Tradeoffs in the Complexity of Backdoor Detection

Bistra Dilkina, Carla P. Gomes, and Ashish Sabharwal

Cornell University, Department of Computer Science, Ithaca, NY 14850, USA
{bistra,gomes,sabhar}@cs.cornell.edu

Abstract. There has been considerable interest in the identification of structural properties of combinatorial problems that lead to efficient algorithms for solving them. Some of these properties are "easily" identifiable, while others are of interest because they capture key aspects of state-of-the-art constraint solvers. In particular, it was recently shown that the problem of identifying a strong Horn- or 2CNF-backdoor can be solved by exploiting equivalence with deletion backdoors, and is NP-complete. We prove that strong backdoor identification becomes harder than NP (unless NP=coNP) as soon as the inconsequential sounding feature of empty clause detection (present in all modern SAT solvers) is added. More interestingly, in practice such a feature as well as polynomial time constraint propagation mechanisms often lead to much smaller backdoor sets. In fact, despite the worst-case complexity results for strong backdoor detection, we show that Satz-Rand is remarkably good at finding small strong backdoors on a range of experimental domains. Our results suggest that structural notions explored for designing efficient algorithms for combinatorial problems should capture both statically and dynamically identifiable properties.

1 Introduction

Capturing and exploiting problem structure is key to solving large real-world combinatorial problems. For example, several interesting tractable classes of combinatorial problems have been identified by restricting the constraint language used to characterize such problem instances. Well-known cases include 2CNF, Horn, Linear Programming (LP), and Minimum Cost Flow problems (MCF). In general, however, such restricted languages are not rich enough to characterize complex combinatorial problems. A very fruitful and prolific line of research that has been pursued in the study of combinatorial problems is the identification of various structural properties of instances that lead to efficient algorithms. Ideally, one prefers structural properties that are "easily" identifiable, such as from the topology of the underlying constraint graph. As an example, the degree of acyclicity of a constraint graph, measured using various graph width parameters, plays an important role with respect to the identification of tractable instances — it is known that an instance is solvable in polynomial time if the treewidth of its constraint graph is bounded by a constant [8, 9, 6, 10, 5, 21]. Interestingly, even though the notion of bounded treewidth is defined with respect

to tree decompositions, it is also possible to design algorithms for constraint satisfaction problems of bounded (generalized) hypertree width that do not perform any form of tree decomposition (see e.g., [3]). Other useful structural properties consider the nature of the constraints, such as their so-called functionality, monotonicity, and row convexity [7, 24].

Another approach for studying combinatorial problems focuses on the role of *hidden structure* as a way of analyzing and understanding the efficient performance of state-of-the-art constraint solvers on many real-world problem instances. 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 [25, 26, 12, 4, 15, 23, 20]. Note that the notion of tractability in the definition of backdoor sets is not necessarily syntactically defined: it may often be defined only by means of a polynomial time algorithm, such as unit propagation. In fact, the notion of backdoor sets came about as a way of explaining the high variance in performance of state-of-the-art SAT solvers, in particular heavy-tailed behavior, and as a tool for analyzing and understanding the efficient performance of these solvers on many real-world instances, in which the propagation mechanisms of fast "sub-solvers" play a key role. In this work the emphasis was not so much on efficiently identifying backdoor sets, but rather on the fact that many real-world instances have surprisingly small sets of backdoor variables and that once a SAT solver instantiates these variables, the rest of the problem is solved easily. In this context, randomization and restarts play an important role in searching for backdoor sets [25, 26].

Even though variable selection heuristics, randomization, and learning in current SAT/CSP solvers are quite effective at finding relatively small backdoors in practice, finding a *smallest backdoor* is in general intractable in the worst case. This intractability result assumes that the size of the smallest backdoor is unknown and can grow arbitrarily with n. However, if the size of the backdoor is small and fixed to k, one can search for the backdoor by considering all $\binom{n}{k}$ subsets of k variables and all 2^k truth assignments to these candidate variables. This is technically a polynomial time process for fixed k, although for moderate values of k the run time becomes infeasible in practice. Can one do better? This is a question considered in the area of fixed-parameter complexity theory. A problem with input size n and a parameter k is called *fixed-parameter tractable* w.r.t. k if it can be solved in time $O(f(k)n^c)$ where f is any computable function and c is a constant. Note that c does not depend on k, meaning that one can in principle search fairly efficiently for potentially large backdoors if backdoor detection for some class is shown to be fixed parameter tractable. Indeed, Nishimura, Ragde, and Szeider [19] showed that detecting strong backdoors (cf. Section 2 for a formal definition) w.r.t. the classes 2CNF and Horn is NP-complete but, interestingly, fixed-parameter tractable. This result for 2CNF and Horn formulas exploits the equivalence between (standard) strong backdoors and "deletion" backdoors, i.e., a set of variables that once deleted from a given formula (without simplification) make the remaining formula tractable. Note, however, that this result is only w.r.t. the tractable classes of *pure* 2CNF/Horn.

In particular, certain kinds of obvious inconsistencies are not detected in these classes, such as having an empty clause in an *arbitrary* formula — clearly, any basic solver detects such inconsistencies. We show that such a seemingly small feature increases the worst-case complexity of backdoor identification, but, perhaps more importantly, can dramatically reduce the size of the backdoor sets.

More specifically, we prove that strong Horn- and 2CNF-backdoor identification becomes both NP- and coNP-hard, and therefore strictly harder than NP assuming NP $\neq$ coNP, as soon as empty clause detection is added to these classes. This increase in formal complexity has however also a clear positive aspect in that adding empty clause detection often considerably reduces the backdoor size. For example, in certain graph coloring instances with planted cliques of size 4, while strong Horn-backdoors involve $\approx 67\%$ of the variables, the fraction of variables in the smallest strong backdoors w.r.t. mere empty clause detection converges to 0 as the size of the graph grows.

