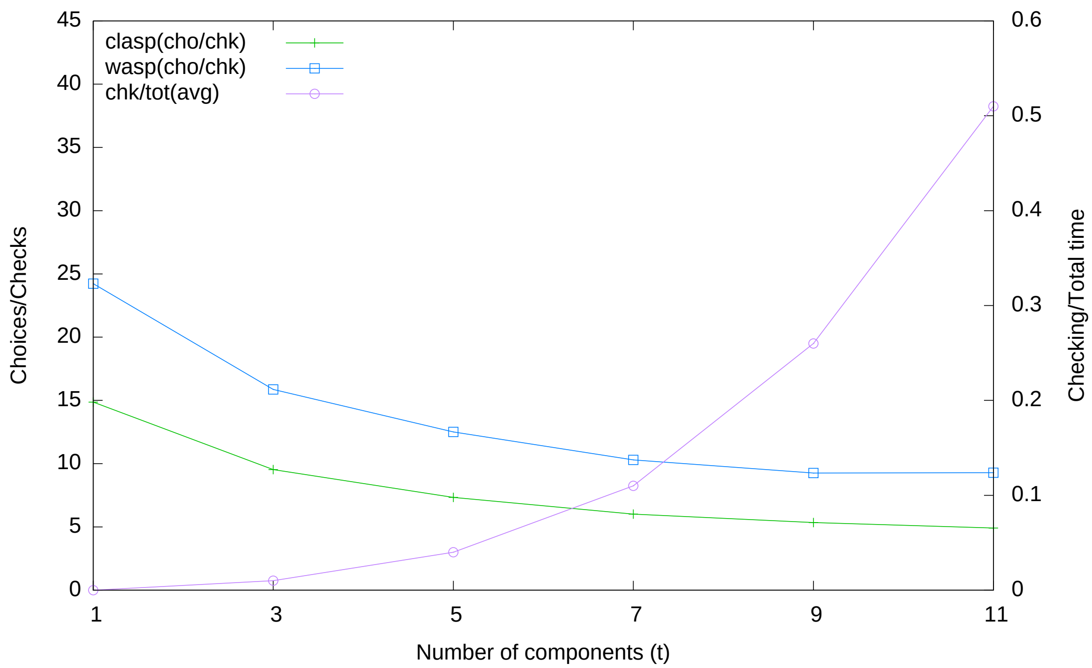


(b) CPU time ratio lingeling/kcnfs

Figure 9: Impact on SAT solving.

