Backdoors in the Context of Learning
(Extended Version)

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

April 10, 2009

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. Given a backdoor variable set B, a systematic search procedure is guaranteed to succeed in efficiently deciding the problem instance independent of the order in which it explores various truth valuations of the variables in B. This definition is oblivious to the fact that "learning during search" is a key feature of modern solution procedures for various classes of combinatorial problems such as SAT and mixed integer programming (MIP). These solvers carry over often highly useful information from previously explored search branches to newly considered branches. In this work, we extend the notion of backdoors to the context of learning during search. In particular, 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 oblivious to these solver features. We also provide an experimental comparison between backdoor sizes with and without learning.

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 [1, 5, 8, 9, 10, 14, 16, 17, 18]. Backdoor sets 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, pure literal elimination, and failed-literal probing. 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 will also present empirical results showing that the added power of learning-sensitive backdoors is also often observed in practice.

2 Preliminaries

Let V be a set of propositional (Boolean) variables, which take value in the set $\{0, 1\}$. We think of 1 as True and 0 as False. Let F be a propositional formula over V. A *solution* to F (also referred to as a satisfying assignment for F) is a 0-1 assignment to all variables in V such that F evaluates to 1. *Propositional Satisfiability* or SAT is the decision problem of determining whether an input formula F has any solutions. This is the canonical NP-complete problem. In practice, one is also interested in finding a solution, if there exists one.

Instances of the SAT problem are often specified in the Conjunctive Normal Form (CNF). Here F is given as a conjunction of *clauses*, each clause is a disjunction of *literals*, and each literal is either a variable or its negation. For example, $F = (a \vee \neg b) \wedge (\neg a \vee c)$ is a CNF formula.

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, 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 studied as sub-solvers. Formally [17], a *sub-solver A for SAT* is any polynomial time algorithm satisfying certain natural properties on every input 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 constraints or an already violated constraint; 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 [17]). Given a Boolean 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 [2]. In practice, modern SAT solvers such as *RSat* [15], *Minisat* [4] and *zChaff* [12] 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. As we will show shortly through a carefully constructed example, clause learning can in fact make an exponential difference in the backdoor size. 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 Boolean 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 $\tau : B \rightarrow \{0, 1\}$ 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 explain the power of learning-sensitivity through the following example formula, for which there is a natural learning-sensitive backdoor of size 1 with respect to 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 consisting of the following 10 clauses:

$$(x \vee p_1), (x \vee p_2), (\neg p_1 \vee \neg p_2 \vee q), \quad (\neg q \vee a), (\neg q \vee \neg a \vee b), (\neg q \vee \neg a \vee \neg b)$$
$$(\neg x \vee q \vee r), \quad (\neg r \vee a), (\neg r \vee \neg a \vee b), (\neg r \vee \neg a \vee \neg b)$$

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,

and will learn the singleton clause $\neg q$.[1] 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 $2^k + k = \Theta(N)$. The formula 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); the formula 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. The details are deferred to the Appendix. □ □

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. We show that clause learning not only helps with unsatisfiable instances, it can also lead to 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 result that can be derived from a slight modification of the construction of formula F_1 used earlier in Example 1 (see Appendix).

[1] We omit the details of the 1-UIP clause learning mechanism and refer the interested reader to Moskewicz et al. [12]

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. By syntactically defined sub-solvers, we mean sub-solvers that are able to determine problem instances if 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. 3, 13]. 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. Please refer to the Appendix for a formal proof of the following statement:

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

In order to experimentally evaluate the effect of clause learning on the size of backdoors, we computed upper bounds on the smallest size of learning-sensitive backdoors w.r.t. unit propagation and of traditional backdoors w.r.t. unit propagation. We used some well-known SAT instances from SATLIB [7].

Leaning-sensitive backdoor size results were obtained using the SAT solver *RSat* [15]. At every search node *RSat* employs unit propagation 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 included code that traced the set of variables used for branching during search, i.e. the backdoor. We ran the modified *RSat* 5000 times per instance and recorded the smallest backdoor set among all runs.

