Substructure in SAT

Ryan Williams **Stanford**

Two Decades of Significant Progress in SAT Solving

Two major applications:
- Checking programs/circuits for bugs
- Finding exploits in software

(“does there exist an input which will yield the following undesired behavior?”)

Many designs can be checked completely by
- reducing the “bug finding” problem to a huge CNF-SAT instance
  (e.g., 1 million variables and 5 million clauses)
- checking UNSAT with a solver

A Huge Theory-Practice Gap

The performance of modern solvers seems to defy the theoretical claim that SAT is hard!

Practice: SAT instances that arise from a wide variety of domains are *easy*, more often than not!

- **The *unreasonable effectiveness* of the Cook-Levin Theorem**

Theory: SAT should not be easy… but it's not impossible

- **Fastest worst-case 3SAT algorithm [Hertli'11]: $O(1.308^n)$ time**
- **Exponential Time Hypothesis [IPZ'01]**
  - **3SAT requires $\Omega\left((1+\varepsilon)^n\right)$ time, for some $\varepsilon > 0$**
- **Strong Exponential Time Hypothesis [CIP'09]**
  - **For all $\varepsilon > 0$ there is a k such that kSAT needs $\Omega\left((2-\varepsilon)^n\right)$ time**

A Huge Theory-Practice Gap

The performance of modern solvers seems to defy the theoretical claim that SAT is hard!

How can we bridge the gap?

There is *tractable* substructure in real-world problems
But what structure? How do we quantify it?

Selman’s World

Bart Selman:
‘Our world may be “friendly enough” to make many typical reasoning tasks poly-time --- challenging the value of the conventional worst-case complexity view in CS.’

We can formalize what “friendly enough” means, and ask precise questions about “how friendly” tasks can be, while remaining in a “worst-case complexity” perspective

Outline

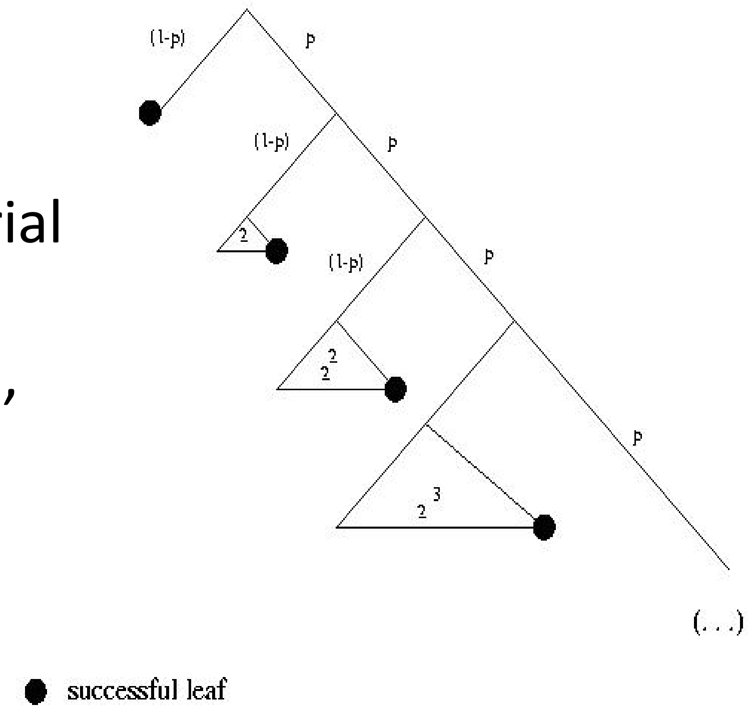
- The Origins of Backdoors
- Intuition and Properties of Backdoors
- Backdoors in Theory
- Backdoors in Practice
- Related Work
- Final Thoughts

The Origin of Backdoors

Heavy-Tailed Running Time Distributions

Many diverse instances of combinatorial search problems, when solved by randomized backtracking algorithms, yield a runtime distribution that empirically looks "heavy-tailed"

[Gomes-Selman-Crato-Kautz'00]