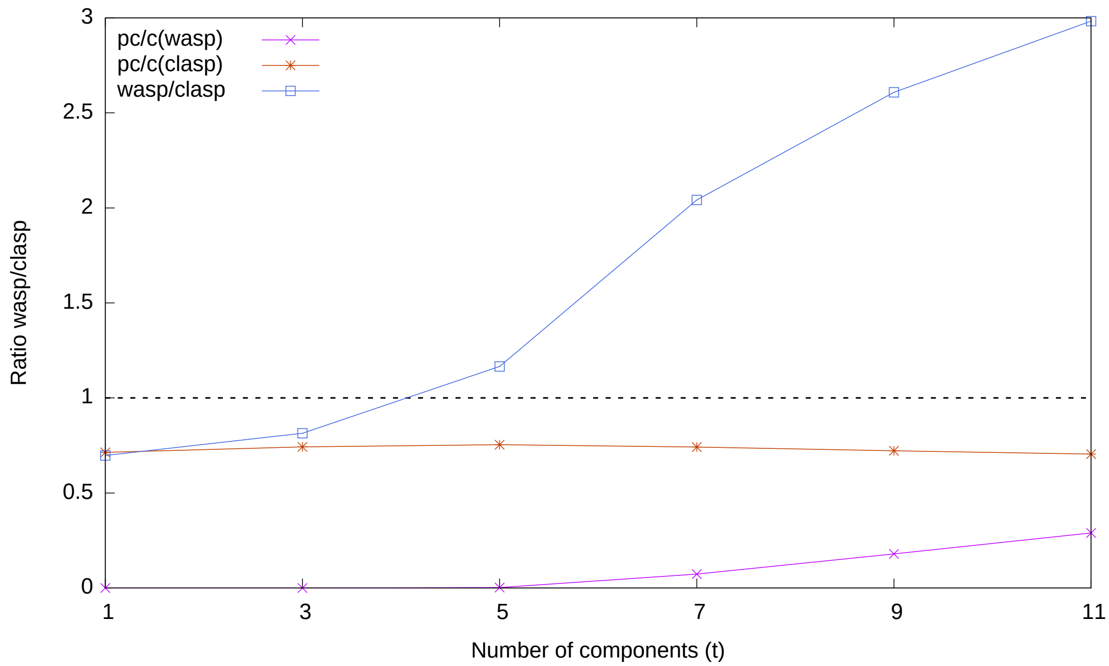


(a) Answer set computation: Multi-component

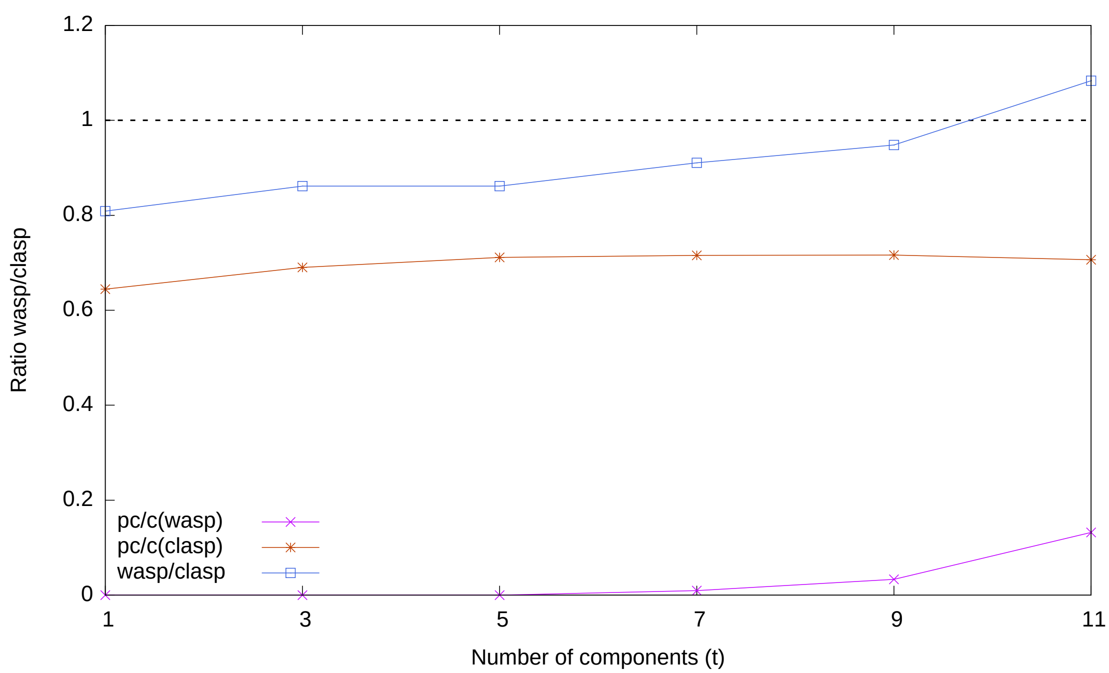


(b) Answer set computation: Multi-component Controlled Model

Figure 10: Impact on ASP solving: answer set computation.



(a) Partial checking: Multi-component



(b) Partial checking: Multi-component Controlled Model

Figure 11: Impact on ASP solving: impact of partial checking.

Impact on QBF Solving. To assess the validity of our multicomponent Chen-Interian 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.

Impact on ASP Solving. For ASP solvers, Figure 10a 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% ($t = 11$). Analogous considerations are supported by Figure 10b, which outlines the impact of the combination of controlled and multi-component model on answer set search for programs corresponding to QBF formulas t-$Q^{ctd}(4,32,16,m)$ with $t \in \{1,3,5,7,9,11\}$. Also in this case, and increase of t causes both (i) a decrease of the ratio between the number of choices made during the search phase and the number of stable model checks, and (ii) an increase of the the time spent in stable model checking and the total execution time for the solver WASP (results for CLASP are analogous). Specifically, the impact of the model checking on the total solving time grows from about about 0.05% ($t = 1$) to 51% ($t = 11$).

It is known that on usual benchmarks ASP solvers spend more time in the model search phase than in the final model checking phase [40] (this also happens on benchmarks we generated for $t = 1$). However, our multi-component models allow us to generate in a controllable way instances that put emphasis on the model checking phase.

Finally, we report some other observations that point to a potential impact of our models in detecting areas of improvement for solvers. Let us recall that the two ASP solvers we studied, CLASP and WASP, employ different strategies for stable model checking. CLASP searches for unfounded sets [23], while WASP searches for a minimal model of the program reduct [4]. Both solvers are able to check partial interpretations, but they employ different heuristics for enabling this search space pruning technique. Figure 11 compares WASP and CLASP by plotting the ratio between the time required by the two solvers for finding an answer set (labeled WASP/CLASP) and the ratio between the number of partial and total checks performed by WASP (labeled pc/c(WASP)) and CLASP (labeled pc/c(CLASP)) for the two multi-component models we studied. The results on the multi-component Chen-Interian model are in Figure 11a. The results on the multi-component controlled model are in Figure 11b. One can see that WASP is faster than CLASP when the number of components is small. When the number of components grows CLASP becomes faster and takes over. Interestingly, the deterioration in the performance of WASP corresponds to the point in which the ratio pc/c starts increasing. In contrast, CLASP maintains consistently the ratio of about 70% of the numbers of partial and total checks, and this seems to pay off for larger values of t. The results suggest that partial checking in WASP was implemented in a less efficient way then in CLASP, and it hinders WASP when the number of components grows. It seems also that for easier instances better performance could be obtained by disabling or reducing the number of partial checks as they do not seem to be essential for the performance. These observations suggest that there is space for solver developers to devise smarter heuristics to improving partial checking.

6. Conclusions

In this paper we proposed the controlled and multi-component models for random propositional formulas, and disjunctive logic programs. The models extend the well-known fixed clause length model for k-SAT, and the Chen-Interian model for QBF.

We provide theoretical bounds that predict the location of the region where the phase-transition occurs, and we present the results of an experimental analysis that confirms our theoretical findings in practice. Our experiments also show that the hardest instances are located in the phase transition region. Moreover, in the multi-component model the hardness of formulas depends significantly on the number of components.

Comparing models, we observed that the controlled model allows one to generate random instances that are much harder than those obtained with the Chen-Interian model with the same number of existential variables. Further, multi-component model allows one to generate random instances with few components that are several orders of magnitude harder than those generated with the same number of variables from the underlying "single-component" model. Finally, a combination of the two new models results in the generation of programs and formulas that are "super-hard" to evaluate.

Our experiments with different solvers and encodings gave consistent results. This supports our claim that the phenomena we observed are inherent properties of the models rather than an artifact of the solver used.

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 and programs can be controlled not only as a function of the ratio of clauses to variables, as it is the case for the earlier models; our experiments showed that the hardness of multi-component models *strongly* depends on the number of components in the model. Thus, the hardness of such models can also be controlled by varying that parameter and even a small number of components can lead to extremely hard instances. Further, in our experiments (as well as in the QBF Competition 2016) instances generated according to our models helped identifying bugs in existing solvers. Moreover, the multi-component model generates formulas that in at least one aspect are similar to instances arising in practice: they are solved better by SAT solvers specialized in industrial benchmarks than by SAT solvers specialized in random ones. This makes them useful for development and testing of solvers intended for practical applications. Finally, our models of random disjunctive programs are the first such models for that class of logic objects. Interestingly, the parameters of our random model of disjunctive programs allow us to control the role of the answer set checking phase. Thus, the model has a potential for applications in the development of ASP solvers.

Our work raises an interesting open question. The controlled model we proposed and studied stipulates that clauses in the matrix of QBFs contain exactly one universal variable. It is possible to lift this requirement. We discuss some natural extensions in Appendix A. It turns out that when the number of universal variables per clause is greater than one, the generalized model generates instances that exhibit a qualitatively different behavior. Arguably, they are easier than formulas from the corresponding Chen-Interian model.

However, a comparison to the Chen-Interian model is not clear cut, a problem we already noted for the one universal variable case. In particular, we chose to compare for hardness formulas from the two models by fixing in each model the number of existential variables to the same value. Under this constraint, the hardest formulas in the basic controlled model contain more universal variables than the hardest formulas from the Chen-Interian model. However, for the generalized controlled model and its *smooth* version, both discussed in Appendix A, this relationship reverses. The hardest formulas from the (smooth) generalized controlled model have *many* fewer universal variables than the hardest ones from the Chen-Interian model. Developing alternative perspectives on formulas from the two models might provide a better understanding of relative hardness. This is an important avenue to explore and it requires further study. Finally, finding tighter bounds on the phase transition region for the controlled model could also be a subject of future work.