Encouraged by the positive effect of slightly extending our notion of Horn-backdoor, we also consider backdoors w.r.t. RHorn (renamable Horn), UP (unit propagation), PL (pure literal rule), UP+PL, and SATZ. For each of these notions, we show on a variety of domains that the corresponding backdoors are significantly smaller than pure, strong Horn-backdoors. For example, we consider the smallest deletion RHorn-backdoors. We provide a 0-1 integer programming formulation for finding such optimal backdoors, and show experimentally that they are in general smaller than strong Horn-backdoors. In particular, in the graph coloring domain, while strong Horn-backdoors correspond to $\approx 67\%$ of the variables, deletion RHorn-backdoors correspond to only $\approx 17\%$ of the variables. More interestingly, when considering real-world instances of a car configuration problem, while strong Horn-backdoor sets vary in size between 10-25% of the variables, deletion RHorn-backdoor sets vary only between 3-8%.

At a higher level, our results show that the size of backdoors can vary dramatically depending on the effectiveness of the underlying simplification and propagation mechanism. For example, as mentioned earlier, empty clause detection can have a major impact on backdoor size. Similarly, Horn versus RHorn has an impact. We also show that there can be a substantial difference between deletion backdoors, where one simply removes variables from the formula, versus strong backdoors, where one factors in the variable settings and considers the propagation effect of these settings. Specifically, we contrast deletion RHorn-backdoors with strong RHorn-backdoors. We prove by construction that there are formulas for which deletion RHorn-backdoors are exponentially larger than the smallest strong RHorn-backdoors.

Finally, despite the worst-case complexity results for strong backdoor detection, we show that *Satz-Rand* [16, 11] is remarkably good at finding small strong backdoors on a range of experimental domains. For example, in the case of our graph coloring instances, the fraction of variables in a small strong SATZ-backdoor converges to zero as the size of the graph grows. For the car configuration problem, strong SATZ-backdoor sets involve 0-0.7% of the variables. We next consider synthetic logistics planning instances over n variables that are

known to have strong UP-backdoors of size $\log n$ [13]. For all these instances, the size of the strong SATZ-backdoor sets is either zero or one. In contrast, the size of deletion RHorn-backdoors corresponds to over 48% of the variables, increasing with n. We also consider instances from game theory for which one is interested in determining whether there is a pure Nash equilibrium. For these instances, while strong Horn-backdoors and deletion RHorn-backdoors involve $\approx 68\%$ and $\approx 67\%$ of the variables, respectively, strong SATZ-backdoors are surprising small at less than 0.05% of the variables.

These results show that real-world SAT solvers such as *Satz* are indeed remarkably good at finding small backdoors sets. At a broader level, this work suggests that the study of structural notions that lead to efficient algorithms for combinatorial problems should consider not only "easily" identifiable properties, such as being Horn, but also properties that capture key aspects of state-of-the-art constraint solvers, such as unit propagation and pure literal rule.

2 Preliminaries and Related Work

A *CNF formula* F is a conjunction of a finite set of clauses, a *clause* is a disjunction of a finite set of literals, and a *literal* is a Boolean variable or its negation. The literals associated with a variable x are denoted by x^ϵ, $\epsilon \in \{0,1\}$. $var(F)$ denotes the variables occurring in F. A (partial) *truth assignment* (or assignment, for short) is a map $\tau : X_\tau \to \{0,1\}$ defined on some subset of variables $X_\tau \subseteq var(F)$. A *solution* to a CNF formula F is a complete variable assignment τ (i.e., with $X_\tau = var(F)$) that satisfies all clauses of F. $F[\epsilon/x]$ denotes the simplified formula obtained from F by removing all clauses that contain the literal x^ϵ and removing, if present, the literal $x^{1-\epsilon}$ from the remaining clauses. For a partial truth assignment τ, $F[\tau]$ denotes the simplified formula obtained by setting the variables according to τ.

A *unit clause* is a clause that contains only one literal. A *pure literal* in F is a literal x^ϵ such that $x \in var(F)$ and $x^{1-\epsilon}$ does not occur in F. A *Horn clause* is a clause that contains at most one positive literal. A *binary clause* is a clause that contains exactly two literals. A formula is called *Horn* (resp., *2CNF*) if all its clauses are Horn (binary). We also use Horn and 2CNF to denote the two corresponding classes of formulas. *Renaming* or flipping a variable x in F means replacing every occurrence of x^ϵ in F with $x^{1-\epsilon}$. F is *Renamable Horn*, also called RHorn, if all clauses of F can be made Horn by flipping a subset of the variables. Following Nishimura et al. [19], we define the *deletion* of a variable x from a formula F as syntactically removing the literals of x from F: $F - x = \left\{c \setminus \left\{x^0, x^1\right\} \mid c \in F\right\}$. For $X \subseteq var(F)$, $F - X$ is defined similarly.

The concept of backdoors and their theoretical foundations were introduced by Williams, Gomes, and Selman [25, 26]. Informally, a strong backdoor set is a set of variables such that for each possible truth assignment to these variables, the simplified formula is tractable. The notion of tractability is quite general, and it even includes tractable classes for which there is not a clean syntactic characterization. It is formalized in terms of a polynomial time sub-solver:

Definition 1 (sub-solver [25]). *A* sub-solver S *is an algorithm that given as input a formula* F *satisfies the following conditions:*

1. Trichotomy: S *either rejects* F *or correctly determines it (as unsatisfiable or satisfiable, returning a solution if satisfiable),*
2. Efficiency: S *runs in polynomial time,*
3. Trivial solvability: S *can determine if* F *is trivially true (has no clauses) or trivially false (has an empty clause,* $\{\}$*), and*
4. Self-reducibility: *If* S *determines* F*, then for any variable* x *and value* $\epsilon \in \{0,1\}$*,* S *determines* $F[\epsilon/x]$*.*

