Hidden Structure in Unsatisfiable Random 3-SAT: an Empirical Study

Inês Lynce and João Marques-Silva
IST/INESC-ID, Technical University of Lisbon, Portugal
{ines,jpms}@sat.inesc-id.pt

Abstract

Recent advances in propositional satisfiability (SAT) include studying the hidden structure of unsatisfiable formulas, i.e. explaining why a given formula is unsatisfiable. Although theoretical work on the topic has been developed in the past, only recently two empirical successful approaches have been proposed: extracting unsatisfiable cores and identifying strong backdoors. An unsatisfiable core is a subset of clauses that defines a sub-formula that is also unsatisfiable, whereas a strong backdoor defines a subset of variables which assigned with all values allow concluding that the formula is unsatisfiable. The contribution of this paper is two-fold. First, we study the relation between the search complexity of unsatisfiable random 3-SAT formulas and the sizes of unsatisfiable cores and strong backdoors. For this purpose, we use an existing algorithm which uses an approximated approach for calculating these values. Second, we introduce a new algorithm that optimally reduces the size of unsatisfiable cores and strong backdoors, thus giving more accurate results. Experimental results indicate that the search complexity of unsatisfiable random 3-SAT formulas is related with the size of unsatisfiable cores and strong backdoors.

1. Introduction

The utilization of SAT in practical applications has motivated work on certifying SAT solvers (e.g. see [5]). Given a problem instance, the certifier needs to be able to verify that the computed truth assignment indeed satisfies a satisfiable instance and that, for an unsatisfiable instance, a proof of unsatisfiability can be generated. Certifying a SAT solver for a satisfiable instance is by far easier. Certifying a SAT solver for an unsatisfiable instance is hard. For an unsatisfiable instance, one has to be able to explain why the instance cannot be satisfied. For instance, one may provide a resolution proof based on an unsatisfiable core [1, 9] or a strong backdoor [8]. Broadly, an unsatisfiable core is a sub-formula that is still unsatisfiable and a strong backdoor is a set of variables which define a search subspace that suffices to prove unsatisfiability.

The main goal of this paper is to make an empirical study on hidden structure in typical case complexity. Theoretical work has already been developed in the past [3], but our focus is to make an *empirical* study. With this purpose, we studied random 3-SAT formulas, which are well-know for exhibiting a phase transition when the ratio of clauses to variables is compared with the search effort [2, 7]. The experimental results given in the paper aim to relate the size of unsatisfiable cores and strong backdoors with the search effort required to solve random 3-SAT unsatisfiable instances.

Empirical results have first been obtained using zChaff, which is the *only* available solver to integrate extraction of unsatisfiable cores [9]. However, this algorithm has the drawback of giving approximate results, meaning that there is no guarantee about the unsatisfiable core having the smallest number of clauses. Hence, we have developed a new model and algorithm that can be used to obtain the smallest size unsatisfiable cores and strong backdoors. Results for the new algorithm confirm the conclusions obtained by using zChaff.

This paper is organized as follows. In the next section we give the definitions, followed by a discussion on unsatisfiable cores and strong backdoors. In Section 5 we give experimental data for running zChaff on random 3-SAT instances. Section 6 relates hardness with hidden structure. Afterwards, we introduce a new model and algorithm for computing smallest size unsatisfiable cores and strong backdoors. Finally, the paper concludes by suggesting future research work.

2. Definitions

The standard SAT definitions of clauses, variables and literals are assumed. A CNF formula φ is a con-

junction of clauses, a clause ω is a disjunction of literals, and a literal l or $\neg l$ is either a variable or its complement. The set of m clauses is denoted by Ω and the set of n variables is denoted by X. For the a formula φ and for each clause ω we can also use set notation. Hence, $\omega \in \varphi$ means that clause ω is a clause of formula φ, and $l \in \omega$ means that l is a literal of clause ω.

A clause is said to be *satisfied* if at least one of its literals assumes value 1, *unsatisfied* if all of its literals assume value 0, *unit* if all but one literal assume value 0, and *unresolved* otherwise. A formula is said to be *satisfied* if all its clauses are satisfied, and is *unsatisfied* if at least one clause is unsatisfied.

