中山大学 Sun Yat-Sen University

SAT 2008

University of Paderborn

Theory and Applications of Satisfiability Testing

Eleventh International Conference May 12 - 15, 2008, Guangzhou, P. R. China

Computation of Renameable Horn Backdoors

Eberhard Karls Universität Tübingen

Stephan Kottler, Michael Kaufmann, Carsten Sinz

University of Tuebingen, Germany

Introduction

To decide satisfiability of real-world SAT instances it is often sufficient to focus on a particular and primarily small subset of variables - a so-called *backdoor set*. In the groundbreaking work [17] Williams, Gomes and Selman already gave examples of instances with approximately 6,700 variables and nearly 440,000 clauses that exhibit backdoor sets with only 12 variables. Ruan, Kautz and Horvitz showed empirically that an extension of the concept of backdoor sets is a good predictor for the hardness of SAT problems [13]. Moreover, Interian showed that random 3-SAT instances exhibit backdoor sets with 30% to 65% of all variables [7].

Knowing a small backdoor set for an instance in advance could speed up the solving process extraordinarily. However, according to the work of Szeider [15], it is in general not possible to decide in reasonable time whether a given SAT instance exhibits a backdoor with limited size with respect to a DPLL based subsolver (see [4, 3]).

Renameable Horn Backdoors

In this work we focus on a variant of strong backdoors, so-called *deletion backdoors* [10, 16]. A backdoor is defined with respect to a base class $\mathcal{C}$ of formulas that can be recognized and solved in polynomial time:

$\mathcal{B} \subset \mathcal{V}$ is a *deletion backdoor* if the formula $F - \mathcal{B}$ belongs to $\mathcal{C}$, where $F - \mathcal{B}$ denotes the result of removing all occurrences (both positive and negative) of the variables in $\mathcal{B}$ from the clauses of formula F. Nishimura, Ragde and Szeider proved that every deletion backdoor is a strong backdoor, if the base class $\mathcal{C}$ is *clause-induced* ($F \in \mathcal{C} \Rightarrow F' \in \mathcal{C}$ for all $F' \subseteq F$) [10].

We study the computation of backdoors where the base class $\mathcal{C}$ is Renameable Horn. A formula is Horn, if every clause contains at most one positive literal and it is Renameable Horn (RHorn) if it can be renamed to a Horn formula by flipping the literals of some variables.

Paris et al. used a two phase approach to compute RHorn backdoors as a preprocessor in a modification of the zChaff SAT solver [12]. In a recent work Dilkina, Gomes and Sabharwal formulated linear programs to compute optimal RHorn backdoors [6].

Sizes of RHorn Backdoors

The table below shows some results for instances that are mostly taken from the last SAT-Competition in 2007. One interesting result are the backdoors for the instances *eq.atree.braun.**. Both could not be solved by any solver in last years competition wheras the computation of the backdoors takes less than two seconds. Although a solving process cannot examine all 2^{755} Renameable Horn instances, this still reduces the amount of 'relevant' variables by more than 62%.