Pr[Running time is at least T] ~ $1/T^{\alpha}$

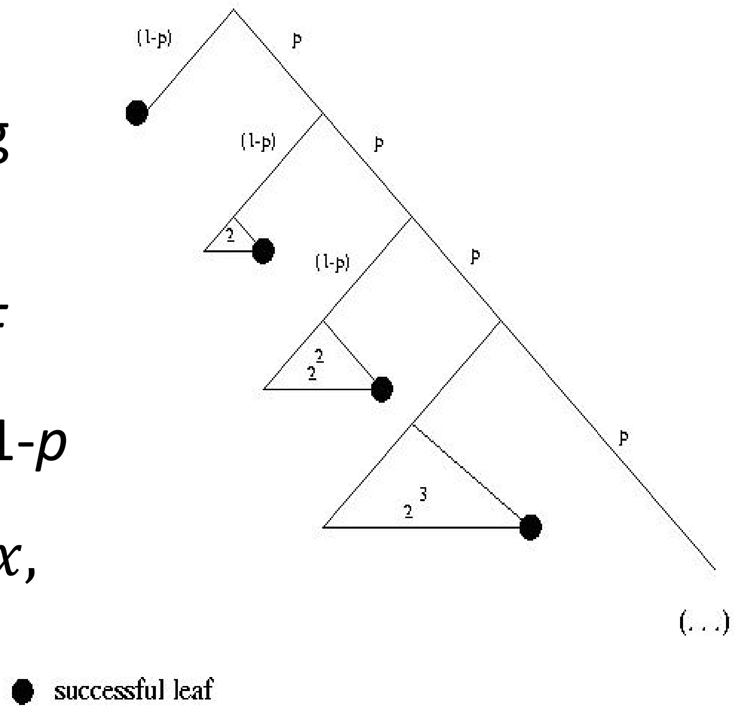
where α is a small positive constant

The Origin of Backdoors

Chen, Gomes, Selman '01

Consider a SAT instance *F* and branching solver *S* with the following properties:

1. There is one "special" variable x in *F*
2. Solver *S* chooses x with probability 1-*p*
3. If *S* chooses the variables $y_1, \ldots, y_k, x$, then *S* runs for 2^k steps



Then, Pr[(Runtime of S) $\geq 2^k$] = p^{k+1}

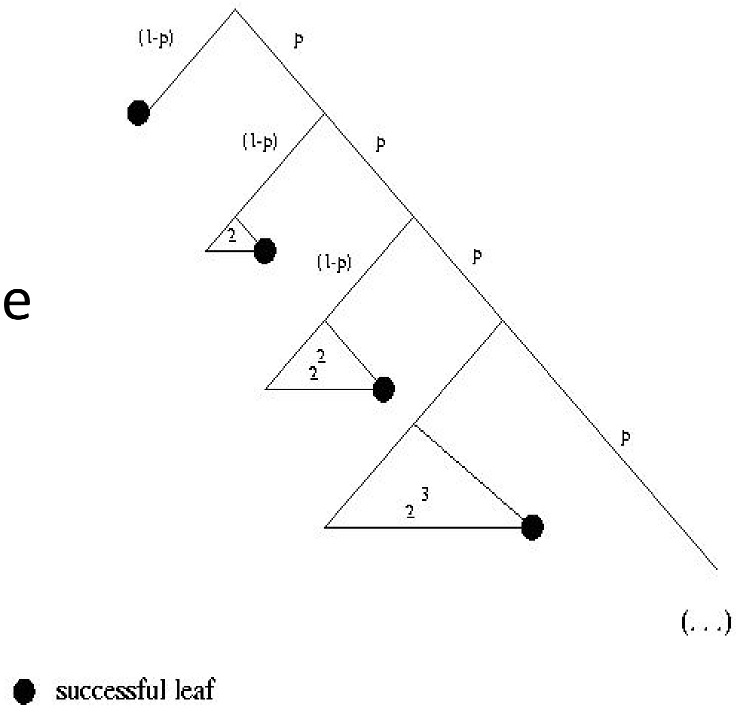
When $p \sim \frac{1}{2^\alpha}$, have heavy-tailed running time

The Origin of Backdoors

Heavy-Tailed Running Time Distributions

There did not seem to be universal agreement about whether the runtime distributions are truly heavy tailed

But there *is* universal agreement that quick restarts of a SAT solver can be remarkably effective!