Acknowledgments

The authors thank Raphael Finkel for help correcting language issues. The work of the first two authors has been partially supported by the Italian Ministry for Economic Development (MISE) under the project "PIUCultura – Paradigmi Innovativi per l'Utilizzo della Cultura" (n. F/020016/01-02/X27), and under project "Smarter Solutions in the Big Data World (S2BDW)" (n. F/050389/01-03/X32) funded within the call "HORIZON2020" PON I&C 2014-2020. The work of the third author has been partially supported by the the NSF grant IIS-1707371.

References

[1] Answer set programming, AI Magazine 37(3) (2016) 5–80, (a series of articles dedicated to answer set programming, G. Brewka, T. Eiter, and M. Truszczynski, editors).

[2] D. Achlioptas, Random satisfiability, in: A. Biere, M. Heule, H. van Maaren, T. Walsh (eds.), Handbook of Satisfiability, vol. 185 of Frontiers in Artificial Intelligence and Applications, IOS Press, 2009, pp. 245–270.
URL http://dx.doi.org/10.3233/978-1-58603-929-5-245

[3] D. Achlioptas, C. Moore, The asymptotic order of the random k -sat threshold, in: 43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings, IEEE Computer Society, 2002, pp. 779–788.
URL http://dx.doi.org/10.1109/SFCS.2002.1182003

[4] M. Alviano, C. Dodaro, N. Leone, F. Ricca, Advances in WASP, in: F. Calimeri, G. Ianni, M. Truszczynski (eds.), Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings, vol. 9345 of Lecture Notes in Computer Science, Springer, 2015, pp. 40–54.
URL http://dx.doi.org/10.1007/978-3-319-23264-5_5

[5] G. Amendola, F. Ricca, M. Truszczynski, Generating hard random boolean formulas and disjunctive logic programs, in: IJCAI-17, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Melbourn, Australia, August, 2017, 2017, p. (to appear).

[6] C. Ansótegui, M. L. Bonet, J. Levy, Towards industrial-like random SAT instances, in: C. Boutilier (ed.), IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009, 2009, pp. 387–392.
URL http://ijcai.org/Proceedings/09/Papers/072.pdf

[7] C. Ansótegui, M. L. Bonet, J. Levy, F. Manyà, Measuring the hardness of SAT instances, in: D. Fox, C. P. Gomes (eds.), Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008, AAAI Press, 2008, pp. 222–228.
URL http://www.aaai.org/Library/AAAI/2008/aaai08-035.php

[8] G. Audemard, J. Lagniez, L. Simon, Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction, in: M. Järvisalo, A. V. Gelder (eds.), Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings, vol. 7962 of Lecture Notes in Computer Science, Springer, 2013, pp. 309–317.
URL http://dx.doi.org/10.1007/978-3-642-39071-5_23

[9] A. Biere, Lingeling essentials, A tutorial on design and implementation aspects of the the SAT solver lingeling, in: D. L. Berre (ed.), POS-14. Fifth Pragmatics of SAT workshop, a workshop of the SAT 2014 conference, part of FLoC 2014 during the Vienna Summer of Logic, July 13, 2014, Vienna, Austria, vol. 27 of EPiC Series in Computing, EasyChair, 2014, p. 88.
URL http://www.easychair.org/publications/?page=1039022100

[10] G. Brewka, T. Eiter, M. Truszczynski, Answer set programming at a glance, Commun. ACM 54 (12) (2011) 92–103.
URL http://doi.acm.org/10.1145/2043174.2043195

[11] R. Brummayer, F. Lonsing, A. Biere, Automated testing and debugging of SAT and QBF solvers, in: O. Strichman, S. Szeider (eds.), Theory and Applications of Satisfiability Testing - SAT 2010, 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings, vol. 6175 of Lecture Notes in Computer Science, Springer, 2010, pp. 44–57.
URL http://dx.doi.org/10.1007/978-3-642-14186-7_6

[12] F. Calimeri, M. Gebser, M. Maratea, F. Ricca, Design and results of the fifth answer set programming competition, Artif. Intell. 231 (2016) 151–181.
URL http://dx.doi.org/10.1016/j.artint.2015.09.008

[13] H. Chen, Y. Interian, A model for generating random quantified boolean formulas, in: L. P. Kaelbling, A. Saffiotti (eds.), IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005, Professional Book Center, 2005, pp. 66–71.
URL http://ijcai.org/Proceedings/05/Papers/0633.pdf

[14] C. Coarfa, D. D. Demopoulos, A. S. M. Aguirre, D. Subramanian, M. Y. Vardi, Random 3-sat: The plot thickens, in: R. Dechter (ed.), Principles and Practice of Constraint Programming - CP 2000, 6th International Conference, Singapore, September 18-21, 2000, Proceedings, vol. 1894 of Lecture Notes in Computer Science, Springer, 2000, pp. 143–159.
URL http://dx.doi.org/10.1007/3-540-45349-0_12

[15] J. M. Crawford, L. D. Auton, Experimental results on the crossover point in random 3-sat, Artif. Intell. 81 (1-2) (1996) 31–57.
URL http://dx.doi.org/10.1016/0004-3702(95)00046-1

[16] N. Creignou, U. Egly, M. Seidl, A framework for the specification of random SAT and QSAT formulas, in: A. D. Brucker, J. Julliand (eds.), Tests and Proofs - 6th International Conference, TAP 2012, Prague, Czech Republic, May 31 - June 1, 2012. Proceedings, vol. 7305 of Lecture Notes in Computer Science, Springer, 2012, pp. 163–168.
URL http://dx.doi.org/10.1007/978-3-642-30473-6_14

[17] G. Dequen, O. Dubois, An efficient approach to solving random *k*-satproblems, J. Autom. Reasoning 37 (4) (2006) 261–276.
URL http://dx.doi.org/10.1007/s10817-006-9025-2

[18] J. Díaz, L. M. Kirousis, D. Mitsche, X. Pérez-Giménez, On the satisfiability threshold of formulas with three literals per clause, Theor. Comput. Sci. 410 (30-32) (2009) 2920–2934.
URL http://dx.doi.org/10.1016/j.tcs.2009.02.020

[19] J. Ding, A. Sly, N. Sun, Proof of the satisfiability conjecture for large k, in: STOC, ACM, 2015, pp. 59–68.

