Backdoors in the Context of Learning

Bistra Dilkina Carla P. Gomes Ashish Sabharwal

Department of Computer Science
Cornell University, Ithaca NY 14853-7501, U.S.A.
{bistra,gomes,sabhar}@cs.cornell.edu

Abstract. The concept of backdoor variables has been introduced as a structural property of combinatorial problems that provides insight into the surprising ability of modern satisfiability (SAT) solvers to tackle extremely large instances. This concept is, however, oblivious to "learning" during search—a key feature of successful combinatorial reasoning engines for SAT, mixed integer programming (MIP), etc. We extend the notion of backdoors to the context of learning during search. We prove that the smallest backdoors for SAT that take into account clause learning and order-sensitivity of branching can be exponentially smaller than "traditional" backdoors. We also study the effect of learning empirically.

1 Introduction

In recent years we have seen tremendous progress in the state of the art of SAT solvers: we can now efficiently solve large real-world problems. A fruitful line of research in understanding and explaining this outstanding success focuses on the role of *hidden structure* in combinatorial problems. One example of such hidden structure is a backdoor set, i.e., a set of variables such that once they are instantiated, the remaining problem *simplifies* to a tractable class [6, 7, 8, 12, 15, 16]. Backdoor sets are defined with respect to efficient sub-algorithms, called *sub-solvers*, employed within the systematic search framework of SAT solvers. In particular, the definition of strong backdoor set B captures the fact that a systematic tree search procedure (such as DPLL) restricted to branching only on variables in B will successfully solve the problem, whether satisfiable or unsatisfiable. Furthermore, in this case, the tree search procedure restricted to B will succeed independently of the order in which it explores the search tree.

Most state-of-the-art SAT solvers rely heavily on clause learning which adds new clauses every time a conflict is derived during search. Adding new information as the search progresses has not been considered in the traditional concept of backdoors. In this work we extend the concept of backdoors to the context of learning, where information learned from previous search branches is allowed to be used by the sub-solver underlying the backdoor. This often leads to much smaller backdoors than the "traditional" ones. In particular, we prove that the smallest backdoors for SAT that take into account clause learning can be exponentially smaller than traditional backdoors oblivious to these solver features. We also present empirical results showing that the added power of learning-sensitive backdoors is also often observed in practice.

2 Preliminaries

For lack of space, we will assume familiarity with Boolean formulas in conjunctive normal form (CNF), the satisfiability testing problem (SAT), and DPLL-based backtrack search methods for SAT. *Backdoor sets* for such formulas and solvers are defined with respect to efficient sub-algorithms, called *sub-solvers*, employed within the systematic search framework of SAT solvers. In practice, these sub-solvers often take the form of efficient procedures such as unit propagation (UP), pure literal elimination, and failed-literal probing. In some theoretical studies, solution methods for structural sub-classes of SAT such as 2-SAT, Horn-SAT, and RenamableHorn-SAT have also been considered as sub-solvers. Formally [16], a *sub-solver A for SAT* is any polynomial time algorithm satisfying certain natural properties on every input CNF formula F: (1) Trichotomy: A either determines F correctly (as satisfiable or unsatisfiable) or fails; (2) A determines F for sure if F has no clauses or contains the empty clause; and (3) if A determines F, then A also determines $F|_{x=0}$ and $F|_{x=1}$ for any variable x.

For a formula F and a truth assignment τ to a subset of the variables of F, we will use $F|_\tau$ to denote the simplified formula obtained after applying the (partial) truth assignment to the affected variables.

Definition 1 (Weak and Strong Backdoors for SAT [16]). *Given a CNF formula F on variables X, a subset of variables $B \subseteq X$ is a* weak backdoor *for F w.r.t. a sub-solver A if for* some *truth assignment $\tau : B \to \{0,1\}$, A returns a satisfying assignment for $F|_\tau$. Such a subset B is a* strong backdoor *if for* every *truth assignment $\tau : B \to \{0,1\}$, A returns a satisfying assignment for $F|_\tau$ or concludes that $F|_\tau$ is unsatisfiable.*

Weak backdoor sets capture the fact that a well-designed heuristic can get "lucky" and find the solution to a hard satisfiable instance if the heuristic guidance is correct even on the small fraction of variables that constitute the backdoor set. Similarly, strong backdoor sets B capture the fact that a systematic tree search procedure (such as DPLL) restricted to branching only on variables in B will successfully solve the problem, whether satisfiable or unsatisfiable. Furthermore, in this case, the tree search procedure restricted to B will succeed independently of the order in which it explores the search tree.