How to explain short runs?

Backdoors to Tractability [WGS'03]

Informally:

- A **backdoor** to a given SAT instance is a subset of variables such that, once assigned appropriately, the remaining instance lies within a tractable subset of SAT
- The **entire set** of variables is always a backdoor set…

  The primary question is: when do *small* backdoors exist?

More formally:

- We define the notion of a "subsolver"
  (handles the tractability of problem instance)
- distinguish two types: *backdoors* and *strong backdoors*

Subsolvers (Polytime Heuristics)

Def. A **subsolver** ***A*** is an algorithm that, given any formula F, satisfies:

(Trichotomy) ***A***(F) $\in$ **{SAT, UNSAT, DK}** and never errs

(Efficiency) ***A*** on F runs in poly(|F|) time

(Triviality) On the formula F with no clauses, ***A***(F) = **SAT**
On every F with an empty clause, ***A***(F) = **UNSAT**

(Self-reducibility) If ***A***(F) $\neq$ **DK**, then for every variable x of F,
A(F[x=0]) $\neq$ **DK** and ***A***(F[x=1]) $\neq$ **DK**

Canonical example:

A*(F) = if F is 2CNF/Horn/anti-Horn then output the answer*
else output **DK**

The only non-trivial property is self-reducibility:

2CNF and Horn formulas are closed under variable substitution

Subsolvers (Polytime Heuristics)

Def. A **subsolver** ***A*** is an algorithm that, given any formula F, satisfies:

(Trichotomy) ***A***(F) ∈ **{SAT, UNSAT, DK}** and never errs

(Efficiency) ***A*** on F runs in poly(|F|) time

(Triviality) On the formula F with no clauses, ***A***(F) = **SAT**
On every F with an empty clause, ***A***(F) = **UNSAT**

(Self-reducibility) If ***A***(F) ≠ **DK**, then for every variable *x* of F,
A(F[*x*=0]) ≠ **DK** and ***A***(F[*x*=1]) ≠ **DK**

1. **Definition is general enough to encompass many polynomial time constraint propagation methods**
   (including those for which there does not exist a clean syntactic characterization of the tractable subproblem)

2. **Notion makes perfect sense for other constraint problems:**
   e.g., Mixed Integer Programming, Constraint Satisfaction

Backdoor Sets (w.r.t. Subsolvers)

Backdoors (applies to satisfiable instances):

Def. A subset S of variables of F is an ***A-backdoor for F*** if there is an assignment a_S: $S \rightarrow \{0,1\}$ such that ***A***(F[a_S]) $=$ **SAT**

Strong backdoors (applies to satisfiable and unsatisfiable instances):

Def. A subset S of variables of F is an ***A-strong backdoor for F*** if for *every* assignment a_S: $S \rightarrow \{0,1\}$ we have ***A***(F[a_S]) $\neq$ **DK**

Backdoors are an *algorithm-dependent* notion

A problem instance may have a "small" backdoor or "large" backdoor depending on which polynomial time heuristics are in the SAT solver

Observation: If P=NP then there exists a subsolver ***A*** such that every SAT formula has an ***A***-backdoor of size ***zero***