[20] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, Ann. Math. Artif. Intell. 15 (3-4) (1995) 289–323.
URL http://dx.doi.org/10.1007/BF01536399

[21] M. Gebser, R. Kaminski, A. König, T. Schaub, Advances in *gringo* series 3, in: J. P. Delgrande, W. Faber (eds.), Logic Programming and Nonmonotonic Reasoning - 11th International Conference, LPNMR 2011, Vancouver, Canada, May 16-19, 2011. Proceedings, vol. 6645 of Lecture Notes in Computer Science, Springer, 2011, pp. 345–351.
URL http://dx.doi.org/10.1007/978-3-642-20895-9_39

[22] M. Gebser, B. Kaufmann, A. Neumann, T. Schaub, *clasp* : A conflict-driven answer set solver, in: C. Baral, G. Brewka, J. S. Schlipf (eds.), Logic Programming and Nonmonotonic Reasoning, 9th International Conference, LPNMR 2007, Tempe, AZ, USA, May 15-17, 2007, Proceedings, vol. 4483 of Lecture Notes in Computer Science, Springer, 2007, pp. 260–265.
URL http://dx.doi.org/10.1007/978-3-540-72200-7_23

[23] M. Gebser, B. Kaufmann, T. Schaub, Advanced conflict-driven disjunctive answer set solving, in: F. Rossi (ed.), IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013, IJCAI/AAAI, 2013, pp. 912–918.
URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6835

[24] M. Gelfond, Answer sets, in: F. van Harmelen, V. Lifschitz, B. W. Porter (eds.), Handbook of Knowledge Representation, vol. 3 of Foundations of Artificial Intelligence, Elsevier, 2008, pp. 285–316.
URL http://dx.doi.org/10.1016/S1574-6526(07)03007-6

[25] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Computing 9 (1991) 365–385.

[26] I. P. Gent, T. Walsh, Beyond NP: the QSAT phase transition, in: J. Hendler, D. Subramanian (eds.), Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence, July 18-22, 1999, Orlando, Florida, USA., AAAI Press / The MIT Press, 1999, pp. 648–653.
URL http://www.aaai.org/Library/AAAI/1999/aaai99-092.php

[27] J. Giráldez-Cru, J. Levy, Generating SAT instances with community structure, Artif. Intell. 238 (2016) 119–134.
URL http://dx.doi.org/10.1016/j.artint.2016.06.001

[28] M. Heule, M. Järvisalo, F. Lonsing, M. Seidl, A. Biere, Clause elimination for SAT and QSAT, J. Artif. Intell. Res. (JAIR) 53 (2015) 127–168.
URL http://dx.doi.org/10.1613/jair.4694

[29] M. Janota, W. Klieber, J. Marques-Silva, E. M. Clarke, Solving QBF with counterexample guided refinement, Artif. Intell. 234 (2016) 1–25.
URL http://dx.doi.org/10.1016/j.artint.2016.01.004

[30] M. Järvisalo, D. L. Berre, O. Roussel, L. Simon, The international SAT solver competitions, AI Magazine 33 (1).
URL http://www.aaai.org/ojs/index.php/aimagazine/article/view/2395

[31] A. C. Kaporis, L. M. Kirousis, E. G. Lalas, Selecting complementary pairs of literals, Electronic Notes in Discrete Mathematics 16 (2003) 47–70.
URL http://dx.doi.org/10.1016/S1571-0653(04)00462-7

[32] H. A. Kautz, B. Selman, Ten challenges redux: Recent progress in propositional reasoning and search, in: F. Rossi (ed.), Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings, vol. 2833 of Lecture Notes in Computer Science, Springer, 2003, pp. 1–18.
URL http://dx.doi.org/10.1007/978-3-540-45193-8_1

[33] W. Klieber, M. Janota, J. Marques-Silva, E. M. Clarke, Solving QBF with free variables, in: C. Schulte (ed.), Principles and Practice of Constraint Programming - 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings, vol. 8124 of Lecture Notes in Computer Science, Springer, 2013, pp. 415–431.
URL http://dx.doi.org/10.1007/978-3-642-40627-0_33

[34] C. Koch, N. Leone, G. Pfeifer, Enhancing disjunctive logic programming systems by SAT checkers, Artif. Intell. 151 (1-2) (2003) 177–212.
URL http://dx.doi.org/10.1016/S0004-3702(03)00078-X

[35] D. G. Mitchell, Resolution complexity of random constraints, in: P. V. Hentenryck (ed.), Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings, vol. 2470 of Lecture Notes in Computer Science, Springer, 2002, pp. 295–309.
URL http://link.springer.de/link/service/series/0558/bibs/2470/24700295.htm

[36] D. G. Mitchell, B. Selman, H. J. Levesque, Hard and easy distributions of SAT problems, in: W. R. Swartout (ed.), Proceedings of the 10th National Conference on Artificial Intelligence. San Jose, CA, July 12-16, 1992., AAAI Press / The MIT Press, 1992, pp. 459–465.
URL http://www.aaai.org/Library/AAAI/1992/aaai92-071.php

[37] G. Namasivayam, M. Truszczynski, Simple random logic programs, in: E. Erdem, F. Lin, T. Schaub (eds.), Logic Programming and Nonmonotonic Reasoning, 10th International Conference, LPNMR 2009, Potsdam, Germany, September 14-18, 2009. Proceedings, vol. 5753 of Lecture Notes in Computer Science, Springer, 2009, pp. 223–235.
URL http://dx.doi.org/10.1007/978-3-642-04238-6_20

[38] M. Narizzano, L. Pulina, A. Tacchella, Report of the third QBF solvers evaluation, JSAT 2 (1-4) (2006) 145–164.
URL http://jsat.ewi.tudelft.nl/content/volume2/JSAT2_6_Narizzano.pdf

[39] J. A. N. Pérez, A. Voronkov, Generation of hard non-clausal random satisfiability problems, in: M. M. Veloso, S. Kambhampati (eds.), Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA, AAAI Press / The MIT Press, 2005, pp. 436–442.
URL http://www.aaai.org/Library/AAAI/2005/aaai05-069.php

[40] G. Pfeifer, Improving the model generation/checking interplay to enhance the evaluation of disjunctive programs, in: V. Lifschitz, I. Niemelä (eds.), Logic Programming and Nonmonotonic Reasoning, 7th International Conference, LPNMR 2004, Fort Lauderdale, FL, USA, January 6-8, 2004, Proceedings, vol. 2923 of Lecture Notes in Computer Science, Springer, 2004, pp. 220–233.