Instance	# Vars	# Cls	MaxOccurr - MinSet	MinSet - MaxOccurr	Approximation
AProVE07-04	38290	475249	4319 (11%) [37.16s]	4233 (11%) [42.44s]	4281 (11%) [8.27s]
AProVE07-06	46335	632886	4485 (9%) [28.90s]	4310 (9%) [36.72s]	4376 (9%) [12.24s]
eq.atree.braun.12	1694	5726	639 (37%) [36.59s]	665 (39%) [1.68s]	634 (37%) [1.07s]
eq.atree.braun.13	2010	6802	765 (38%) [45.17s]	785 (39%) [3.51s]	755 (37%) [1.86s]
dspam_vc1080	118298	375379	32018 (27%) [289.55m]	32034 (27%) [85.11m]	40220 (33%) [78.53m]
mizh-md5-47-3	65604	273522	15077 (22%) [25.0m]	15062 (22%) [15.22m]	16687 (25%) [1.15m]
mizh-sha0-35-4	48689	204067	11898 (24%) [12.51m]	11897 (24%) [8.13m]	13077 (26%) [44.26s]
dp10s10	8372	23004	1490 (17%) [1.45m]	1449 (17%) [26.90s]	1498 (17%) [2.08s]
vda_gr_rcs_w9	6498	130997	4461 (68%) [9.31m]	4488 (69%) [6.17m]	4293 (66%) [5.50m]
3col60_5_1	120	516	75 (62%) [0.00s]	80 (66%) [0.00s]	79 (65%) [0.00s]
avg-checker-5-35	1188	40441	362 (30%) [15.49s]	362 (30%) [1.56s]	362 (30%) [0.97s]
c5315-s	5408	15110	1291 (23%) [11.51m]	1262 (23%) [1.31m]	1300 (24%) [2.52s]
Comp-048-503	4703	45358	3371 (71%) [3.48s]	3368 (71%) [4.04s]	3346 (71%) [3.33s]
QG6-ukn2726	2123	9177	710 (33%) [15.47s]	760 (35%) [27.19s]	491 (23%) [30.18s]
bqwh.40.520-433	2211	14710	1431 (64%) [8.28s]	1442 (65%) [8.40s]	1458 (65%) [0.64s]
contest02-Mat26	744	2464	376 (50%) [0.13s]	332 (44%) [0.16s]	351 (47%) [0.08s]
gensys-ukn002	2129	8961	702 (32%) [18.26s]	770 (36%) [26.20s]	483 (22%) [29.01s]
unif-k3-r4.25	450	1912	238 (52%) [0.76s]	243 (54%) [0.60s]	243 (54%) [0.28s]
unif-k7-r89	75	6675	74 (98%) [0.15s]	74 (98%) [0.18s]	74 (98%) [0.19s]
unif2p-p0.7	3500	9344	1065 (30%) [1.23m]	1114 (31%) [1.11m]	1116 (31%) [1.4m]
unif2p-p0.8	1295	4026	488 (37%) [3.82s]	504 (38%) [4.04s]	513 (39%) [2.92s]
unif2p-p0.9	1170	4234	525 (44%) [3.87s]	557 (47%) [3.63s]	552 (47%) [3.32s]

[minimal backdoor, best time, best time and minimal backdoor]

Renameable Horn Backdoors as Graph Problem

For a given formula F we create a so-called *dependency graph* $G = (V_G, E_G)$ with $2 * |\mathcal{V}|$ vertices. Each variable v_i entails two vertices k_i^0 and k_i^1 that represent the facts that variable v_i has to be renamed (k_i^0) respectively must not be renamed (k_i^1) in order to make F a Horn formula. The directed edges of G represent the implications of renaming or not renaming variables, according to the clauses of F (see [8, 1]).

Consider for example a clause $(x_i \vee \overline{x_k} \vee \ldots)$. We know that if variable x_i is not renamed than variable x_k must also not be renamed. Moreover, if variable x_k is renamed, than variable x_i has to be renamed as well.

The following properties can be found in [1, 2, 11] or derived from these results:

Definition: We call a vertex k_i^q ($q \in \{0,1\}$) a *conflict vertex* if there is a path from k_i^q to $k_i^{(q\oplus1)}$. A variable $x_i \in \mathcal{V}$ has a *conflict loop* if k_i^0 and k_i^1 are both conflict vertices. We call the set of variables involved in a conflict loop a *conflict set*.

Lemma 1: If there is no path from k_i^q to $k_i^{(q\oplus1)}$ then none of the vertices that can be reached from k_i^q is a conflict vertex.