A *truth assignment* $A_{X'} : X' \subseteq X \rightarrow \{true, false\}$ for a formula φ is a subset of assigned variables and their corresponding binary values. An assignment $A_{X'}$ is *complete iff* $|X'| = n$; otherwise it is *partial*. Moreover, $\varphi[A_{X'}]$ denotes formula φ after setting the partial truth assignment $A_{X'}$. The SAT problem consists in deciding whether there exists a truth assignment to the variables such that the formula becomes satisfied.

Random 3-SAT instances are obtained by randomly generating clauses with length 3. For an instance with n variables and m clauses, each literal of the m clauses is randomly selected from the $2n$ possible literals such that each literal is selected with the same probability of $1/2n$. Clauses with repeated literals or with a literal and its negation (tautological clauses) are discarded.

Random k-SAT formulas are particularly interesting due to the occurrence of a phase-transition or threshold phenomenon, i.e. a rapid change in complexity when increasing (or decreasing) the ratio of clauses to variables [2, 7]. For a small ratio almost all formulas are under-constrained and therefore satisfiable. As the value of m/n increases, almost all instances are over-constrained and therefore unsatisfiable. Experiments strongly suggest that for random 3-SAT there is a threshold at some critical ratio of clauses to variables $m/n \approx 4.3$ such that beyond this value the probability of generating a satisfiable instance drops to almost zero.

3. Unsatisfiable Cores

Research in unsatisfiable cores can be distinguished between theoretical and experimental work. In the theoretical field, unsatisfiable cores complexity has been analyzed and formal algorithms have been proposed [3, 4]. Experimental work includes contributions of Bruni and Sassano [1] and Zhang and Malik [9]. Both approaches extract unsatisfiable cores. The first approach proposes an adaptive search guided by clauses hardness. The second approach is motivated by considering that a CNF formula is unsatisfiable *iff* it is possible to generate an empty clause by resolution from the original clauses. In this case, the resolution steps are emulated by the creation of nogoods. The unsatisfiable core is given by the set of original clauses involved in the derivation of the empty clause.

Definition 1 (Unsatisfiable Core) *Given a formula φ, UC is an unsatisfiable core for φ iff UC is a formula s.t. UC is unsatisfiable and $UC \subseteq \varphi$.*

Observe that an unsatisfiable core can be defined as any subset of the original formula that is unsatisfiable. Consequently, there may exist many different unsatisfiable cores, with different number of clauses, for the same problem instance, such that some of these cores can be subsets of others. Also, and in the worst case, the unsatisfiable core corresponds exactly to the set of original clauses.

Definition 2 (Minimal Unsatisfiable Core) *An unsatisfiable core UC for φ is a minimal unsatisfiable core iff removing any clause $\omega \in UC$ from UC implies that $UC - \{\omega\}$ is satisfiable.*

Definition 3 (Minimum Unsatisfiable Core) *An unsatisfiable core UC for φ is a minimum unsatisfiable core iff it is a minimal unsatisfiable core of minimum cardinality.*

Interestingly, the existing experimental work described above [1, 9] has very little concern regarding extraction of *minimal* unsatisfiable cores. Nonetheless, the work in [9] proposes an iterative solution for *reducing* an unsatisfiable core, by iteratively invoking the SAT solver on each computed sub-formula. This solution, albeit capable of reducing the size of computed unsatisfiable cores, does not provide *any* guarantees regarding the unsatisfiable core being either minimal or minimum. However, in some practical applications it may be useful identifying the *minimum* unsatisfiable core of a given problem instance, i.e. the *smallest* number of clauses that make the instance unsatisfiable.