[41] F. Pigorsch, C. Scholl, An aig-based qbf-solver using SAT for preprocessing, in: S. S. Sapatnekar (ed.), Proceedings of the 47th Design Automation Conference, DAC 2010, Anaheim, California, USA, July 13-18, 2010, ACM, 2010, pp. 170–175.
URL http://doi.acm.org/10.1145/1837274.1837318

[42] B. Selman, D. G. Mitchell, H. J. Levesque, Generating hard satisfiability problems, Artif. Intell. 81 (1-2) (1996) 17–29.
URL http://dx.doi.org/10.1016/0004-3702(95)00045-3

[43] G. Tseitin, On the complexity of derivation in propositional calculus, in: Automation of Reasoning, Symbolic Computation, Springer Berlin Heidelberg, 1983, pp. 466–483.

[44] K. Wang, L. Wen, K. Mu, Random logic programs: Linear model, TPLP 15 (6) (2015) 818–853.
URL http://dx.doi.org/10.1017/S1471068414000611

[45] L. Wen, K. Wang, Y. Shen, F. Lin, A model for phase transition of random answer-set programs, ACM Trans. Comput. Log. 17 (3) (2016) 22:1–22:34.
URL http://doi.acm.org/10.1145/2926791

[46] D. Wong, J. Schlipf, M. Truszczyński, On the distribution of programs with stable models, dagstuhl Seminar 05171: Nonmonotonic Reasoning, Answer Set Programming and Constraints (2005).

[47] Y. Zhao, F. Lin, Answer set programming phase transition: A study on randomly generated programs, in: C. Palamidessi (ed.), Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings, vol. 2916 of Lecture Notes in Computer Science, Springer, 2003, pp. 239–253.
URL http://dx.doi.org/10.1007/978-3-540-24599-5_17

Appendix A. Generalized Controlled Model

The controlled model $Q^{ctd}(k,A,E)$ introduced in Section 3.1, stipulates that every clause in the matrix of a QBF from the model contains exactly one universal variable, and that each universal variable occurs in exactly two clauses, in one of them as a positive literal (not negated) and in the other one as the negative literal (negated). It follows that $Q^{ctd}(k,A,E) \subseteq Q(1,k-1;A,E;2A)$, that is, the controlled model $Q^{ctd}(k,A,E)$ is a restriciton of the Chen-Interian model $Q(1,k-1;A,E;2A)$. Moreover, the key property of the controlled model $Q^{ctd}(k,A,E)$ is that, for every truth assignment to the universal variables in X, once we simplify the matrix accordingly we are left with *exactly* A $(k-1)$-literal clauses over E variables, whereas in the case of the Chen-Interian model $Q(1,k-1;A,E;2A)$, similar simplifications leave us with with varying number of $(k-1)$-clauses, with the *average* number being A.

We now generalize the model to allow clauses with exactly h occurrences of universal variables, where h is a fixed integer satidfying $1 \leq h \leq k$. More precisely, the model consists of QBFs $\forall X \exists Y\ F$, where F consists of $\binom{A}{h}2^h$ k-literal clauses, each clause consists of h literals over X and $k-h$ literals over Y (with no repetitions of variables), and where for every consistent set of h literals over X there is a single clause in the formula that contains them. A QBF in this model (to be precise, its matrix) is obtained by generating $\binom{A}{h}2^h$ h-literal clauses over X and extending each of them by a randomly generated consistent $(k-h)$-element set of literals over Y. We denote the set of QBFs obtained in this way by $Q^{gctd}(h,k-h,A,E)$ and call it the *generalized controlled model*.

Clearly, $Q^{ctd}(k,A,E) = Q^{gctd}(1,k-1,A,E)$. Thus, the controlled model we discussed in the paper is a special case of the model described here. We also note that $Q^{gctd}(h,k-h,A,E) \subseteq Q(h,k-h;A,E;\binom{A}{h}2^h)$. Thus, the generalized controlled model is a restriction of the appropriate Chen-Interian model — while random with respect to variables in Y (existentially quantified variables), the way variables in X (universally quantified variables) are treated is fully deterministic. In particular, for every truth assignment on X, once we simplify the matrix accordingly, we are left with *exactly* $\binom{A}{h}$ $(k-h)$-literal clauses over E variables in Y, while in the case of the Chen-Interian model $Q(h,k-h;A,E;\binom{A}{h}2^h)$, similar simplifications leave us with $(k-h)$-CNF formulas with varying number of clauses, with the *average* number being $\binom{A}{h}$.

Example 1. *Consider a set $X = \{x_1,x_2,x_3\}$ of $A = 3$ universal variables and a set $Y = \{y_1,y_2,y_3,y_4\}$ of $E = 4$ existential variables. We are interested in "generalized controlled formulas" having $k = 5$ literals with $h = 2$ of them over X. That is, we are interested in the model $Q^{gctd}(2,3,3,4)$. According to the definition, we have to build $\binom{3}{2}2^2 = 12$ clauses of length 5. For an appropriate enumeration C_i, $i = 1,\ldots,12$, these clauses will satisfy:*

$$\begin{array}{rclcrclcrcl}
\{x_1,x_2\} & \subseteq & C_1 & & \{x_1,x_3\} & \subseteq & C_5 & & \{x_2,x_3\} & \subseteq & C_9 \\
\{\neg x_1,x_2\} & \subseteq & C_2 & & \{\neg x_1,x_3\} & \subseteq & C_6 & & \{\neg x_2,x_3\} & \subseteq & C_{10} \\
\{x_1,\neg x_2\} & \subseteq & C_3 & & \{x_1,\neg x_3\} & \subseteq & C_7 & & \{x_2,\neg x_3\} & \subseteq & C_{11} \\
\{\neg x_1,\neg x_2\} & \subseteq & C_4 & & \{\neg x_1,\neg x_3\} & \subseteq & C_8 & & \{\neg x_2,\neg x_3\} & \subseteq & C_{12}
\end{array}$$

Let us choose any truth assignment σ on X, for instance, $\sigma(x_1) = x_1$, $\sigma(x_2) = \neg x_2$, and $\sigma(x_3) = \neg x_3$. Once we simplify the clauses with respect to this assignment, exactly $\binom{3}{2} = 3$ clauses $C_2 \setminus \{\neg x_1,x_2\}$, $C_6 \setminus \{\neg x_1,x_3\}$ and $C_9 \setminus \{x_2,x_3\}$ remain (all other clauses after the simplifications become tautologies and can be dropped).