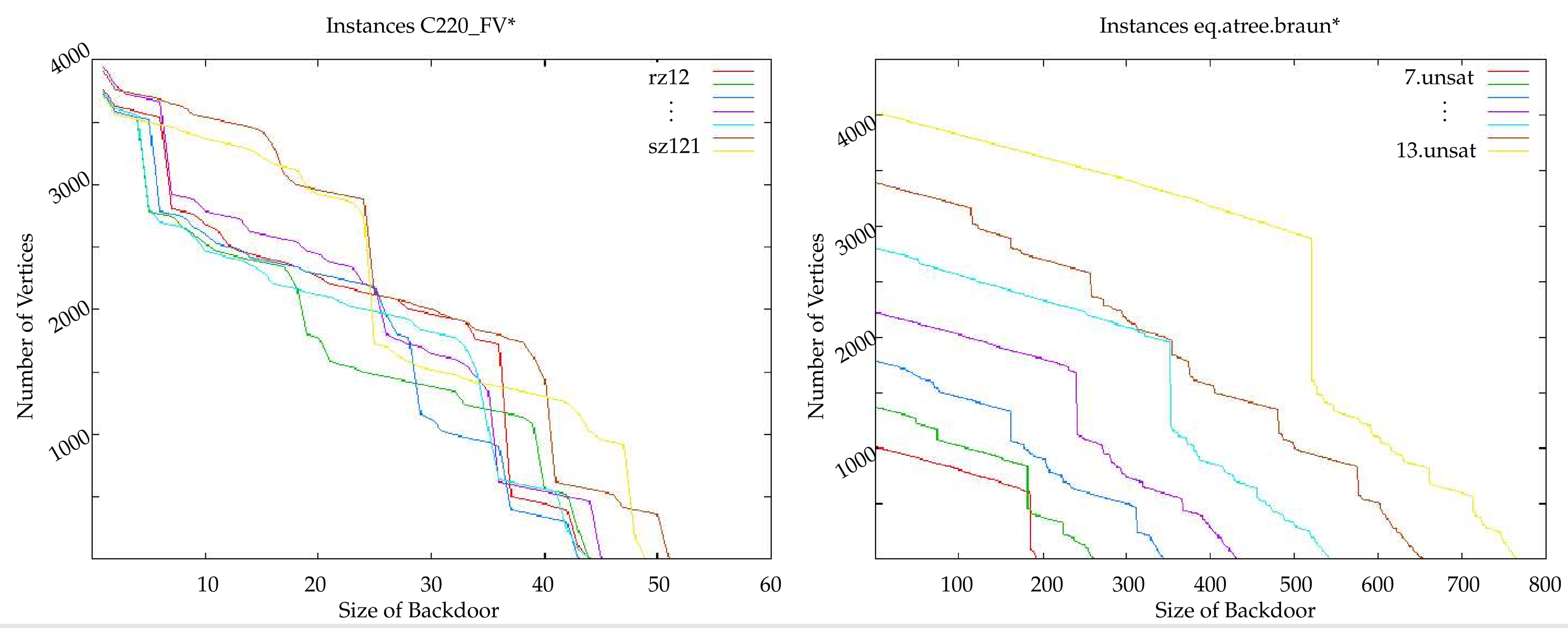
Lemma 2: A formula F is Renameable Horn iff there exists no variable that has a conflict loop in the dependency graph.

Lemma 3: If variable $x_i \in \mathcal{V}$ does not have a conflict loop than neither vertex k_i^0 nor vertex k_i^1 can be involved in a conflict loop of any other variable.

According to the second Lemma a RHorn backdoor can be computed by **destroying all conflict loops** in the dependency graph. Hence, we aim to delete a minimal amount of variables from the formula such that the deletion of the according vertices and their incident edges results in a dependency graph without any conflict loops.

Simplification of the Graph

An interesting aspect when analyzing the computation of RHorn backdoors for industrial SAT instances is the simplification of the dependency graph. For easy instances like those of the family *C220_FV** (left plot) there are several break downs where numbers of vertices can be disregarded and hence deleted according to Lemma 3. On the other hand the computation of backdoors for the very hard real-world instances of the family *eq.atree.braun** (right plot) nearly behaves like the computations for generated instances. Applying Lemma 3 is practically impossible in the first two-thirds of the computation.



Greedy Heuristics

This approach mainly considers small conflict sets and variables that occur in many of these conflict sets:

Function `greedyRHornBackdoor`(F)
- $G = (V_G, E_G) \leftarrow$ Dependency Graph of F
- $S \leftarrow$ `computeConflictSets`$(G, \mathcal{V})$
- $\mathcal{B} \leftarrow \emptyset$ (start with an empty backdoor)
- **while** $S \neq \emptyset$ **do**
  - $x_i \leftarrow$ choose variable according to heuristic
  - $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_i\}$; $\mathcal{V} \leftarrow \mathcal{V} \setminus \{x_i\}$
  - $E_G \leftarrow E_G \setminus$ {incident edges to k_i^0, k_i^1}; $V_G \leftarrow V_G \setminus \{k_i^0, k_i^1\}$
  - $\mathcal{U} \leftarrow$ variables, whose conflict loops were destroyed by deleting $\{k_i^0, k_i^1\}$
  - $S \leftarrow S \cup$ `computeConflictSets`$(G,\mathcal{U})$
  - Apply reduction rules according to Lemma 1 and Lemma 3
- **return** $\mathcal{B}$