While *RSat* is mainly geared to utilize clause learning, it might not be very good at finding small set of branching variables. On the other hand, the SAT solver *Satz*, which does not use clause-learning, relies heavily on good variable selection heuristics in order to minimize the solution time. Hence *Satz* is better at discovering smaller branching sets than *RSat*. For this reason, we used a modified version of *Satz-rand* [6, 11] that uses unit propagation as a sub-solver and also traces the set of branch variables. We ran the modified *Satz* 5000 times per instance and recorded the smallest backdoor set among all runs. This gives us an upper bound on the actual minimum weak backdoor size for satisfiable instances and strong backdoor size for unsatisfiable instances. Using *Satz* instead of a modified version of *RSat* with learning turned off gave us much better bounds on traditional UP backdoors.

The results are summarized in Table 1. Across all satisfiable instances the learning-sensitive backdoor upper bounds are significantly smaller than the traditional ones. In 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%.

Table 1: Upper bounds on the minimum backdoor size 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.

Inst	Status	Var	Clause	UP+CL	UP
bf0432-007	UNSAT	1040	3668	12.12%	13.65%
bf1355-075	UNSAT	2180	6778	3.90%	5.92%
bf1355-638	UNSAT	2177	6768	3.86%	6.84%
bf2670-001	UNSAT	1393	3434	1.22%	2.08%
apex7_gr_2pin_w4	UNSAT	1322	10940	12.25%	20.73%
parity_unsat_4_5	UNSAT	2508	17295	9.85%	39.07%
anomaly	SAT	48	261	4.17%	4.17%
medium	SAT	116	953	1.72%	14.66%
huge	SAT	459	7054	1.09%	3.27%
bw_large.a	SAT	459	4675	1.53%	3.49%
bw_large.b	SAT	1087	13772	1.93%	11.59%
bw_large.c	SAT	3016	50457	2.95%	13.76%
bw_large.d	SAT	6325	131973	3.37%	43.27%

Acknowledgments

A short version of this Technical Report may be found in the Proceedings of SAT-09, the 12-th International Conference on Theory and Applications of Satisfiability Testing, Swansea, Wales, United Kingdom, June 2009.

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 the NSERC PGS Fellowship. Part of this work was done while the third author was visiting McGill University.

References

[1] H. Chen, C. P. Gomes, and B. Selman. Formal models of heavy-tailed behavior in combinatorial search. In *7th CP*, vol. 2239 of *LNCS*, Paphos, Cyprus, Nov. 2001.

[2] B. Dilkina, C. P. Gomes, Y. Malitsky, A. Sabharwal, and M. Sellmann. Backdoors to combinatorial optimization: Feasibility and optimality. In *6th CPAIOR*, 2009.

[3] B. Dilkina, C. P. Gomes, and A. Sabharwal. Tradeoffs in the complexity of backdoor detection. In *13th CP*, vol. 4741 of *LNCS*, pp. 256–270, Providence, RI, Sept. 2007.

[4] N. Eén and N. Sörensson. MiniSat: A SAT solver with conflict-clause minimization. In *8th SAT*, St. Andrews, U.K., June 2005.

[5] C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. Auto. Reas.*, 24(1-2):67–100, 2000.

[6] C. P. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *15th AAAI*, pp. 431–437, Madison, WI, July 1998.

[7] H. H. Hoos and T. Stützle. SATLIB: An online resource for research on SAT. In I. P. Gent, H. van Maaren, and T. Walsh, editors, *SAT2000*, pp. 283–292. IOS Press, 2000. URL http://www.satlib.org.

[8] P. Kilby, J. K. Slaney, S. Thibaux, and T. Walsh. Backbones and backdoors in satisfiability. In *20th AAAI*, pp. 1368–1373, 2005.

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

[10] 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.

[11] C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *15th IJCAI*, pp. 366–371, Nagoya, Japan, Aug. 1997.

[12] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. In *38th DAC*, pp. 530–535, Las Vegas, NV, June 2001.

[13] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *7th SAT*, Vancouver, BC, Canada, May 2004.

[14] L. Paris, R. Ostrowski, P. Siegel, and L. Sais. Computing Horn strong backdoor sets thanks to local search. *ICTAI-06: 17th ICTAI*, pp. 139–143, Nov. 2006.

[15] K. Pipatsrisawat and A. Darwiche. A lightweight component caching scheme for satisfiability solvers. In *10th SAT*, pp. 294–299, 2007.