3 Backdoor Sets for Clause Learning SAT Solvers

The last point made in Section 2—that the systematic search procedure will succeed independent of the order in which it explores various truth valuations of variables in a backdoor set B—is, in fact, a very important notion that has only recently begun to be investigated, in the context of mixed-integer programming [1]. In practice, many modern SAT solvers employ *clause learning* techniques, which allow them to carry over information from previously explored branches to newly considered branches. Prior work on proof methods based on clause learning and the resolution proof system suggests that, especially for

unsatisfiable formulas, some variable-value assignment orders may lead to significantly shorter search proofs than others. In other words, it is very possible that "learning-sensitive" backdoors are much smaller than "traditional" strong backdoors. To make this notion of incorporating learning-during-search into backdoor sets more precise, we introduce the following extended definition:

Definition 2 (Learning-Sensitive Backdoors for SAT). *Given a CNF formula F on variables X, a subset of variables $B \subseteq X$ is a* learning-sensitive backdoor *for F w.r.t. a sub-solver A if there exists a search tree exploration order such that a clause learning SAT solver branching only on the variables in B, with this order and with A as the sub-solver at the leaves of the search tree, either finds a satisfying assignment for F or proves that F is unsatisfiable.*

Note that, as before, each leaf of this search tree corresponds to a truth assignment $\tau : B \to \{0, 1\}$ and induces a simplified formula $F|_\tau$ to be solved by A. However, the tree search is naturally allowed to carry over and use learned information from previous branches in order to help A determine $F|_\tau$. Thus, while $F|_\tau$ may not always be solvable by A *per se*, additional information gathered from previously explored branches may help A solve $F|_\tau$. We note that incorporating learned information can, in principle, also be considered for the related notion of *backdoor trees* [14], which looks at the smallest search tree size rather than the set of branching variables.

We explain the power of learning-sensitivity through the following example formula, for which there is a natural learning-sensitive backdoor of size one w.r.t. unit propagation but the smallest traditional strong backdoor is of size 2. We will then generalize this observation into an exponential separation between the power of learning-sensitive and traditional strong backdoors for SAT.

Example 1. Consider the unsatisfiable SAT instance, F_1:

$$\begin{aligned}(x \vee p_1), (x \vee p_2), (\neg p_1 \vee \neg p_2 \vee q), \quad & (\neg q \vee a), (\neg q \vee \neg a \vee b), (\neg q \vee \neg a \vee \neg b) \\ (\neg x \vee q \vee r), \quad & (\neg r \vee a), (\neg r \vee \neg a \vee b), (\neg r \vee \neg a \vee \neg b)\end{aligned}$$

We claim that $\{x\}$ is a learning-sensitive backdoor for F_1 w.r.t. the unit propagation sub-solver, while all traditional strong backdoors are of size at least two. First, let's understand why $\{x\}$ does work as a backdoor set when clause learning is allowed. When we set $x = 0$, this implies—by unit propagation—the literals p_1 and p_2, these together imply q which implies a, and finally, q and a together imply both b and $\neg b$, causing a contradiction. At this point, a clause learning algorithm will realize that the literal q forms what's called a unique implication point (UIP) for this conflict [10], and will learn the singleton clause $\neg q$. Now, when we set $x = 1$, this, along with the learned clause $\neg q$, will unit propagate one of the clauses of F_1 and imply r, which will then imply a and cause a contradiction as before. Thus, setting $x = 0$ leads to a contradiction by unit propagation as well as a learned clause, and setting $x = 1$ after this also leads to a contradiction.