Definition 2 (strong S-backdoor [25]). *A set* B *of variables is a* strong backdoor set *for a formula* F *w.r.t a sub-solver* S *if* $B \subseteq var(F)$ *and for every truth assignment* $\tau : B \to \{0,1\}$*,* S *returns a satisfying assignment for* $F[\tau]$ *or concludes that* $F[\tau]$ *is unsatisfiable.*

Clearly, if B is a strong S-backdoor for F, then so is any B' such that $B \subseteq B' \subseteq var(F)$. For any sub-solver S, given $\langle F, k\rangle$ as input, the problem of deciding whether F has a strong S-backdoor of size k is in the complexity class Σ_2^{P}: we can formulate it as, "does *there exist* a $B \subseteq var(F), |B| = k$, such that *for every* truth assignment $\tau : B \to \{0,1\}$, S correctly determines $F[\tau/B]$?" We are interested in the complexity of this problem for specific sub-solvers.

The most *trivial sub-solver* that fulfills the conditions in Definition 1 is the one that only checks for the empty formula and for the empty clause. Lynce and Marques-Silva [17] show that the search effort required by the SAT solver *zChaff* [18] to prove a random 3-SAT formula unsatisfiable is correlated with the size of the strong backdoors w.r.t. this trivial sub-solver.

More relevant sub-solvers employed by most state-of-the-art SAT solvers are Unit Propagation and Pure Literal Elimination, and their combination. Given a formula F, the Unit Propagation sub-solver (UP) checks whether the formula is empty or contains the empty clause, in which case it is trivially solvable, otherwise it checks whether the formula contains a unit clause. If yes, it assigns the variable in the unit clause the corresponding satisfying value, and recurses on the simplified formula. If the formula does not contain any more unit clauses, it is rejected. The Pure Literal Elimination sub-solver (PL) checks for variables that appear as pure literals, assigning them the corresponding value and simplifying, until the formula is trivially solvable or is rejected (when no more pure literals are found). The sub-solver that uses both of these rules is referred to as UP+PL.

We note that unit propagation by itself is known to be sufficient for computing a satisfying assignment for any satisfiable Horn formula: set variables following unit propagation until there are no more unit clauses, and set the remaining variables to 0. A similar result is known for RHorn formulas. Interestingly, this does not mean that the smallest UP-backdoors are never larger than Horn- and RHorn-backdoors. For example, any (satisfiable) Horn formula with $k \geq 2$ literals per clause has a strong Horn-backdoor of size zero but no strong UP-backdoor of size $k - 2$.

Szeider [23] studied the complexity of finding strong backdoors w.r.t. the above sub-solvers. For $S \in \{\text{UP, PL, UP+PL}\}$ and with k as the parameter of interest, he proved that the problem of deciding whether there exists a strong C-backdoor of size k is complete for the parameterized complexity class W[P]. Interestingly, the naïve brute-force procedure for this problem is already in W[P]; it has complexity $O(n^k 2^k n^\alpha)$ and works by enumerating all subsets of size $\leq k$, trying all assignments for each such subset, and running the $O(n^\alpha)$ time sub-solver. Hence, the in the worst case we cannot hope to find a smallest strong backdoor w.r.t. UP, PL, or UP+PL more efficiently than with brute-force search.

Satz [16] is a DPLL-based SAT solver that incorporates a strong variable selection heuristic and an efficient simplification strategy based on UP and PL. Its simplification process and lookahead techniques can be thought of as a very powerful sub-solver. Kilby et al. [15] study strong SATZ-backdoors: sets of variables such that for every assignment to these variables, *Satz* solves the simplified formula without any branching decisions (i.e., with a "branch-free" search). They measure problem hardness, defined as the logarithm of the number of search nodes required by *Satz*, and find that it is correlated with the size of the smallest strong SATZ-backdoors.

A sub-solver S correctly determines a subclass of CNF formulas and rejects others, and hence implicitly defines the class C_S of formulas that it can determine. A natural variation of the definition of the backdoor does not explicitly appeal to a sub-solver, but rather requires the remaining formula, after setting variables in the backdoor, to fall within a known tractable sub-class, such as 2CNF, Horn, or RHorn. We will refer to such backdoors as Horn-backdoor, RHorn-backdoor, etc. Note that this way of defining the backdoor de facto corresponds to relaxing the assumption of the sub-solver's trivial solvability and therefore trivially satisfiable or unsatisfiable formulas need not lie within the tractable class. For example, an arbitrary formula with an empty clause may not be Horn. Such formulas — with an empty clause in them — are important for our discussion and we use the following notation:

Definition 3. *$C_{\{\}}$ is the class of all formulas that contain the empty clause, $\{\}$. For any class C of formulas, $C^{\{\}}$ denotes the class $C \cup C_{\{\}}$.*

We will show that strong backdoors w.r.t. $\text{2CNF}^{\{\}}$ and $\text{Horn}^{\{\}}$ behave very differently, both in terms of the complexity of finding backdoors as well as backdoor size, compared to 2CNF and Horn. In our arguments, we will use two properties of formula classes defined next.

Definition 4. *A class C of formulas is* closed under removal of clauses *if removing arbitrary clauses from any formula in C keeps the formula in C.*

Definition 5. *A class C of formulas is said to* support large strong backdoors *if there exists a polynomial (in k) time constructible family $\{G_k\}_{k\geq 0}$ of formulas such that the smallest strong C-backdoors of G_k have size larger than k.*

Note that (pure) 2CNF, Horn, and RHorn are closed under removal of clauses, while $C^{\{\}}$ is in general not: removing the empty clause may put a formula outside

$C^{\{\}}$. Further, 2CNF and Horn support large strong backdoors as witnessed by the following simple single-clause family of formulas: $G_k = (x_1 \vee x_2 \vee \ldots \vee x_{k+3})$. It can be easily verified that the all-0's assignment to any set of k variables of G_k leaves a clause with three positive literals, which is neither 2CNF nor Horn.