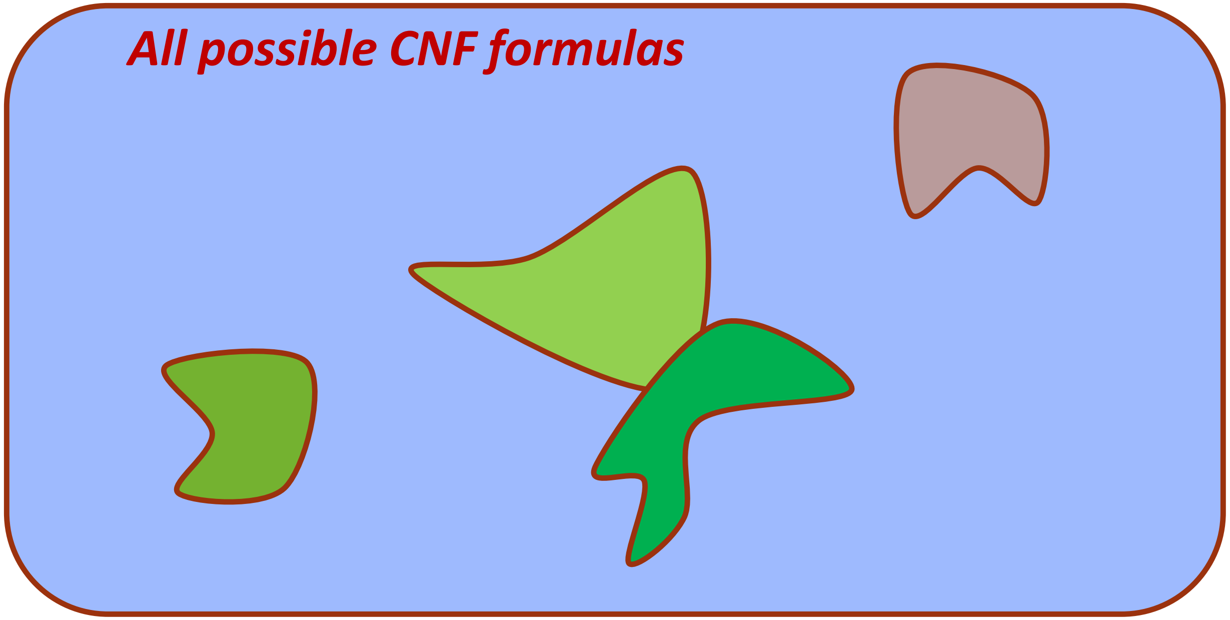
Outline

- The Origins of Backdoors
- **Intuition and Properties of Backdoors**
- Backdoors in Theory
- Backdoors in Practice
- Related Work
- Final Thoughts