Let $q^{gctd}(h,k-h,A,E)$ denote the probability that a random formula in $Q^{gctd}(h,k-h,A,E)$ is true. We define $\mu_l^{gctd}(h,k-h)$ to be the supremum over all positive real numbers ρ such that

$$\lim_{E\to\infty} q^{gctd}(h,k-h,\lfloor \rho E^{1/h} \rfloor,E) = 1,$$

and $\mu_u^{gctd}(h,k-h)$ to be the infimum over all positive real numbers ρ such that

$$\lim_{E\to\infty} q^{gctd}(h,k-h,\lfloor \rho E^{1/h} \rfloor,E) = 0.$$

We will now derive bounds on $\mu_l^{gctd}(h,k-h)$ and $\mu_u^{gctd}(h,k-h)$ by exploiting results on random $(k-h)$-CNF formulas. The proof is an adaptation of the proof of Theorem 2.

Theorem 8. *For every integers k and h such that $k \geq 2$ and $1 \leq h < k$, $\mu_l^{gctd}(h, k-h)$ and $\mu_u^{gctd}(h, k-h)$ are well defined.*

Proof. Let $\Phi \in Q^{gctd}(h, k-h, A, E)$, $X = \{x_1, ..., x_A\}$, and $Y = \{y_1, ..., y_E\}$. By definition, $\Phi = \forall X \exists Y F$, where $F = C_1 \wedge \ldots \wedge C_N$ is a k-CNF formula of $N = \binom{A}{h} 2^h$ clauses $C_i = l_{i,1} \vee \ldots \vee l_{i,k}$ such that $l_{i,1}, \ldots, l_{i,h}$ are literals over X and $l_{i,h+1}, \ldots, l_{i,k}$ are literals over Y. We define $C_i^Y = l_{i,h+1} \vee \ldots \vee l_{i,k}$ and $F^Y = C_1^Y \wedge \ldots \wedge C_N^Y$. Moreover, for every interpretation I of X we define $F|_I = \bigwedge \{C_i^Y \mid C_i \in F \text{ and } I \not\models l_{i,1} \vee \ldots \vee l_{i,h}\}$.

Let us assume that Φ is selected from $Q^{gctd}(h, k-h, A, E)$ uniformly at random. By the definition of the model $Q^{gctd}(h, k-h, A, E)$, F^Y can be regarded as selected from $C(k-h, N, E)$ uniformly at random and, for each truth assignment I of X, $F|_I$ can be regarded as selected uniformly at random from $C(k-h, M, E)$, where $M = \binom{A}{h}$.

To show that $\mu_u^{gctd}(h, k-h)$ are well defined, it is enough to show that there are r and s such that

$$\lim_{E \to \infty} q^{gctd}(h, k-h, \lfloor rE^{1/h} \rfloor, E) = 0 \quad \text{and} \quad \lim_{E \to \infty} q^{gctd}(h, k-h, \lfloor sE^{1/h} \rfloor, E) = 1.$$

The proof relies on an obvious property that for every fixed positive integer h, there are positive constants α_h and β_h such that for every sufficiently large positive integer A,

$$\beta_h A^h \geq \binom{A}{h} \geq \alpha_h A^h.$$

To prove the existence of r, let us fix any real r such that $\alpha_h (r/2)^h > \rho_u(k-h)$, and let $A = \lfloor rE^{1/h} \rfloor$. Next, let $\Phi = \forall X \exists Y F$ be a QBF selected uniformly at random from $Q^{gctd}(h, k-h, A, E)$ and I be a truth assignment on X. Clearly, if $F|_I$ is unsatisfiable, then Φ is false.

For all sufficiently large E, we have $A \geq (r/2) E^{1/h}$. Consequently, $A^h \geq (r/2)^h E$ and

$$\binom{A}{h} \geq \alpha_h (r/2)^h E.$$

Since $\alpha_h (r/2)^h > \rho_u(k-h)$, it follows that the probability that $F|_I$ is unsatisfiable converges to 1 as E approaches infinity. Thus, the probability that Φ is false converges to 0 as E approaches infinity. In other words,

$$\lim_{E \to \infty} q^{gctd}(h, k-h, \lfloor rE^{1/h} \rfloor, E) = 0.$$

To prove the existence of s we proceed similarly. Let s be any positive real such that $2^h \beta_h s^h \leq \rho_l(k-h)$ and let $A = \lfloor sE^{1/h} \rfloor$. Further, as before, let $\Phi = \forall X \exists Y F$ be a QBF selected uniformly at random from $Q^{gctd}(h, k-h, A, E)$.

Clearly, if the formula F^Y is satisfiable, then for every interpretation I of X, the formula $F|_I$ is satisfiable or, equivalently, Φ is true. In our case, we have that $A \leq sE^{1/h}$. Thus, $A^h \leq s^h E$. It follows that $2^h \binom{A}{h} \leq 2^h \beta_h s^h E$. Thus, $2^h \binom{A}{h} / E \leq 2^h \beta_h s^h < \rho_l(k-h)$. It follows that the probability that F^Y is satisfiable converges to 1 as E approaches infinity and so, also the probability that Φ is true converges to 1 as E approaches infinity. In other words,

$$\lim_{E \to \infty} q^{gctd}(h, k-h, \lfloor sE^{1/h} \rfloor, E) = 1.$$

□

Empirical Behavior.. We now discuss properties of the generalized controlled model presented above. In particular we compare the generalized controlled and the Chen-Interian models with respect to the hardness of formulas having the same number of variables, and comment on one possible extension of the generalized model.