A different notion of backdoors, motivated by the work of Nishimura et al. [19], involves a set of variables such that once these variables are "deleted" from the formula, the remaining formula falls into a given tractable class (without considering any simplification due to truth assignments). Formally,

Definition 6 (deletion C-backdoor). *A set B of variables is a deletion backdoor set of a formula F w.r.t. a class C if $B \subseteq var(F)$ and $F - B \in C$.*

When membership in C can be checked in polynomial time, the problem of deciding whether F has a deletion C-backdoor of size k is trivially in NP. This problem is in fact NP-complete when C is 2CNF [19], Horn [19], or RHorn [2].

In general, a deletion C-backdoor may not be a strong C-backdoor. E.g., when C includes $C_{\{\}}$, any 3CNF formula F has a trivial deletion C-backdoor of size 3: select any clause and use its variables as the deletion backdoor. Unfortunately, such a "backdoor" set is of limited practical use for efficiently solving F. When the class C is closed under removal of clauses, every deletion C-backdoor is indeed also a strong C-backdoor. Conversely, strong C-backdoors often are not deletion C-backdoors, because assigning values to variables usually leads to further simplification of the formula. Nonetheless, for $C \in \{\text{2CNF, Horn}\}$, deletion and strong backdoors are equivalent, a key fact underlying the fixed parameter algorithm of Nishimura et al. [19]. We will show that this equivalence between deletion backdoors and strong backdoors does not hold for RHorn.

Paris et al. [20] studied deletion RHorn-backdoors. They proposed a two step approach: find a renaming that maximizes the number of Horn clauses using a local search method and then greedily delete variables from the remaining non-Horn clauses until the renamed formula becomes Horn. The variables deleted in the second step form a deletion RHorn-backdoor. They find that branching on these variables can significantly speed up DPLL solvers.

3 Theoretical Results

In this section, we first prove that the two problems of deciding whether a formula has a strong backdoor w.r.t. $\text{2CNF}^{\{\}}$ and $\text{Horn}^{\{\}}$, respectively, are NP-hard as well as coNP-hard. This shows that unless NP=coNP, this problem is much harder than detecting strong backdoors w.r.t. 2CNF and Horn, which are both known to be in NP [19]. Recall that adding $C_{\{\}}$ to 2CNF and Horn corresponds to adding empty clause detection to the two classes. We then consider the class RHorn and prove that strong RHorn-backdoors can be exponentially smaller than deletion RHorn-backdoors, and are therefore more likely to succinctly capture structural properties of interest in formulas.

Lemma 1. *Let $C \in \{$Horn, 2CNF$\}$. Given a formula F and $k \geq 0$, the problem of deciding whether F has a strong $C^{\{\}}$-backdoor of size k is NP-hard.*

Proof. We extend the argument originally used by Nishimura et al. [19] for (pure) 2CNF/Horn. The polynomial time reduction is from the NP-complete Vertex Cover problem: given an undirected graph $G = (V, E)$ and a number $k \geq 0$, does G have a vertex cover of size k? Recall that a vertex cover U is a subset of V such that every edge in E has at least one end point in U. Given an instance $\langle G, k\rangle$ of this problem, we will construct a formula F_{Horn} with all positive literals such that F_{Horn} has a strong Horn$^{\{\}}$-backdoor of size k iff G has a vertex cover of size k. Similarly, we will construct F_{2CNF}.

F_{Horn} has $|V|$ variables and $|E|$ clauses. The variables are x_v for each $v \in V$. For each edge $e = \{u, v\} \in E, u < v$, F_{Horn} contains the binary clause $(x_u \vee x_v)$. It is easy to see that if G has a vertex cover U, then the corresponding variable set $X_U = \{x_u \mid u \in U\}$ is a strong Horn$^{\{\}}$-backdoor: for any assignment τ to X_U, $F[\tau/X_U]$ only contains unit clauses or the empty clause, and is thus in Horn$^{\{\}}$. For the other direction, suppose X_U is a strong Horn$^{\{\}}$-backdoor. We claim that variables of X_U must touch every clause of F_{Horn} so that the corresponding vertices U touch every edge of G and thus form a vertex cover. To see this, consider the all-1's assignment τ_1 to X_U. Since X_U is a strong Horn$^{\{\}}$-backdoor and assigning variables according to τ_1 cannot result in creating the empty clause, $F_{\text{Horn}}[\tau_1/X_U]$ must be Horn. If X_U did not touch a clause c of F_{Horn}, then c would appear in $F_{\text{Horn}}[\tau_1/X_U]$ as a binary clause with two positive literals, violating the Horn property. Hence, the claim holds.

F_{2CNF} has $|V| + |E|$ variables and $|E|$ clauses. The variables are x_v for each $v \in V$ and $y_{u,v}$ for each $\{u, v\} \in E, u < v$. For each such $\{u, v\}$, F_{2CNF} contains the ternary clause $(x_u \vee x_v \vee y_{u,v})$. The argument for the correctness of the reduction is very similar to above, relying on the all-1's assignment. The only difference is that if we have a strong backdoor X_U, it may contain some of the y variables, so that there is no direct way to obtain a vertex cover out of X_U. However, this is easy to fix. If X_U contains $y_{u,v}$ and also at least one of x_u and x_v, we can simply disregard $y_{u,v}$ when constructing a vertex cover. If X_U contains $y_{u,v}$ but neither x_u nor x_v, we can replace $y_{u,v}$ with either of these two variables and obtain a backdoor set with fewer (and eventually no) such y variables. □

We now prove coNP-hardness of backdoor detection w.r.t. Horn$^{\{\}}$ and 2CNF$^{\{\}}$, exploiting the notions introduced in Definitions 4 and 5.