4. Strong Backdoors

A backdoor is a special subset of variables that characterizes hidden structure in problem instances [8]. Backdoor definition depends on a sort of algorithm called *sub-solver*. A sub-solver $\mathcal{S}$ always runs in polynomial time. For example, $\mathcal{S}$ could be a solver that is able to solve 2-SAT instances but rejects K-SAT instances, with $K \geq 3$. Given a partial truth assignment $A_{X'} : X' \subseteq X \rightarrow \{true, false\}$, a sub-solver $\mathcal{S}$ is able to solve the formula $\varphi[A_{X'}]$ in polynomial time.

Definition 4 (Backdoor) *A nonempty subset Y of the variables set X is a backdoor for φ w.r.t. $\mathcal{S}$ if for some partial truth assignment $A_Y : Y \rightarrow \{true, false\}$, $\mathcal{S}$ returns a satisfying assignment of $\varphi[A_Y]$.*

Clearly, the definition of backdoor given above only applies to *satisfiable* formulas. Moreover, observe that there may exist many backdoor sets for a given formula. (In the worst case, there is only one backdoor that corresponds exactly to the set of all variables.)

Definition 5 (Minimal Backdoor) *A nonempty backdoor set Y for φ w.r.t. $\mathcal{S}$ is minimal iff removing any variable $v \in Y$ from Y implies that $Y - \{v\}$ is not a backdoor set.*

Definition 6 (Minimum Backdoor) *A nonempty backdoor set Y for φ w.r.t. $\mathcal{S}$ is minimum iff it is a minimal backdoor of minimum cardinality.*

Since the definition of backdoor given above only considers satisfiable instances, Williams *et al.* [8] introduced the definition of strong backdoor for unsatisfiable instances. This definition holds for both satisfiable and unsatisfiable instances.

Definition 7 (Strong Backdoor) *A nonempty subset Y of the variables set X is a strong backdoor for φ* w.r.t. *$\mathcal{S}$ if for all $A_Y : Y \rightarrow \{true, false\}$, $\mathcal{S}$ returns a satisfying assignment for $\varphi[A_Y]$ or concludes unsatisfiability of $\varphi[A_Y]$.*

The definition of strong backdoor contrasts with the definition of backdoor to the extent that for a strong backdoor Y no truth assignment is specified. This means that all possible assignments of Y have to be considered. Observe that minimum and minimal strong backdoors can be defined similarly to minimum and minimal backdoors.

5. Random 3-SAT and zChaff

In this section we analyze zChaff's results on random 3-SAT instances. We used zChaff [6] for being an efficient DLL-based SAT solver integrating the extraction of unsatisfiable cores. (Clearly, solving different sub-formulas with *any* complete solver would also allow us to extract unsatisfiable cores, although not so efficiently.)

Besides zChaff being enhanced with clause recording, its behavior on solving 3-SAT instances is overall similar to the behavior reported in the literature for a basic DLL solver (e.g. see [7]). This is explained by clause learning being very useful for structured instances that usually come from real-world domains, rather than for random instances.

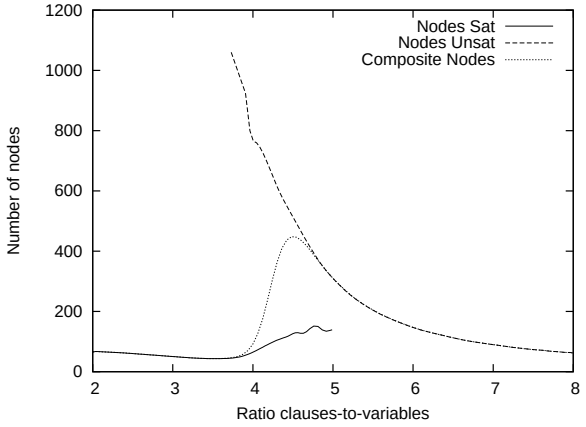

Figure 1. zChaff on solving random 3-SAT formulas with 100 variables.

Figure 1 gives the number of nodes when using zChaff for solving satisfiable and unsatisfiable random 3-SAT formulas with 100 variables as a function of the ratio of clauses to variables. Observe that similar results have been obtained in the past with a DLL solver [2, 7]. Moreover, the graph would exhibit a similar shape independently of the number of variables, although as the number of variables increases the steeper are the curves. Overall, the maximum value for the number of nodes is observed when the ratio of clauses/variables is ≈ 4.3.