We consider formulas from the Chen-Interian model $Q(a, e; A, E; m)$, where we set $a = 2$, $e = 3$, and $E = 32$, and vary the number A of universal variables over the range [2..80] and the number m of clauses over the range [10..1200]. As we did in Section 5.3, we compare the hardness properties of the generalized controlled and the Chen-Interian models with the same number of existential variables. These results are shown in Figure A.12. For the controlled model, for each value of A, the value on the corresponding

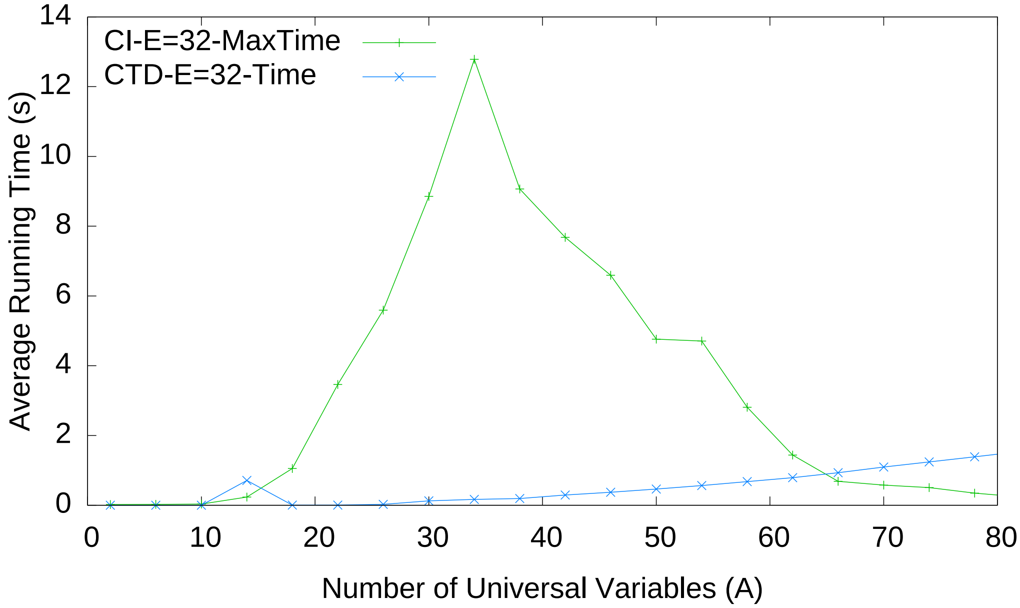


Figure A.12: Comparing Chen-Interian and Generalized Controlled model: hardness comparison.

hardness graph (the blue line) is obtained by averaging the solve times on formulas generated from the model $Q^{gctd}(2,3,A,32)$. The matrices of these formulas are 5-CNF formulas over $A+32$ variables and with $4\binom{A}{2}$ clauses. The corresponding point on the hardness graph for the Chen-Interian model is obtained by averaging the solve times on formulas generated from the model $Q(2,3;A,32;max)$, where for each A and $E=32$, max is selected to maximize the solve times (in particular, it falls in the phase transition region for the combination of the values A and $E=32$). The matrices of these formulas are 5-CNF formulas over $A+32$ variables and m clauses. The results show that the peak hardness regions for the two models are not aligned. Comparing this results with the one in Figure 4 we note that the generalized controlled model instances are much easier to solve than Chen-Interian ones, almost in every setting. The peak hardness from the generalized controlled model instances happens before the maximum hardness the peak hardness region for the Chen-Interian model. This is the opposite of what happens for (basic) controlled model instances as shown in Figure 4.

One possible weakness of the generalized controlled model is that the number of clauses, $m=4\binom{A}{2}$, grows quadratically with the number A of universal variables. Informally, this growth creates "long jumps" in terms of the number of clauses in a formula as we increment A and so, also the corresponding jumps in the ratio of the number of clauses to the number of existential variables. That may cause the model to miss the "sweet spot" of maximum hardness. For example, already in our experiment with $h=2$, formulas with $A=14$ feature 364 clauses, and formulas with $A=15$ feature 420 clauses. We established experimentally that formulas with $A=14$ are satisfied with the frequency 0.1, whereas the frequency of a satisfiable instance for $A=15$ is 1.

In order to verify whether the "jumps" contribute to the generation of easier formulas, we further extended the generalized controlled model to fill the gaps. Specifically, the *smooth* generalized controlled model, denoted by $Q^{sgctd}(h,k-h,E;m)$, where we specify the number of existential variables and the number of clauses in the matrix, and where the number of universal variables is determined by the constraint $2^h\binom{A-1}{h}+1\leq m\leq 2^h\binom{A}{h}$. In particular, if $m=2^h\binom{A}{h}$, $Q^{sgctd}(h,k-h,E;m)$ is defined to coincide with the generalized controlled model $Q^{gctd}(h,k-h,A,E)$. Formulas for m satisfying $2^h\binom{A-1}{h}+1\leq m<2^h\binom{A}{h}$ are obtained by generating an instance of $Q^{gctd}(h,k-h,A,E)$ and randomly choosing m among its clauses.

A phase transition result holds also for the smooth generalized controlled model. Let $q^{sgctd}(h,k-h,E;m)$ denote the probability that a random formula in $Q^{sgctd}(h,k-h,E;m)$ is true. We define $\mu_l^{sgctd}(h,k-h)$ to

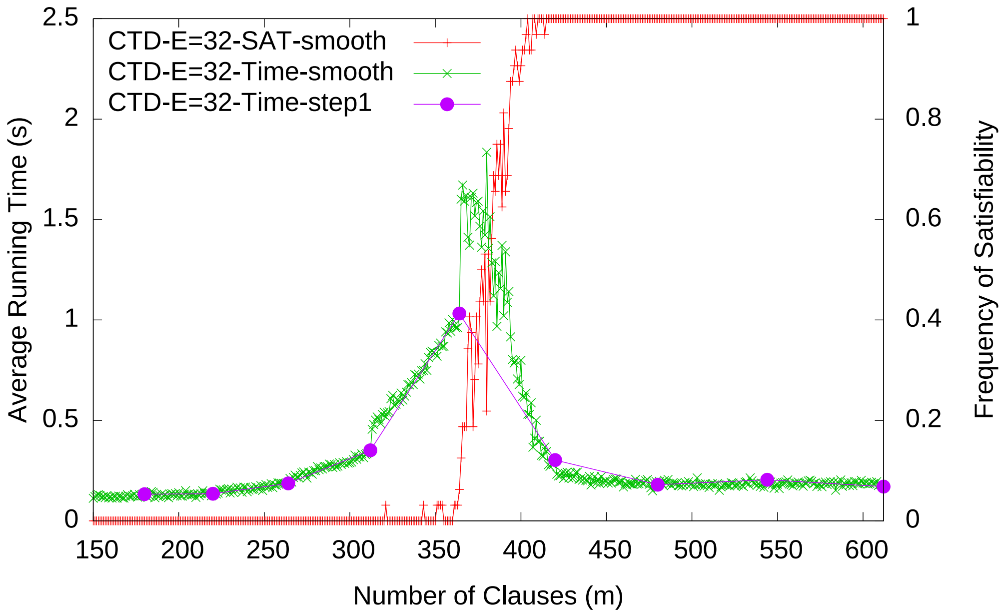


Figure A.13: Comparing Generalized Controlled model with Smooth Generalized Controlled model.

be the supremum over all positive real numbers ρ such that

$$\lim_{E\to\infty} q^{sgctd}(h,k-h,E,\lfloor \rho E \rfloor) = 1,$$

and $\mu_u^{sgctd}(h,k-h)$ to be the infimum over all positive real numbers ρ such that

$$\lim_{E\to\infty} q^{sgctd}(h,k-h,E,\lfloor \rho m \rfloor) = 0.$$

Theorem 8 implies the following result.