Lemma 2. *Let C be a class of formulas such that (1) C is closed under removal of clauses and (2) C supports large strong backdoors. Then, given a formula F and $k \geq 0$, the problem of deciding whether F has a strong $C^{\{\}}$-backdoor of size k is coNP-hard.*

Proof. Let UNSAT denote the coNP-complete problem of deciding whether a given CNF formula is unsatisfiable. We prove the lemma by reducing UNSAT to $C^{\{\}}$-backdoor detection. Let H be a CNF formula over variables V_H, $|V_H| = k$. We create a formula F such that F has a strong $C^{\{\}}$-backdoor of size k iff H is unsatisfiable. The idea is to start with H and append to it a formula on a disjoint set of variables such that for any assignment to k backdoor variables, the combined formula does not reduce to a formula in C and must therefore contain the empty clause in order to belong to $C^{\{\}}$.

F is constructed as follows. Using the fact that C supports large strong backdoors, construct in polynomial time a formula G over a disjoint set of variables (i.e., variables not appearing in H) such that G does not have a strong C-backdoor of size k. Now let $F = H \wedge G$. We prove that $H \in \text{UNSAT}$ iff F has a strong $C^{\{\}}$-backdoor of size k.

($\Rightarrow$) Suppose H is unsatisfiable. This implies that every truth assignment τ to V_H, the variables of H, violates some clause of H. It follows that for each such τ, $F[\tau/V_H] = H[\tau/V_H] \wedge G[\tau/V_H]$ contains the empty clause and is therefore in $C^{\{\}}$. Hence V_H gives us the desired backdoor of size k.

($\Leftarrow$) Suppose F has a strong $C^{\{\}}$-backdoor B of size k. Partition B into $B_H \cup B_G$, where B_H has the variables of H and B_G has the variables of G. By the construction of G and because $|B_G| \leq k$, B_G cannot be a strong C-backdoor for G. In other words, there exists an assignment τ_G to B_G such that $G[\tau_G/B_G] \notin C$. Because of the closure of C under removal of clauses and and the variable disjointness of H and G, it follows that $F[\tau/B] \notin C$ for every extension $\tau = (\tau_H, \tau_G)$ of τ_G to all of B. However, since B is a strong $C^{\{\}}$-backdoor for F, it must be that $F[\tau/B] \in C^{\{\}}$, and the only possibility left is that $F[\tau/B] \in C_{\{\}}$. Since $G[\tau_G/B_G] \notin C_{\{\}}$, it must be that $H[\tau_H/B_H] \in C_{\{\}}$ for all such extensions τ of τ_G. In words, this says that $H[\tau_H/B_H]$ contains a violated clause for every truth assignment to B_H. Therefore, H is unsatisfiable as desired. □

Lemmas 1 and 2 together give us our main theorem:

Theorem 1. *Let $C \in \{$Horn, 2CNF$\}$. Given a formula F and $k \geq 0$, the problem of deciding whether F has a strong $C^{\{\}}$-backdoor of size k is both NP-hard and coNP-hard, and thus harder than both NP and coNP, assuming NP $\neq$ coNP.*

We now turn our attention to the relationship between strong and deletion backdoors. While these two kinds of backdoors are known to be equivalent for the classes 2CNF and Horn, we prove an exponential separation for the class RHorn. The main idea is the following. Suppose B is a strong RHorn-backdoor for F. Then for each assignment τ to the variables in B, there exists a renaming r_τ such that $F[\tau/B]$ under the renaming r_τ yields a Horn formula. If F is such that for different τ, the various renamings r_τ are different and mutually incompatible, then there is no single renaming r under which $F - B$, the formula obtained by deleting the variables in B, becomes Horn.

The following example illustrates this point, which we will generalize to an exponential separation in the proof of Theorem 2. Let $F = (x_1 \vee x_2) \wedge (\neg y_1 \vee \neg y_2) \wedge (\neg x_1 \vee y_1 \vee z) \wedge (\neg x_1 \vee y_2 \vee \neg z) \wedge (\neg x_2 \vee y_1 \vee z) \wedge (\neg x_2 \vee y_2 \vee \neg z)$. First we observe that $B = \{z\}$ is a strong RHorn-backdoor for F because for $z = 0$ we can rename x_1 and y_1, and for $z = 1$ we can rename x_1 and y_2 to get a Horn formula. On the other hand, $\{z\}$ certainly does not work as a deletion backdoor because we must rename at least one of x_1 and x_2, which forces both y_1 and y_2 to be renamed and violates the Horn property. In fact, it can be easily verified both $\{x_1\}$ and $\{y_1\}$ are also not deletion RHorn-backdoors. From the symmetry between x_1 and x_2 and between y_1 and y_2, it follows that F does not have a deletion RHorn-backdoor of size 1.

Theorem 2. *There are formulas for which the smallest strong RHorn-backdoors are exponentially smaller than any deletion RHorn-backdoors.*

Proof. Let s be a power of 2, $t = s + \log_2 s$, and $n = s + \log_2 s + t = 2 \cdot (s + \log_2 s)$. We will prove the theorem by explicitly constructing a family of formulas $\{F_n\}$ such that F_n is defined over n variables, F_n has a strong RHorn-backdoor of size $\log_2 s = \Theta(\log n)$, and every deletion RHorn-backdoor for F_n is of size at least $s + \log_2 s - 1 = \Theta(n)$.

F_n is constructed on three kinds of variables: $\{x_i \mid 1 \leq i \leq t\}$, $\{y_j \mid 1 \leq j \leq s\}$, and $\{z_k \mid 1 \leq k \leq \log_2 s\}$. Variables z_k are used to encode all s 0-1 sequences of length $\log_2 s$. Specifically, for $1 \leq j \leq s$, let D_z^j be the unique clause involving all z variables where each z_k appears negated in D_z^j iff the k^{th} bit of j, written in the binary representation, is a 1. For example, for $j = 01101$, $D_z^j = (z_1 \vee \neg z_2 \vee \neg z_3 \vee z_4 \vee \neg z_5)$. Note that D_z^j is falsified precisely by the unique assignment that corresponds to the binary representation of j.