Intuition for Backdoors

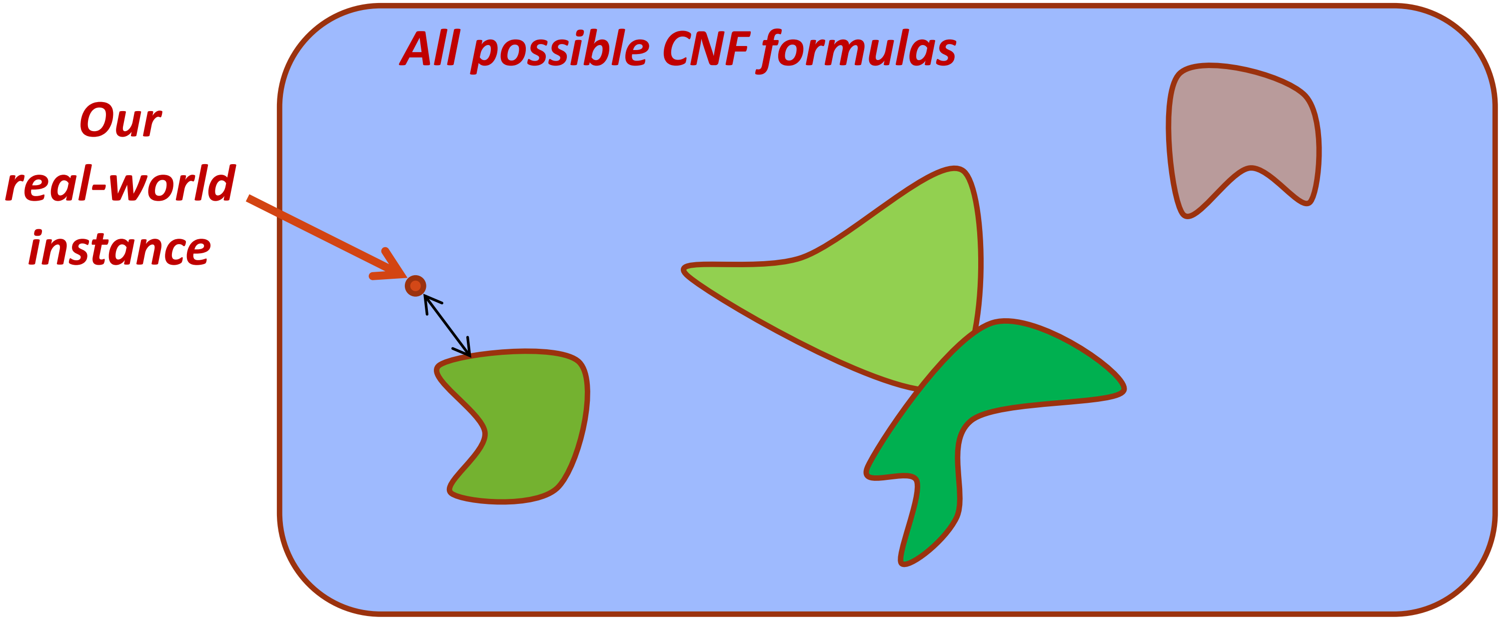
All possible CNF formulas

Intuition for Backdoors



"Islands of tractability"

Intuition for Backdoors



"Islands of tractability"

A "small" backdoor set means that the problem instance is "close" to one of these "islands of tractability"

After setting a small number of variables, we arrive at some island

The Importance of Self-Reducibility

Lemma ("Backdoors are monotone") If S is an A-backdoor for F, then for all $T \supset S$, the set T is also an A-backdoor for F

Proof: Suppose F is SAT and S is an A-backdoor for F
Then there is $a_S : S \to \{0,1\}$ such that A(F[a_S]) = SAT
That is, a_S can be extended to a SAT assignment a on all variables
Let a_T: $T \to \{0,1\}$ be the restriction of a to the set T
(i.e., for all variables x in T, $a_T(x) = a(x)$)
By self-reducibility, if A(F[a_S]) = SAT then A(F[a_T]) = SAT as well
QED

This property seems critical to the utility of backdoors in SAT solvers. One only has to assign the backdoor variables *at some point* in the branching, rather than having to necessarily choose them first

The existence of small backdoor sets is not tautological!

Just because a problem instance is solved efficiently in practice, that does not necessarily imply the instance ***must*** *have a small backdoor (w.r.t. the subsolver being used)*

For example: it could be that even the smallest backdoors are "large" but there are many of them, so finding a backdoor is like finding hay in a haystack

Proposition: A "small" backdoor of size B implies that there are at least $\binom{n-B}{k}$ backdoors of size $k+B$

Possessing small backdoors is a stronger condition than possessing many backdoors

Outline

- The Origins of Backdoors
- Intuition and Properties of Backdoors
- **Backdoors in Theory**
- Backdoors in Practice
- Related Work
- Final Thoughts

Almost all formulas *don't have* small (weak or strong) backdoors

Theorem: Let ***A*** be a subsolver handling **2-SAT or Horn-SAT**
Whp, for sufficiently large ***d (below the k-SAT threshold)*** a random k-SAT instance with ***n*** variables and ***dn*** clauses has minimum ***A*-**backdoor size at least **cn**

Intuition: With high probability, a backdoor set of variables must "hit" many clauses in order to simplify a *random* k-CNF instance enough to become Horn or 2-SAT

So for *"almost all"* instances, there is no small backdoor set with respect to these natural subsolvers.

(This could also explain why randomized backtracking performs poorly on large enough random 3-SAT instances)

The existence of small backdoors in a problem instance means that it is "far from random"

Every satisfiable k-CNF formula has a backdoor of "nontrivial" size

Theorem [Implicit in PPZ'99, "Satisfiability Coding Lemma"]

Let A be a subsolver that does unit propagation

(whenever it finds a clause of size 1, it sets the variable)

Every satisfiable k-CNF formula contains a backdoor set (wrt A) of size at most **$n(1-1/(2k))$**

Furthermore, such a backdoor can be found whp, by simply trying random variable assignments and unit propagation.