The main conclusion is essentially that satisfiable and unsatisfiable sets are quite different when comparing the number of nodes. Most satisfiable instances are very easy to solve. Satisfiable instances with a higher ratio clauses/variables are slightly more difficult to solve. Unsatisfiable instances with a small ratio clauses/variables are the most difficult. Also, unsatisfiable instances with a larger ratio are still hard.

6. Hardness and Hidden Structure

Early studies on complexity relate hardness of k-SAT instances with the ratio of the number of clauses to the number of variables [2, 7]. Theoretical work has already related hardness and hidden structure [3]. However, little effort has ever been made in order to empirically relate these two aspects. Interestingly, recent empirical work on unsatisfiable cores and strong backdoors has brought some new insights on the topic.

Our first intuition was that hardness and the size of unsatisfiable cores and strong backdoors would be related due to the following reasons:

- Unsatisfiability is proved when the search space

is exhausted. For a DLL solver with an *accurate* heuristic the search space can be reduced to 2^b, where b is the size of the minimum strong backdoor. Also, for a solver with clause recording, the smallest is the size of the unsatisfiable core, the smallest is the number of steps required to derive the empty clause. (Although a recorded clause may include more than one resolution step.)

- The probability of generating satisfiable instances exhibits a phase-transition (see Figure 1), i.e. at a certain value of the ratio of clauses to variables the probability of generating an satisfiable clause *quickly* decreases to 0% as we add clauses to the formula. Conversely, the probability of generating unsatisfiable instances *quickly* increases to 100% at a certain value of the ratio of clauses to variables. Hence, unsatisfiable instances with a ratio of clauses to variables m/n above ≈ 4.3 are probably unsatisfiable with less than m clauses.

  For example, let us consider the generation of a *typical* unsatisfiable formula φ with n variables and m clauses, where $m/n > 4.3$. Consider that formula φ has a set of clauses $\Omega = \{\omega_1, ..., \omega_p, ..., \omega_m\}$. Suppose that φ is built by adding clauses in Ω one at a time. Moreover, with clauses $\{\omega_1..\omega_{p-1}\}$ ($p \approx m - 4.3n$) the formula is satisfiable but with all the clauses $\{\omega_1..\omega_p\}$ the formula is unsatisfiable. Thus the minimum unsatisfiable core size is $\leq p$. Furthermore, adding clauses $\{\omega_{p+1}, ..., \omega_m\}$ to the formula may only reduce the size of the minimum unsatisfiable core.

Clearly, the same reasoning can be applied to strong backdoors. This allows us to conclude that unsatisfiable cores and strong backdoors sizes are related, to the extent that both sizes decrease with the increasing of the ratio of the number of clauses to variables.

Figure 2 shows the evolution on the size of unsatisfiable cores and strong backdoors. More precisely, results indicate the percentage of clauses in the unsatisfiable cores with respect to the total number of clauses and the percentage of variables in the strong backdoors with respect to the total number of variables. Results are given for random unsatisfiable 3-SAT formulas with 50, 100 and 150 variables, as a function of the ratio of the number of clauses to variables.

The size of unsatisfiable cores has been computed by zChaff. The size of strong backdoors has been obtained from the corresponding unsatisfiable cores: for each instance, all variables in the clauses of the unsatisfiable core have been considered to be part of the strong backdoor. This means that each strong backdoor Y for a formula φ has been defined w.r.t. a sub-solver

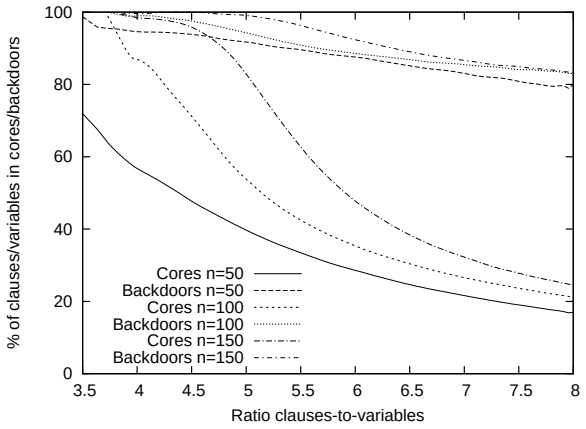

Figure 2. Unsatisfiable cores and strong backdoors for unsatisfiable random 3-SAT formulas.