To see that there is no traditional strong backdoor of size one with respect to unit propagation (and, in particular, $\{x\}$ does not work as a strong backdoor

without the help of the learned clause $\neg q$), observe that for every variable of F_1, there exists at least one polarity in which it does not appear in any 1- or 2-clause (i.e., a clause containing only 1 or 2 variables) and therefore there is no empty clause generation or unit propagation under at least one truth assignment for that variable. (Note that F_1 does not have any 1-clauses to begin with.) E.g., q does not appear in any 2-clause of F_1 and therefore setting $q = 0$ does not cause any unit propagation at all, eliminating any chance of deducing a conflict. Similarly, setting $x = 1$ does not cause any unit propagation. In general, no variable of F_1 can lead to a contradiction by itself under both truth assignments to it, and thus cannot be a traditional strong backdoor. Note that $\{x, q\}$ does form a traditional strong backdoor of size two for F_1 w.r.t. unit propagation. □

Theorem 1. *There are* unsatisfiable *SAT instances for which the smallest learning-sensitive backdoors w.r.t. unit propagation are exponentially smaller than the smallest traditional strong backdoors.*

Proof (Sketch). We, in fact, provide two proofs of this statement by constructing two unsatisfiable formulas F_2 and F_3 over $N = k + 3 \cdot 2^k$ variables and $M = 4 \cdot 2^k$ clauses, with the following property: both formulas have a learning-sensitive backdoor of size $k = \Theta(\log N)$ but no traditional strong backdoor of size smaller than $2^k + k = \Theta(N)$. F_2 is perhaps a bit easier to understand and has a relatively weak ordering requirement for the size k learning-sensitive backdoor to work (namely, that the all-1 truth assignment must be evaluated at the very end); F_3, on the other hand, requires a strict value ordering to work as a backdoor (namely, the lexicographic order from $000\ldots0$ to $111\ldots1$) and highlights the strong role a good branching order plays in the effectiveness of backdoors. For lack of space, the details are deferred to an extended Technical Report [3]. □

In fact, the discussion in the proof of Theorem 1 also reveals that for the constructed formula F_3, any value ordering that starts by assigning 0's to all x_i's will lead to a learning-sensitive backdoor of size no smaller than 2^k. This immediately yields the following result under-scoring the importance of the "right" value ordering even amongst various learning-sensitive backdoors.

Corollary 1. *There are* unsatisfiable *SAT instances for which one value ordering of the variables can lead to exponentially smaller learning-sensitive backdoors w.r.t. unit propagation than a different value ordering.*

We now turn our attention to the study of strong backdoors for *satisfiable* instances, and show that clause learning can also lead to strictly smaller (strong) backdoors for satisfiable instances. In fact, our experiments suggest a much more drastic impact of clause learning on backdoors for practical satisfiable instances than on backdoors for unsatisfiable instances. We have the following formal result that can be derived from a slight modification of the construction of formula F_1 used earlier in Example 1 (see Technical Report [3]).

Theorem 2. *There are* satisfiable *SAT instances for which there exist learning-sensitive backdoors w.r.t. unit propagation that are smaller than the smallest traditional strong backdoors.*

As a closing remark, we note that the presence of clause learning does not affect the power of weak backdoors w.r.t. a natural class of *syntactically-defined* sub-solvers, i.e., sub-solvers that work when the constraint graph of the instance satisfies a certain polynomial-time verifiable property. Good examples of such syntactic classes w.r.t. which strong backdoors have been studied in depth are 2-SAT, Horn-SAT, and RenamableHorn-SAT [cf. 2, 11, 12]. Most of such syntactic classes satisfy a natural property, namely, they are *closed under clause removal*. In other words, if F is a 2-SAT or Horn formula, then removing some clauses from F yields a smaller formula that is also a 2-SAT or Horn formula, respectively. We have the following observation (see Technical Report [3] for a proof):

Proposition 1. *Clause learning does not reduce the size of weak backdoors with respect to syntactic sub-solver classes that are closed under clause removal.*

4 Experimental Results

We evaluate the effect of clause learning on the size of backdoors in a set of well-known SAT instances from SATLIB [5]. Upper bounds on the size of the smallest leaning-sensitive backdoor w.r.t. UP were obtained using the SAT solver *RSat* [13]. At every search node *RSat* employs UP and at every conflict it employs clause learning based on UIP. We turned off restarts and randomized the variable and value selection. In addition, we traced the set of variables used for branching during search—the backdoor. We ran the modified *RSat* 5,000 times per instance and recorded the smallest backdoor set among all runs.

Upper bounds on the size of the smallest traditional backdoor w.r.t. UP were obtained using a modified version of *Satz-rand* [4, 9] that employs UP as a sub-solver and also traces the set of branch variables. We ran the modified *Satz* 5,000 times per instance and recorded the smallest backdoor set among all runs. Note that these results concern traditional weak backdoors for satisfiable instances and strong backdoors for unsatisfiable instances. *Satz* relies heavily on good variable selection heuristics in order to minimize the solution time. Hence, using *Satz* instead of a modified version of *RSat* with learning turned off gave us much better bounds on traditional backdoors w.r.t. UP.