F_n has exactly $st+2$ clauses: $C_x = (x_1 \vee x_2 \vee \ldots \vee x_t)$, $C_y = (\neg y_1 \vee \neg y_2 \vee \ldots \vee \neg y_s)$, and for each $i \in \{1, \ldots, t\}, j \in \{1, \ldots, s\}$, the clause $C_z^{i,j} = (\neg x_i \vee y_j \vee D_z^j)$.

We now analyze RHorn-backdoors for F_n. First, we show that $\{z_k \mid 1 \leq k \leq \log_2 s\}$ is a strong RHorn-backdoor for F_n. To see this, fix any assignment $\tau \in \{0,1\}^{\log_2 s}$ to the z variables. By the discussion above, τ satisfies all but one clause D_z^j. Let us denote this falsified clause by D_z^τ. It follows that the reduced formula, $F_n[\tau/z]$, consists of C_x, C_y, and for each $i \in \{1, \ldots, t\}$, the binary clause $(\neg x_i \vee y_\tau)$. We can convert this formula to Horn by renaming or flipping the signs of all x_i, and of y_τ. This renaming makes C_x Horn. Further, it preserves the Horn property of C_y as well as of each of the t residual binary clauses. Hence the z variables form a strong RHorn-backdoor of size $\log_2 s$.

To derive a lower bound on the size of every deletion RHorn-backdoor B, notice that if B includes at least $t-1$ of the x variables, then $|B| \geq t - 1 = s + \log_2 s - 1$, as claimed. Otherwise, B does not contain at least two of the x variables, and we must therefore rename at least one of these two variables, say x_1, to make C_x Horn. This implies that we must flip all variables $y_j \notin B$ because of the clauses $C_z^{1,j}$ which now already have a positive literal, x_1. However, because of the clause C_y, we can flip at most one y variable, and it follows that at least $s-1$ of the y variables are in B. Moreover, we also have that all $\log_2 s$ of the z variables are in B, because otherwise, irrespective of how the z variables are renamed, in at least one $C_z^{1,j}$ clause a z variable will appear positively, violating the Horn property. Hence, $|B| \geq s - 1 + \log_2 s$, as claimed. This finishes the proof. □

4 Computing the Smallest Backdoors

Smallest Strong Horn-Backdoors: The problem of finding a smallest strong Horn-backdoor can be formulated as a 0-1 integer programming problem using the equivalence to deletion backdoors [19]. Given a formula F, associate with each Boolean variable x_i a 0-1 variable y_i, where $y_i = 0$ denotes that the corresponding variable x_i is deleted from F (and added to the backdoor). For a

clause $c \in F$, let $c^+ = \{i \mid x_i^1 \in c\}$ and $c^- = \{i \mid x_i^0 \in c\}$. The smallest (deletion) Horn-backdoor problem is formulated as follows:

$$\begin{array}{rll} \text{minimize} & \sum_{i \in var(F)} (1 - y_i) & \\ \text{subject to} & \sum_{i \in c^+} y_i \leq 1, & \forall c \in F \\ & y_i \in \{0, 1\}, & \forall x_i \in var(F) \end{array}$$

The constraints ensure that each clause is Horn (in each clause, the total number of not-deleted positive literals is at most one). The objective function minimizes the size of the backdoor.

Smallest Deletion RHorn-Backdoors: The problem of finding a smallest deletion RHorn-backdoor can be formulated similarly. Given a formula F, associate with each Boolean variable x_i three 0-1 variables y_{1i}, y_{2i}, y_{3i}, where $y_{1i} = 1$ denotes that x_i is not renamed in F, $y_{2i} = 1$ denotes that x_i is renamed in F, and $y_{3i} = 1$ denotes that x_i is deleted from F (and added to the deletion backdoor). The smallest deletion RHorn-backdoor problem is formulated as follows:

$$\begin{array}{rll} \text{minimize} & \sum_{i \in var(F)} y_{3i} & \\ \text{subject to} & y_{1i} + y_{2i} + y_{3i} = 1, & \forall x_i \in var(F) \\ & \sum_{i \in c^+} y_{1i} + \sum_{i \in c^-} y_{2i} \leq 1, & \forall c \in F \\ & y_{1i}, y_{2i}, y_{3i} \in \{0, 1\}, & \forall x_i \in var(F) \end{array}$$

The first set of constraints ensures that each Boolean variable x_i is either not-renamed, renamed, or deleted. The second set of constraints ensures that each clause is Horn (in each clause, the total number of not-renamed positive literals and renamed negative literals is at most one). The objective function minimizes the size of the backdoor.

We use the above encodings and the ILOG CPLEX libraries [14] for experimenting with Horn- and RHorn-backdoors.

Smallest Strong SATZ-, UP-, and (UP+PL)-Backdoors: Following previous work [25, 15], we consider strong backdoors w.r.t. branch-free search by *Satz-Rand* [11], a randomized version of *Satz* [16]. These are referred to as SATZ-backdoors. We obtain an upper bound on the size of the smallest strong SATZ-backdoors by running *Satz-Rand* without restarts multiple times with different seeds and recording the set of variables on which the solver branches when proving unsatisfiability. We also record the set of variables not set by UP and PL. This gives us an upper bound on the strong (UP+PL)-backdoor size. Similarly we record all variables set in *Satz-Rand* by anything but the UP procedure to obtain an upper bound on the smallest strong UP-backdoor size.