S that for all assignments $A_Y : Y \rightarrow \{true, false\}$ simply checks that at least one clause is unsatisfied and finally concludes unsatisfiability of $\varphi[A_Y]$.

With respect to the size of unsatisfiable cores, results in Figure 2 clearly confirm our intuition. Observe that the reduction in the size of unsatisfiable cores is not only due to the increasing number of clauses with the increasing ratio of clauses to variables. Indeed, the absolute value for the size of unsatisfiable cores also decreases as a function of the ratio clauses/variables. Hence, one may conclude that harder instances have unsatisfiable cores much larger than easier instances with a higher ratio of clauses to variables. In addition, the relation between hardness and strong backdoors size is also suggested, although not so clearly. One may argue that the sub-solver S involved in the extraction of the strong backdoor does not favor getting a small strong backdoor. (Using different sub-solvers is future research work.)

7. Accurating Results

The previous plot exhibits a clear trend towards relating hardness with the size of unsatisfiable cores and strong backdoors. However, one may strengthen the obtained conclusions with more accurate results. In this section, we provide a model for identifying minimum and minimal unsatisfiable cores and strong backdoors.

Clearly, a brute-force algorithm can be used for exploring the whole search space while keeping track of the minimum unsatisfiable core. But we can do significantly better: we can emulate hiding each of the clauses in order to perform the search in all possible subsets of clauses. Also, we can learn from the conflicts.

We assume that each formula φ is defined over n variables, $X = \{x_1, \ldots, x_n\}$, and has m clauses, $\Omega = \{\omega_1, \ldots, \omega_m\}$. We start by defining a set S of m new variables, $S = \{s_1, \ldots, s_m\}$, and create a new formula φ' defined on $n + m$ variables, $X \cup S$, and with m clauses, $\Omega = \{\omega'_1, ..., \omega'_m\}$. Each clause $\omega'_i \in \varphi'$ is defined from a corresponding clause $\omega_i \in \varphi$ and from a variable s_i s.t. $\omega'_i = \{\neg s_i\} \cup \omega_i$.

Example 7.1 *Consider formula φ having variables $X = \{x_1, x_2, x_3\}$ and clauses $\Omega = \{\omega_1, ..., \omega_6\}$:*

$\omega_1 = x_1 \vee \neg x_3$ $\quad$ $\omega_4 = \neg x_2 \vee \neg x_3$
$\omega_2 = x_2$ $\quad$ $\omega_5 = x_2 \vee x_3$
$\omega_3 = \neg x_2 \vee x_3$ $\quad$ $\omega_6 = \neg x_1 \vee x_2 \vee \neg x_3$

Given the CNF formula φ given above, the new formula φ' is defined on variables $X \cup S = \{x_1, x_2, x_3, s_1, ..., s_6\}$ and clauses $\Omega' = \{\omega'_1, ..., \omega'_6\}$, such that:

$\omega'_1 = \neg s_1 \vee x_1 \vee \neg x_3$ $\quad$ $\omega'_4 = \neg s_4 \vee \neg x_2 \vee \neg x_3$
$\omega'_2 = \neg s_2 \vee x_2$ $\quad$ $\omega'_5 = \neg s_5 \vee x_2 \vee x_3$
$\omega'_3 = \neg s_3 \vee \neg x_2 \vee x_3$ $\quad$ $\omega'_6 = \neg s_6 \vee \neg x_1 \vee x_2 \vee \neg x_3$

Observe that S variables can be interpreted as *clause selectors* which allow considering or not each clause ω_i. For example, assigning $s_2 = 0$ makes clause ω'_2 satisfied and therefore variable x_2 does not have to be assigned value 1, as it was for the original clause $\omega_2 = x_2$. Moreover, φ' is readily satisfiable by setting all s_i variables to 0.