Intuition: A 1/(2k)-fraction of the variables will be assigned by unit propagation, in expectation

The rest is set to correct values with probability $\geq 2^{-n(1-1/(2k)}$

Corollary **k-SAT is solvable in $O(2^{n(1-1/(2k)})$ time**

Generic Strategies for Solving SAT with Small Backdoors

Three simple strategies for solving instances with small backdoor sets, that work for all subsolvers

- **A deterministic algorithm**
- **A randomized algorithm**
  - ***Provably better worst-case performance over the deterministic one***
- **A heuristic randomized algorithm**
  - ***Assumes existence of a good heuristic for choosing variables to branch on***
  - ***We believe this is close to what happens in practice***

Easy SAT algorithm for small backdoors

For increasing k=1,2,…

Try all k-sets S of variables and all possible Boolean assignments to S.

If the subsolver outputs SAT on some S, output "SAT"

If there is an S for which the subsolver outputs UNSAT on all assignments to S, output "UNSAT"

When there is a backdoor of size k,

takes about $O(\binom{n}{k} 2^k)$ calls to the sub-solver

Randomized algorithm

Idea: Try to backtrack on a superset of t variables that *contain* the backdoor set of size k

Assume a backdoor of size k.
A randomly chosen t-set of variables contains the backdoor, with probability at least $\binom{n-k}{t-k}/\binom{n}{t} \geq \binom{t}{k}/\binom{n}{k}$

Pick such a set and try all 2^t assignments with the subsolver.

Repeat for $2\binom{n}{k}/\binom{t}{k}$ times; takes about $2^t\binom{n}{k}/\binom{t}{k}$ calls.
When $2^k\binom{t}{k} > 2^t$ then this strategy is faster

For example, if t = 2k then $2^k\binom{2k}{k} > 7^k > 2^t$

OPEN: What is the *optimal* randomized strategy?
(Count only the number of calls to the subsolver)

Heuristic Randomized Algorithm

Assume we have:

DFS, a generic depth first search randomized backtrack search solver with:

- *(polytime)* subsolver ***A***
- Heuristic ***H*** that (randomly) chooses variables to branch on, in polynomial time
  - ***H*** has probability *1/h* of choosing a backdoor variable *(h > 1 is a fixed constant)*

Call this ensemble **(DFS, *H*, *A*)**

Heuristic Randomized Algorithm

Theorem [WGS'03]

If the minimum A-backdoor for F has size O(log n)/(log h), then (DFS, H, A) has a restart strategy that solves F in polynomial time.

If **there is a small backdoor,**

then ***(DFS, H, A)*** **has a restart strategy that runs in polynomial time.**

Outline

- The Origins of Backdoors
- Intuition and Properties of Backdoors
- Backdoors in Theory
- **Backdoors in Practice**
- Related Work
- Final Thoughts

Backdoors in Practice

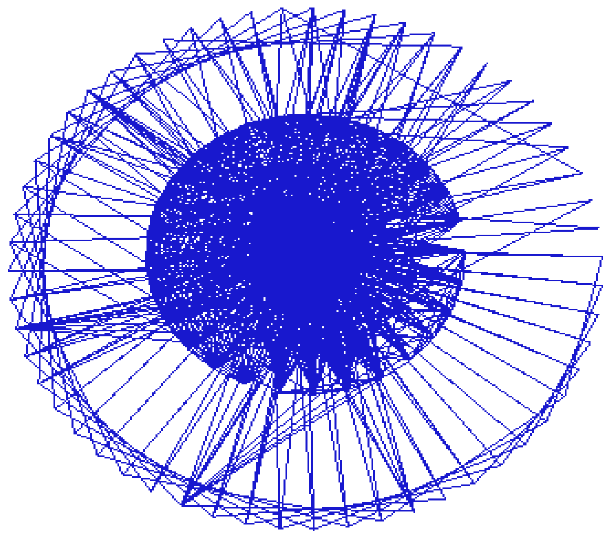
instance	# vars	# clauses	backdoor	fract.
logistics.d	6783	437431	12	0.0018
3bitadd_32	8704	32316	53	0.0061
pipe_01	7736	26087	23	0.0030
qg_30_1	1235	8523	14	0.0113
qg_35_1	1597	10658	15	0.0094

Subsolver used here: the SATz heuristics

A great deal of follow-up work since the initial experiments!