Corollary 3. *For every integers k and h such that $k \geq 2$ and $1 \leq h < k$, $\mu_l^{sgctd}(h,k-h)$ and $\mu_u^{sgctd}(h,k-h)$ are well defined.*

We experimented with the smooth generalized controlled model on the same setting as before but focusing on the phase transition region, that is, on values of A that are close to 14. The results reported in Figure A.13 were, thus, obtained varying A from 10 to 18 (so $150 \leq m \leq 612$). It can be noted that the smooth model allows us to generate formulas that are precisely in the phase transition zone, moreover we can obtain harder formulas. Nonetheless, the smooth generalized controlled model remains less hard than the Chen-Interian model, if we compare the hardest formulas that can be generated with the same number of existential variables, disregarding the number of universal variables. As we noted in the main part of the paper, alternative ways to compare the hardness of the models may exist and finding them is an important open research question.

Appendix B. Additional notes on the generation of formulas

Let X be a set consisting of N elements. We will consider the following method to generate random elements of X^t (the set of all t-tuples over X):

for each position i, $1 \leq i \leq t$, select an element from X uniformly at random.

Clearly, every element of X^t is equally likely to appear as the result of this method. Thus, the method generates t-tuples over X uniformly at random.

Let S be a property of t-tuples over X and let p_N be the probability that a t-tuple generated by the method described above has the property S. It follows that p_N is the probability that an t-tuple selected from X^t uniformly at random has the property S.

Next, let us define

$$D^t(X) = \{\langle x_1, \ldots, x_t \rangle \in X^t : x_i \neq x_j, \text{ for } i \neq j\}$$

In other words, $D^t(X)$ is the set of all tuples in X^t with no repeating elements.

Let S be a property of tuples in X^t. We will denote by p'_N the probability that a tuple selected from $D^t(X)$ uniformly at random has the property S. Then, if t is sufficiently smaller than N, p'_N can be closely estimated by p_N. To show that, let us define

$$R^t(X) = X^t \setminus D^t(X).$$

Clearly,

$$p_N = \frac{|S|}{|X^t|} \quad \text{and} \quad p'_N = \frac{|D^t(X) \cap S|}{|D^t(X)|}.$$

It follows that

$$p_N - \frac{|R^t(X) \cap S|}{|X^t|} \leq p'_N \leq p_N + \frac{|R^t(X) \setminus S|}{|X^t|}$$

and so,

$$p_N - \frac{|R^t(X)|}{|X^t|} \leq p'_N \leq p_N + \frac{|R^t(X)|}{|X^t|}$$

or, more explicitly,

$$p_N - \big(1 - \frac{(N-t+1)\cdots(N-1)N}{N^t}\big) \leq p'_N \leq p_N + \big(1 - \frac{(N-t+1)\cdots(N-1)N}{N^t}\big).$$

Lemma 1. *If* $\lim_{N\to\infty} t^2/N = 0$, *then*

$$\lim_{N\to\infty} \frac{(N-t+1)\cdots(N-1)N}{N^t} = 1$$

Proof. : Clearly,

$$\Big(\frac{N-t}{N}\Big)^t \leq \frac{(N-t+1)\cdots(N-1)N}{N^t} \leq 1.$$

Moeover,

$$\Big(\frac{N-t}{N}\Big)^t = \left[\Big(1 - \frac{1}{N/t}\Big)^{N/t}\right]^{t^2/N}$$

Since $\lim_{N\to\infty} t^2/N = 0$ and t is a positive integer, $\lim_{N\to\infty} N/t = \infty$. Thus,

$$\lim_{N\to\infty} \Big(1 - \frac{1}{N/t}\Big)^{N/t} = 1/e$$

and, consequently,

$$\lim_{N\to\infty} \left[\Big(1 - \frac{1}{N/t}\Big)^{N/t}\right]^{t^2/N} = 1.$$

□

Corollary 4. *If* $\lim_{N\to\infty} t^2/N = 0$, *then there is a sequence* ε_N *such that* $\lim_{N\to\infty} \varepsilon_N = 0$ *and*

$$p_N - \varepsilon_N \leq p'_N \leq p_N + \varepsilon_N.$$

Next, we observe that if the property S does not depend on the order of the elements in a tuple in $D^t(X)$, that is, the probability that a tuple in $D^t(X)$ has the property S is the same for every permutation of the elements in the tuple), then the probability that a set of t elements from X has a property S (its "set version" to be precise) is given by p'_N.

Our earlier discussion shows then that to estimate the probability that a t-element subset of X selected uniformly at random has a property S, it is sufficient to estimate the probability that a t-tuple over X (an element of X^t) selected uniformly at random has the property S.

In this paper, we take advantage of this observation in the case when X consists of formulas and S is the property that a set (tuple) of formulas is satisfiable (*SAT*), and unsatisfiable (UNSAT).

In particular, we consider in the paper the case when X is the set of all non-tautological k-literal clauses over the set of n propositional variables. We note that $|X| = 2^k\binom{n}{k}$. It follows that when studying the probability that a k-CNF formula with m clauses is satisfiable, where $m = O(n)$, the results above apply and the probabillity, in the limit, is the same no matter whether we vew formulas as sets or ordered tuples of clauses.

We also consider the case, when X is the set of all k-CNF formulas with m clauses over a set of n variables, that is, the set $C(k,n,m)$. Also here, it makes no difference whether a disjunction of such formulas is considered a set of those formulas or an ordered tuple of such formulas. Since we consider disjunctions of t CNF formulas, where t is fixed, the probability of such a disjunction being satisfiable is, in the limit, not affected by how we interpret the disjunction — as a set or an ordered tuple.