5 Experimental Evaluation

For our experimental evaluation, we considered four problem domains: graph coloring, logistics planning, equilibrium problems from game theory, and car configuration. The results are shown in Table 1.

instance set	num vars	num clauses	Horn %	RHorn (del) %	SATZ %	UP+PL %	UP %	$C_{\{\}}$ %
gcp_100	300	7557.7	66.67	17.00	0.30	1.20	1.23	4.00
gcp_200	600	30122.0	66.67	16.83	0.17	0.60	0.60	2.00
gcp_300	900	67724.4	66.67	16.78	0.11	0.51	0.60	1.33
gcp_400	1200	119997.4	66.67	16.75	0.08	0.38	0.55	1.00
gcp_500	1500	187556.0	66.67	16.73	0.07	0.28	0.80	0.80
map_5_7	249	720	38.96	37.75	0	2.01	2.01	
map_10_17	1284	5000	44.55	44.31	0	1.17	1.17	
map_20_37	5754	33360	47.31	47.25	0	0.61	0.61	
map_30_57	13424	103120	48.21	48.19	0	0.41	3.23	
map_40_77	24294	232280	48.66	48.65	0	0.31	3.20	
map_50_97	38364	438840	48.93	48.92	0	0.25	3.19	
pne	2000	40958.9	67.88	66.86	0.05	0.38	0.42	
pne	3000	60039.7	67.66	66.55	0.00	0.17	0.20	
pne	4000	79666.4	67.96	66.92	0.00	0.14	0.16	
pne	5000	98930.8	67.80	66.80	0.00	0.13	0.15	
C168_FW_SZ	1698	5646.8	14.32	2.83	0.16	0.77	5.70	
C168_FW_UT	1909	7489.3	23.62	5.50	0.00	0.36	1.03	
C170_FR_SZ	1659	4989.8	9.98	3.57	0.13	0.57	15.19	
C202_FS_SZ	1750	6227.8	12.31	4.55	0.13	0.61	9.42	
C202_FW_SZ	1799	8906.9	14.48	6.12	0.22	0.89	10.86	
C202_FW_UT	2038	11352.0	21.25	7.61	0.00	0.20	1.86	
C208_FA_SZ	1608	5286.2	10.52	4.51	0.06	0.40	6.50	
C208_FA_UT	1876	7335.5	23.13	7.46	0.00	0.05	0.05	
C208_FC_RZ	1654	5567.0	10.28	4.59	0.36	1.12	12.91	
C208_FC_SZ	1654	5571.8	10.47	4.68	0.16	0.63	10.41	
C210_FS_RZ	1755	5764.3	11.64	4.22	0.55	1.25	12.12	
C210_FS_SZ	1755	5796.8	11.77	4.35	0.30	0.91	15.56	
C210_FW_RZ	1789	7408.3	12.54	4.81	0.65	1.42	12.97	
C210_FW_SZ	1789	7511.8	13.74	5.37	0.23	0.78	11.15	
C210_FW_UT	2024	9720.0	20.73	7.31	0.00	0.64	4.42	
C220_FV_SZ	1728	4758.2	9.14	2.92	0.19	0.46	8.88	

Table 1. Strong backdoor sizes for various ensembles of instances: Graph Coloring (gcp), MAP planning (map), Pure Nash Equilibrium (pne), and Automotive Configuration (Cxxx). Each row reports the average over several instances. Backdoor sizes are shown as the average % of the number of problem variables. The RHorn numbers are for deletion backdoors. Horn- and RHorn-backdoor sizes are for the smallest corresponding backdoors, while the rest are upper bounds.

We generated graph coloring instances using the clique hiding graph generator of Brockington and Culberson [1]. All instances were generated with the probability of adding an edge equal to 0.5 and with a hidden clique of size 4. All SAT-encoded instances are unsatisfiable when the number of colors is 3. The twelve variables representing color assignments to the four vertices in the hidden 4-clique constitute a strong $C_{\{\}}$-backdoor, since any assignment of colors to these four vertices will fail at least one coloring constraint. This domain illustrates how the strong Horn-backdoors and deletion RHorn-backdoors can be significantly larger than backdoors w.r.t. empty clause detection; it also shows that deletion RHorn-backdoors (involving $\approx 17\%$ of the variables) are considerably smaller than strong Horn-backdoors ($\approx 67\%$). We note that *Satz* is very efficient at finding the small backdoors corresponding to the planted cliques.

The MAP problem domain is a synthetic logistics planning domain for which the size of the strong UP-backdoors is well understood [13]. In this domain, n is the number of nodes in the map graph and k is the number of locations to visit. All MAP instances considered are unsatisfiable, encoding one planning step less than the length of the optimal plan. Hoffmann et al. [13] identify that MAP instances with $k = 2n-3$ (called asymmetric) have logarithmic size DPLL refutations (and backdoors). We evaluate the size of the backdoors in asymmetric MAP instances of various sizes ($n = 5..50$). In this domain, strong Horn-backdoors and deletion RHorn-backdoors are of comparable size and relatively large (37-48%); as expected strong UP-backdoors are quite small. Interestingly, *Satz* solves these instances without any search at all, implying that the smallest strong SATZ-backdoor is of size 0.

The game theory instances encode the problem of computing an equilibrium strategy. In a game, interactions between players can be represented by an undirected graph where nodes represent players and edges represent mutual dependencies between players. Each player has a finite set of actions and a payoff function that assigns a real number to every selection of actions by him and his neighbors. Here we consider binary games, where each player has exactly two action choices. Our focus will be on random graphical games where each payoff value is chosen uniformly and independently at random from the interval $[0, 1]$ and the interaction graphs are drawn from the Erdös-Rényi random graph model $\mathbb{G}(n, p)$. In a pure Nash equilibrium (PNE), each player chooses an action and has no incentive to unilaterally deviate and change his action, given the actions chosen by the other players (i.e. each player has chosen a best response action to the choices of his neighbors). We encode the problem of deciding whether a graphical game has a PNE as a CNF formula that is satisfiable iff the given game has a PNE. For every player p, there is a Boolean variable x_p encoding the action chosen by p. For each possible action assignment for the neighbors of p, we add a clause ruling out the non-best response action of p. For this domain, while strong Horn-backdoor sets and deletion RHorn-backdoor involve $\approx 68\%$ and $\approx 67\%$ of the variables, respectively, strong SATZ-backdoors are surprisingly small, close to 0% of the variables.