[Survey by Gomes, Kautz, Sabharwal, and Selman '07]

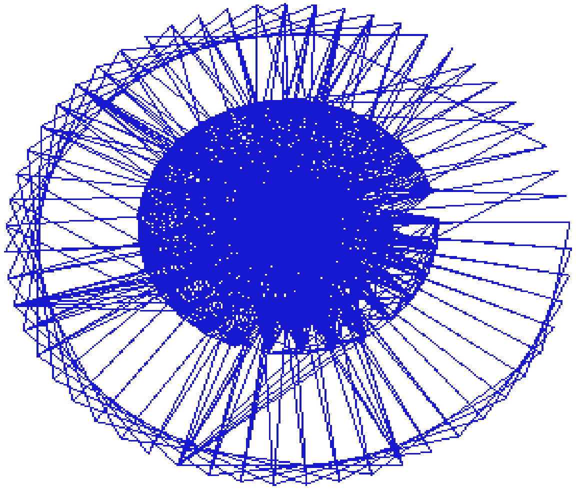
A Dynamic View (Bart's Movies)



Variable-Variable Graph of an UNSAT Instance as a SAT solver is being run on it

(random selection of variables to branch on)

A Dynamic View (Bart's Movies)



Graph when SAT solver backtracks
directly on strong backdoor of UNSAT instance
(variables chosen by heuristics of solver)

Backdoors can help explain why QBF is still hard in practice

Recall QBF = Quantified Boolean Formulas

e.g. $(\exists x)(\forall y)\ (\exists z)((x\ \ \mathrm{AND}\ \ \mathrm{NOT}(y))\ \mathrm{OR}\ z)$

With QBF, the order of the quantified variables is critical: one can't just pick any old variables to branch on

If the presence of small backdoor sets are helping SAT solvers work well, this makes sense:

In SAT, you can branch on any desired variable, so small "bottleneck" variables can be eliminated early in search

(Note: Samer and Szeider have a notion of backdoor sets for QBF)

Outline

- The Origins of Backdoors
- Intuition and Properties of Backdoors
- Backdoors in Theory
- Backdoors in Practice
- **Related Work**
- Final Thoughts

Related Work

1. Operations Research [Crama, Ekin, Hammer '97]
   Control sets: Small sets of variables for a formula that, once those variables are deleted/set to the right value, the resulting formula has some "nice property"

2. Parameterized algs [Guo, Hueffner, Niedermeier '04]
   Distance from triviality: Suppose one can make k "edits" to a problem instance so that it's then easy to solve.

   (Presumably such edits preserve the solvability.)

   Can we solve the instance in $\mathbf{O(f(k)n^c)}$?

Outline

- The Origins of Backdoors
- Intuition and Properties of Backdoors
- Backdoors in Theory
- Backdoors in Practice
- Related Work
- **Final Thoughts**

Final Thoughts

A *backdoor set* of variables tries to isolate the "difficult part" of a problem instance

Since instances in practice are often easy, this part is often small.

Many real-world instances have small backdoors w.r.t. modern SAT solver heuristics, and these solvers do exploit it

A significant question still remains…

Why are the backdoors there?

Are there deeper reasons why these small bottlenecks exist in practice, but not in random instances?

[Hemaspaandra-Williams '12]
Does the *compressibility* of practical SAT instances relate to the sizes of backdoors?

The CNFs encountered in practice are the outputs of highly regular reductions -- and the reductions are given inputs which also highly regular.

Do "compressible solutions" always arise from "compressible instances"?

Does structure imply suboptimality?

Small backdoors for hardware/software verification are typically seen as a very *positive aspect*

But their presence can also indicate *inefficiencies* in the designs of these systems.

(Indeed, SAT solvers can also be used to quickly find security exploits as well!) [Brumley, Engler]

Theory would predict that we must be missing a wide range of efficient and more secure software designs, if everything we verify in practice has such extreme structure.

[W '10,'11,'13] Improved algorithms solving circuit SAT

➔ Circuit complexity lower bounds!

Thank you!