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

[17] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *18th IJCAI*, pp. 1173–1178, Acapulco, Mexico, Aug. 2003.

[18] R. Williams, C. Gomes, and B. Selman. On the connections between heavy-tails, backdoors, and restarts in combinatorial search. In *6th SAT*, pp. 222–230, Santa Margherita Ligure, Italy, May 2003.

Appendix: Proof and Construction Details

Proof of Theorem 1. 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 $2^k + k = \Theta(N)$. The formula 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); the formula 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.

Let σ denote a string of k 0's and 1's, such as $101\ldots0$. This naturally corresponds to a truth assignment to k Boolean variables. The variables of both F_2 and F_3 will be

$$\{x_i \mid 1 \le i \le k\} \bigcup \{q_\sigma, a_\sigma, b_\sigma \mid \sigma \text{ is a 0-1 string of length } k\}$$

For a string σ, consider the unique clause C_σ over the k variables x_i that falsifies σ. E.g., for $k = 3$ and $\sigma = 001$, the clause C_{001} is $(x_1 \vee x_2 \vee \neg x_3)$. One of the strings, w.l.o.g. the all-1's string $111\ldots1$, will play a special role below, and for succinctness we denote it by the bold typesetting $\tilde{\mathbf{1}}$. The formula F_2 is defined as follows:

$$F_2 = \bigwedge_{\sigma \neq \tilde{\mathbf{1}}} ((C_\sigma \vee q_\sigma) \wedge (\neg q_\sigma \vee a_\sigma) \wedge (\neg q_\sigma \vee \neg a_\sigma \vee b_\sigma) \wedge (\neg q_\sigma \vee \neg a_\sigma \vee \neg b_\sigma))$$
$$\bigwedge \left(\left(C_{\tilde{\mathbf{1}}} \vee \bigvee_{\sigma'} q_{\sigma'} \right) \wedge (\neg q_{\tilde{\mathbf{1}}} \vee a_{\tilde{\mathbf{1}}}) \wedge (\neg q_{\tilde{\mathbf{1}}} \vee \neg a_{\tilde{\mathbf{1}}} \vee b_{\tilde{\mathbf{1}}}) \wedge (\neg q_{\tilde{\mathbf{1}}} \vee \neg a_{\tilde{\mathbf{1}}} \vee \neg b_{\tilde{\mathbf{1}}}) \right)$$

Notice that the set of clauses in F_2 corresponding to the $2^k - 1$ strings $\sigma \neq \tilde{\mathbf{1}}$ are similar in structure and involve distinct variables other than the k x_i variables. The first clause corresponding to set for the string $\tilde{\mathbf{1}}$ is, however, much longer—it includes *all* 2^k q variables, along with the k x_i variables. The difference in the formula F_3 below is that, if we think of the 2^k strings σ as being ordered lexicographically, the first clause in the set for σ includes the q variables for all preceding σ's. More precisely, using $\preceq$ for the lexicographic order over the strings σ, we have

$$F_3 = \bigwedge_{\sigma} \left(\left(C_\sigma \vee \bigvee_{\sigma' \preceq \sigma} q_{\sigma'} \right) \wedge (\neg q_\sigma \vee a_\sigma) \wedge (\neg q_\sigma \vee \neg a_\sigma \vee b_\sigma) \wedge (\neg q_\sigma \vee \neg a_\sigma \vee \neg b_\sigma) \right)$$

We claim that $B = \{x_i \mid 1 \leq i \leq k\}$ forms a learning-sensitive backdoor for both F_2 and F_3 w.r.t. the unit propagation sub-solver. To see this, consider the all-0's assignment to the variables in B. This assignment falsifies $C_{000\ldots0}$ and by unit propagation implies $q_{000\ldots0}$ in both F_2 and F_3. As earlier in Example 1, this implies $a_{000\ldots0}$ and eventually soon leads to a contradiction. At this point, the search procedure, again as in the example earlier, learns the singleton 1-UIP clause $\neg q_{000\ldots0}$. Now, consider the next string in the lexicographic order, namely, $000\ldots1$, which falsifies $C_{000\ldots1}$. In F_2, this immediately implies $q_{000\ldots1}$, while in F_3, this along with the learned clause $\neg q_{000\ldots0}$ implies $q_{000\ldots1}$. In either case, we continue as before, derive a contradiction, and learn the singleton clause $\neg q_{000\ldots1}$. The process continues till we arrive at the last string, $\sigma = \tilde{\mathbf{1}}$. Now both F_2 and F_3 use *all* of the $2^k - 1$ clauses learned so far in order to deduce $q_{\tilde{\mathbf{1}}}$ and thus derive a contradiction. This completes the proof that the set B forms a learning-sensitive backdoor for both F_2 and F_3.