Let us now consider a backtrack search SAT solver where decisions are first made on the S variables (defining the S space) and afterwards on the X variables (defining the X space); hence, each assignment to the S variables defines a potential core. Now, for each assignment to the S variables, the resulting sub-formula may be satisfiable or unsatisfiable.

An unsatisfiable core is computed whenever the search backtracks from the X space to the S space, meaning that there is no solution to the formula given the current S assignments, i.e. the original formula φ was proved to be unsatisfiable. For each unsatisfiable sub-formula, the number of S variables assigned value 1 indicates how many clauses are contained in the unsatisfiable core. The *minimum unsatisfiable core* is obtained from the unsatisfiable sub-formula with the *least* number of S variables assigned value 1. Moreover, for SAT solvers with clause recording, a clause is recorded after each conflict that allows backtracking from the X space to the S space. Hence, an unsatisfiable core can be easily obtained from the new recorded clause. Observe that this unsatisfiable core is restricted to the clauses involved in the derivation of the empty clause.

Example 7.2 *Given formula φ' from Example 7.1, recording clause $\omega'_7 = \neg s_2 \vee \neg s_3 \vee \neg s_4$ means that the unsatisfiable core $\{\omega_2, \omega_3, \omega_4\}$ has been identified.*

The key challenge of the proposed model is the search space. For the original problem instance the search space is 2^n, where n is the number of variables, whereas for the transformed problem instance the search space becomes 2^{n+m}, where m is the number of clauses. Nevertheless, a few key optimizations can be applied. First, the SAT-based algorithm can start with an upper bound on the size of the minimum unsatisfiable core. For this purpose, the algorithm proposed in [9] can be used. Hence, when searching for the minimum unsatisfiable core, we just need to consider assignments to the S variables which yield smaller unsatisfiable cores. This additional constraint can be modeled as a cardinality constraint. Furthermore, each computed unsatisfiable core can be used for backtracking *non-chronologically* on the S variables, thus potentially reducing the search space.

Besides the *traditional* clause recording scheme, where each new clause corresponds to a sequence of resolution steps, a new clause is recorded whenever *a solution is found*. The new clause contains all the S literals assigned value 0 (thus satisfying the corresponding clause), *except* for those clauses that would also be satisfied by the X variables in the computed solution.

Example 7.3 *Consider again formula φ' from Example 7.1, and suppose that the current set of assignments is $\{s_1$=0, s_2=0, s_3=1, s_4=1, s_5=0, s_6=1, x_1=1, x_2=0, x_3=0$\}$. At this stage of the search, all clauses are satisfied, and therefore a solution is found. Consequently, a new clause is recorded to avoid finding again the same solution and also to force finding an unsatisfiable core in the future.*

Although S literals assigned value 0 are s_1, s_2 and s_5, clause ω'_1 is also satisfied by assigning $x_1 = 1$. Hence, the new recorded clause is $\omega'_8 = s_2 \vee s_5$. The new clause means that for finding an unsatisfiable core either clause ω_2 or clause ω_5 has to be part of the formula.

Finally, observe that *minimal* unsatisfiable cores can also be obtained by this algorithm as long as the solver is given any unsatisfiable sub-formula instead of the whole formula.

A similar algorithm can be used to obtain a minimum strong backdoor. Again, the idea is to extract a strong backdoor from the corresponding unsatisfiable core. Besides having additional variables for selecting clauses, we also need a set T of new variables to be used as selectors for variables in the original formula. (Satisfying variable $t_i \in T$ implies variable x_i being part of a strong backdoor.) For each variable x_i a new constraint is added,