Finally, we also consider a real-world SAT benchmark from product configuration. The instances encode problems from the validation and verification of automotive product configuration data for the Daimler Chrysler's Mercedes car lines [22]. We consider a set of unsatisfiable instances available at http://www-sr.informatik.uni-tuebingen.de/∼sinz/DC/. Here, while strong Horn-backdoors vary between 10-25% of the variables, RHorn-backdoor sets are considerably smaller at 3-8%. Strong SATZ-backdoors involve only 0-0.7% of the variables.

6 Conclusions

The complexity of finding backdoors is influenced significantly by the features of the underlying sub-solver or tractable problem class. In particular, while the problem of identifying a strong Horn- or 2CNF-backdoor is known to be in NP and fixed parameter tractable, strong backdoor identification w.r.t. to Horn and 2CNF becomes harder than NP (unless NP=coNP) as soon as the seemingly small feature of empty clause detection (present in all modern SAT solvers) is incorporated. While such a feature increases the worst-case complexity of finding backdoors, our experiments show that in practice it also has a clear positive impact: it reduces the size of the backdoors dramatically. For the class RHorn, we prove that deletion backdoors can be exponentially larger than strong backdoors, in contrast with the known results for 2CNF- and Horn-backdoors. Nonetheless, we show experimentally that deletion RHorn-backdoors can be substantially smaller than strong Horn-backdoors. We also demonstrate that strong backdoors w.r.t. UP, PL, and UP+PL can be substantially smaller than strong Horn-backdoors and deletion RHorn-backdoors. Finally, despite the worst-case complexity results for strong backdoor detection, we show that *Satz-Rand* is remarkably good at finding small strong backdoors on a range of problem domains.

Acknowledgments

The authors would like to thank Joerg Hoffmann for providing the generator for the MAP domain, and the reviewers for their thoughtful comments. This research was supported by IISI, Cornell University, AFOSR Grant FA9550-04-1-0151.

References

[1] M. Brockington and J. C. Culberson. Camouflaging independent sets in quasi-random graphs. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26, pages 75–88. American Mathematical Society, 1996.

[2] V. Chandru and J. N. Hooker. Detecting embedded Horn structure in propositional logic. *Information Processing Letters*, 42(2):109–111, 1992.

[3] H. Chen and V. Dalmau. Beyond hypertree width: Decomposition methods without decompositions. In *CP'05*, pages 167–181, 2005.

[4] H. Chen, C. Gomes, and B. Selman. Formal models of heavy-tailed behavior in combinatorial search. In *CP'01*, 2001.
[5] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers Inc., 2003. ISBN 1558608907.
[6] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artif. Intell.*, 34(1):1–38, 1987. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/0004-3702(87)90002-6.
[7] Y. Deville and P. Van Hentenryck. An efficient arc consistency algorithm for a class of csp problems. In *IJCAI'91*, pages 325–330, 1991.
[8] E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29(1): 24–32, 1982. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/322290.322292.
[9] E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32 (4):755–761, 1985. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/4221.4225.
[10] E. C. Freuder. Complexity of k-tree structured constraint satisfaction problems. In *AAAI'90*, pages 4–9, Boston, MA, 1990.
[11] C. Gomes, B. Selman, and H. Kautz. Boosting Combinatorial Search Through Randomization. In *AAAI'98*, pages 431–438, New Providence, RI, 1998.
[12] C. P. Gomes, B. Selman, N. Crato, and H. Kautz. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *J. Autom. Reason.*, 24(1-2): 67–100, 2000. ISSN 0168-7433.
[13] J. Hoffmann, C. Gomes, and B. Selman. Structure and problem hardness: Goal asymmetry and DPLL proofs in SAT-based planning. *Logical Methods in Computer Science*, 3(1:6), 2007.
[14] ILOG, SA. CPLEX 10.1 Reference Manual, 2006.
[15] P. Kilby, J. K. Slaney, S. Thibaux, and T. Walsh. Backbones and backdoors in satisfiability. In *AAAI'05*, pages 1368–1373, 2005.
[16] C. M. Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *IJCAI'97*, pages 366–371, 1997.
[17] I. Lynce and J. Marques-Silva. Hidden structure in unsatisfiable random 3-SAT: An empirical study. In *ICTAI'04*, 2004.
[18] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: engineering an efficient SAT solver. In *DAC'01*, pages 530–535, 2001. ISBN 1-58113-297-2.
[19] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and binary clauses. In *SAT'04*, 2004.
[20] L. Paris, R. Ostrowski, P. Siegel, and L. Sais. Computing Horn strong backdoor sets thanks to local search. *ICTAI'06*, 0:139–143, 2006. ISSN 1082-3409. doi: http://doi.ieeecomputersociety.org/10.1109/ICTAI.2006.43.
[21] M. Samer and S. Szeider. Constraint satisfaction with bounded treewidth revisited. In *CP'06*, pages 499–513, 2006.
[22] C. Sinz, A. Kaiser, and W. Küchlin. Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engr. Design, Analysis and Manufacturing*, 17(1):75–97, Jan. 2003. Special issue on configuration.
[23] S. Szeider. Backdoor sets for DLL subsolvers. *J. of Automated Reasoning*, 2005.
[24] P. van Beek and R. Dechter. On the minimality and global consistency of row-convex constraint networks. *J. ACM*, 42(3):543–561, 1995. ISSN 0004-5411. doi: http://doi.acm.org/10.1145/210346.210347.
[25] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *IJCAI'03*, pages 1173–1178, 2003.
[26] R. Williams, C. Gomes, and B. Selman. On the connections between heavy-tails, backdoors, and restarts in combinatorial search. In *SAT'03*, pages 222–230, 2003.