Approximation

This 2-phase algorithm approximates an optimal RHorn backdoor with a ratio equal to the size of the biggest chosen conflict loop:

Function `approxRHornBackdoor`(F)
- $G = (V_G, E_G) \leftarrow$ Dependency Graph of F; $\mathcal{B} \leftarrow \emptyset$
- **while** *G contains at least one conflict loop* **do**
  - $C \leftarrow$ vertices of one (preferably small) conflict loop
  - $\mathcal{B} \leftarrow \mathcal{B} \cup \{var(k) : k \in C\}$
  - Hide in G all vertices related to variables in B (and incident edges)
  - Apply reduction rules according to Lemma 1 and Lemma 3
- **forall** $x \in \mathcal{B}$ **do**
  - Reinsert vertices (and edges) related to x
  - **if** *G contains no conflict loop* **then** $\mathcal{B} \leftarrow \mathcal{B} \setminus \{x\}$
  - **else** Undo reinsertion of vertices and edges related to x
- **return** $\mathcal{B}$

Optimal RHorn Backdoors

Dilkina, Gomes and Sabharwal formulated linear programs to compute optimal backdoors for different base classes [6]. The table below compares some minimum RHorn backdoors with the results of our 2-phase algorithm:

Instances	# Vars	# Cls	RHorn BD Opt [6]	RHorn BD Approx	Approx / Opt	Time
C168_FW_SZ	1698	5646.8	2.83%	3.03%	1.07	0.17s
C170_FR_SZ	1659	4989.8	3.57%	3.65%	1.02	0.07s
C202_FS_SZ	1750	6227.8	4.55%	4.79%	1.05	0.35s
C202_FW_UT	2038	11352	7.61%	8.00%	1.05	0.77s
C208_FA_SZ	1608	5286.2	4.51%	4.79%	1.06	0.06s
C208_FA_UT	1876	7335.5	7.46%	7.89%	1.06	1.50s
C208_FC_SZ	1654	5571.8	4.68%	4.75%	1.02	0.07s
C210_FS_RZ	1755	5764.3	4.22%	4.52%	1.07	0.08s
C210_FW_RZ	1789	7408.3	4.81%	4.97%	1.02	0.11s

All instances are taken from Automotive Configuration [14]. Each row reports the average of several instances.

References

[1] B. Aspvall, M. F. Plass, and R. E. Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Proc. Lett.*, 8:121–123, 1979.

[2] J. Buresh-Oppenheim and D. G. Mitchell. Minimum witnesses for unsatisfiable 2CNFs. In *SAT*, 2006.

[3] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[4] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[5] C. Demetrescu and I. Finocchi. Combinatorial algorithms for feedback problems in directed graphs. *Inf. Process. Lett.*, 86(3):129–136, 2003.

[6] B. Dilkina, C. P. Gomes, and A. Sabharwal. Tradeoffs in the complexity of backdoor detection. In *Principles and Practice of Constraint Programming - CP 2007*, 2007.

[7] Y. Interian. Backdoor sets for random 3-sat. In *SAT*, 2003.

[8] H. R. Lewis. Renaming a set of clauses as a horn set. *J. ACM*, 25:134–135, 1978.

[9] N. Nishimura, P. Ragde, and S. Szeider. Detecting backdoor sets with respect to Horn and Binary clauses. In *SAT*, 2004.

[10] N. Nishimura, P. Ragde, and S. Szeider. Solving #SAT using vertex covers. *Acta Informatica*, 44(7-8):509–523, 2007.

[11] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[12] L. Paris, R. Ostrowski, P. Siegel, and L. Sais. Computing horn strong backdoor sets thanks to local search. In *ICTAI '06*. IEEE Computer Society, 2006.

[13] Y. Ruan, H. A. Kautz, and E. Horvitz. The backdoor key: A path to understanding problem hardness. In *AAAI*, pages 124–130, 2004.

[14] C. Sinz. SAT benchmarks. www-sr.informatik.uni-tuebingen.de/~sinz/DC, 2003.

[15] S. Szeider. Backdoor sets for dll subsolvers. *J. Autom. Reasoning*, 35:73–88, 2005.

[16] S. Szeider. Matched formulas and backdoor sets. In *SAT*, 2007.

[17] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity. In *IJCAI*, 2003.