Note that the use of learned clauses was crucial in the above refutations. In the case of traditional strong backdoors, we will, unfortunately, not have access to learned clauses. To see that all traditional strong backdoors for F_2 and F_3 will need at least $2^k + k$ variables, we make use of the observation that these formulas are minimally unsatisfiable, i.e., removing any single clause turns them into a satisfiable formula. It follows that any proof of unsatisfiability must "make use" of all clauses of these formulas. Specifically, consider the longest clause, $C_{\text{long}} \equiv (C_{\tilde{\mathbf{1}}} \vee \bigvee_{\sigma'} q_{\sigma'})$, which involves $2^k + k$ variables. Consider a partial truth assignment τ that satisfies all clauses of the formula other than C_{long}; τ exists because of the formula being minimally unsatisfiable. It follows that for any traditional backdoor set B, the truth assignment $\tau|_B$ consistent with σ cannot cause any unit propagation through clauses other than C_{long}, nor can it generate an empty clause and deduce a conflict through these other clauses themselves. To prove the formula unsatisfiable under truth assignment τ, the sub-solver must therefore deduce that clause C_{long} is violated. Without access to previously learned clauses, all literals of C_{long} must be negated by either being included in B and assigned truth values, or because of being implied by unit propagation of other clauses—which we argued does not happen for τ. Thus, we conclude that all variables of C_{long} must belong to B, implying $|B| \geq 2^k + k$ as claimed. □

Proof of Theorem 2. Consider a satisfiable variant F_1^{SAT} of the instance F_1 from Example 1, obtained by replacing the last clause, $(\neg r \vee \neg a \vee \neg b)$ with the clause $(\neg r \vee p_1)$. It is easy to see that F_1^{SAT} is satisfiable, with $(x, p_1, p_2, q, r, a, b) = (1, 1, 0, 0, 1, 1, 1)$ being the only solution. As before, for every variable of F_1^{SAT}, there exists at least one polarity in which it does not occur in any 1- or 2-clause. Furthermore, no literal appears in all clauses, and so satisfying just that literal cannot make the unit propagation sub-solver conclude that F_1^{SAT} is satisfied by the current partial truth

assignment. Putting these two observations together, we have that fixing the value of a variable to at least one polarity does not cause any unit propagation and does not automatically satisfy all clauses. Therefore, this variable by itself is not a traditional strong backdoor for F_1^{SAT}.

On the other hand, $\{x\}$ again forms a learning-sensitive backdoor for this formula. To see this, note that when $x = 0$, we deduce as conflict as before for F_1 and learn the singleton clause $\neg q$. When x is not set to 1, this, together with the learned clause $\neg q$, imply r, which implies a and then b. r also implies p_1, which, together with $\neg q$, implies $\neg p_2$. At this point, all variables have truth values assigned by unit propagation and all clauses are satisfied. Thus, $\{x\}$ correctly serves as a learning-sensitive backdoor for F_1^{SAT}. □

Proof of Proposition 1. Let F be a CNF formula for which B is a learning-sensitive backdoor w.r.t. a syntactic sub-solver class $\mathcal{C}$ closed under clause removal. Let τ be the final truth assignment to B explored by the clause learning SAT solver before solving F through the use of the sub-solver. In other words, the resulting formula F', consisting of $F|_\tau$ along with all clauses learned so far and restricted by τ, is in the class $\mathcal{C}$. However, since $\mathcal{C}$ is closed under clause removal, this means that $F|_\tau$ itself belongs to $\mathcal{C}$. Hence, B is also a traditional weak backdoor for F w.r.t. the sub-solver class $\mathcal{C}$, as witnessed by the truth assignment τ. □