$$t_i \leftrightarrow \bigvee_{s \in S_i} s$$

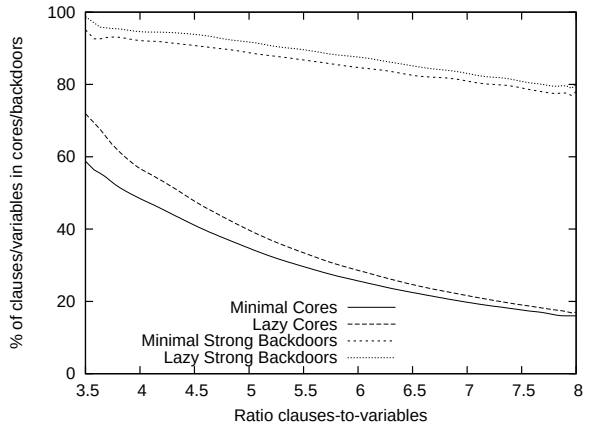

Figure 3. Minimal unsatisfiable cores and strong backdoors for unsatisfiable random 3-SAT.

where S_i is the subset of S variables occurring in clauses with x_i or $\neg x_i$. The *minimum strong backdoor* is obtained from the unsatisfiable sub-formula with the *least* number of T variables assigned value 1. With these additional constraints, we guarantee that a variable x_i is part of a strong backdoor *iff* a clause with x_i or $\neg x_i$ is part of a given unsatisfiable core.

Example 7.4 *Given formula φ' from Example 7.1, the CNF clauses to be added w.r.t. variable x_1 would be the following: $\neg t_1 \vee s_1 \vee s_6$, $\neg s_1 \vee t_1$ and $\neg s_6 \vee t_1$.*

The proposed algorithm is able to identify minimum or minimal strong backdoors, depending on the input being either the original formula or an unsatisfiable sub-formula. A key optimization consists in using the size of the smallest strong backdoor extracted so far as a cardinality constraint.

Figure 3 gives the size of *minimal* unsatisfiable cores and strong backdoors as a percentage of clauses and variables in the formula, respectively. Due to the complexity of this optimization problem, the data is restricted to the minimal (and not *minimum*) unsatisfiable cores and strong backdoors for random 3-SAT formulas with only 50 variables. However, it is predictable that the same figures would be obtained for minimum values and for instances with more variables.

Interesting conclusions may be drawn from Figure 3. First of all, it is clear that the values obtained by a lazy approach do not correspond to minimal values. Second, it is possible to relate the values for the lazy approach with the minimal values by an almost constant gap. Finally, this plot confirms that hardness can be related with hidden structure, i.e. hard unsatisfiable random 3-SAT formulas exhibit larger unsatisfiable cores and strong backdoors. Again, the relation between hardness and strong backdoors is still not being as clear as the relation between hardness and unsatisfiable cores, although we believe that this is due to the sub-solver used being far from giving small backdoors.

8. Conclusions and Future Work

In this paper we studied the relation between hardness of unsatisfiable random 3-SAT formulas and the sizes of unsatisfiable cores and strong backdoors. Besides using an existing algorithm, we introduced an algorithm that is able to identify minimal or minimum unsatisfiable cores and strong backdoors. Experimental results indicate that hard unsatisfiable instances exhibit larger unsatisfiable cores and strong backdoors. Future research work should definitely consider other sub-solvers for identifying strong backdoors. In addition, the experimental study should be extended to structured real-world instances.

References

[1] R. Bruni and A. Sassano. Restoring satisfiability or maintaining unsatisfiability by finding small unsatisfiable subformulae. In *LICS Workshop on Theory and Applications of Satisfiability Testing*, June 2001.

[2] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 331–337, 1991.

[3] V. Chvtal and E. Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[4] H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theoretical Computer Science*, 289(1):503–516, 2002.

[5] K. L. McMillan. Interpolation and SAT-based model checking. In *Proceedings of Computer Aided Verification*, 2003.

[6] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Engineering an efficient SAT solver. In *Proceedings of the Design Automation Conference*, pages 530–535, June 2001.

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

[8] R. Williams, C. P. Gomes, and B. Selman. Backdoors to typical case complexity. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2003.

[9] L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Proceedings of the Design and Test in Europe Conference*, pages 10880–10885, March 2003.