The results are summarized in Table 1. Across all satisfiable instances the learning-sensitive backdoor upper bounds are significantly smaller than the traditional ones. For unsatisfiable instances, the upper bounds on the learning-sensitive and traditional backdoors are not very different. However, a notable exception is the *parity* instance where including clause learning reduces the backdoor upper bound to less than 10% from almost 39%.

Acknowledgments

This research was supported by IISI, Cornell University (AFOSR grant FA9550-04-1-0151), NSF Expeditions in Computing award for Computational Sustainability (Grant 0832782) and NSF IIS award (Grant 0514429). The first author was partially supported by an NSERC PGS Scholarship. Part of this work was done while the third author was visiting McGill University.

Table 1. Upper bounds on the size of the smallest backdoor when using clause learning and unit propagation (within RSat) and when using only unit propagation (within Satz). Results are given as percentage of the number of variables.

Instance	Status	Vars	Clauses	**UP+CL**	UP
bf0432-007	UNSAT	1,040	3,668	**12.12%**	13.65%
bf1355-075	UNSAT	2,180	6,778	**3.90%**	5.92%
bf1355-638	UNSAT	2,177	6,768	**3.86%**	6.84%
bf2670-001	UNSAT	1,393	3,434	**1.22%**	2.08%
apex7_gr_2pin_w4	UNSAT	1,322	10,940	**12.25%**	20.73%
parity_unsat_4_5	UNSAT	2,508	17,295	**9.85%**	39.07%
anomaly	SAT	48	261	4.17%	4.17%
medium	SAT	116	953	**1.72%**	14.66%
huge	SAT	459	7,054	**1.09%**	3.27%
bw_large.a	SAT	459	4,675	**1.53%**	3.49%
bw_large.b	SAT	1,087	13,772	**1.93%**	11.59%
bw_large.c	SAT	3,016	50,457	**2.95%**	13.76%
bw_large.d	SAT	6,325	131,973	**3.37%**	43.27%

References

[1] B. Dilkina, C. P. Gomes, Y. Malitsky, A. Sabharwal, and M. Sellmann. Backdoors to combinatorial optimization: Feasibility and optimality. In *CPAIOR*, 2009.

[2] B. Dilkina, C. P. Gomes, and A. Sabharwal. Tradeoffs in the complexity of backdoor detection. In *CP*, pp. 256–270, 2007.

[3] B. Dilkina, C. P. Gomes, and A. Sabharwal. Backdoors in the context of learning (extended version). Technical report, Cornell University, Computing and Information Science, Apr. 2009. URL http://hdl.handle.net/1813/12231.

[4] C. P. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *AAAI*, pp. 431–437, 1998.

[5] H. H. Hoos and T. Stützle. SATLIB: An online resource for research on SAT. In *SAT*, pp. 283–292, 2000. URL http://www.satlib.org.

[6] P. Kilby, J. K. Slaney, S. Thibaux, and T. Walsh. Backbones and backdoors in satisfiability. In *AAAI*, pp. 1368–1373, 2005.

[7] O. Kullmann. Investigating a general hierarchy of polynomially decidable classes of cnf's based on short tree-like resolution proofs. *ECCC*, vol. 41, 1999.

[8] O. Kullmann. Upper and lower bounds on the complexity of generalised resolution and generalised constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence*, 40(3-4):303–352, 2004. ISSN 1012-2443.

[9] C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *IJCAI*, pp. 366–371, 1997.

[10] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. In *DAC*, pp. 530–535, 2001.

[11] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *SAT*, pp. 96-103, 2004.

[12] L. Paris, R. Ostrowski, P. Siegel, and L. Sais. Computing Horn strong backdoor sets thanks to local search. In *ICTAI*, pp. 139–143, 2006.

[13] K. Pipatsrisawat and A. Darwiche. A lightweight component caching scheme for satisfiability solvers. In *SAT*, pp. 294–299, 2007.

[14] M. Samer and S. Szeider. Backdoor trees. In *AAAI*, pp. 363–368, 2008.

[15] S. Szeider. Backdoor sets for DLL subsolvers. *J. Auto. Reas.*, 35(1-3):73–88, 2005.

[16] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *IJCAI*, pp. 1173–1178, 2003.