Backbones and Backdoors in Satisfiability

Philip Kilby
ANU
Canberra, Australia
Philip.Kilby@anu.edu.au

John Slaney
NICTA and ANU
Canberra, Australia
John.Slaney@anu.edu.au

Sylvie Thiébaux
NICTA and ANU
Canberra, Australia
Sylvie.Thiebaux@anu.edu.au

Toby Walsh
NICTA and UNSW
Sydney, Australia
tw@cse.unsw.edu.au

March 2005

Overview

Study backbones and backdoors of SAT

Results:

- Backbones are hard even to approximate
- Procedures for computing backdoors
- *Little correlation between being in the backbone and in a backdoor*
- Problem hardness (for DPLL)
  - correlated with the size of strong backdoors?
  - weakly correlated with the size of the backbone?
  - not correlated to size of weak backdoors nor their number.

Definitions

Backbone The set of literals that appear in all solutions

Weak Backdoor A set of literals such that when set, the remainder of the problem can be solved in polynomial time by some algorithm

- E.g. After fixing backdoor literals, *satz* finds a solution without backtrack

Strong Backdoor A set of variables such that, regardless of how they are set, the remaining problem can be decided in polynomial time by some algorithm.

- E.g. Given any instantiation of strong backdoor variables, *satz* decides satisfiability without backtrack

Backbones are hard to approximate

Theorem 1:
If P $\neq$ NP then no sound approximation procedure can return a fixed fraction α or greater of the SAT backbone in polynomial time.

Proof
Suppose there was such a procedure. Since it returns a fixed fraction of the backbone (rounded up) it must return at least one backbone literal if the backbone is non-empty. Set this literal to true and simplify the formula. If the backbone is empty, set an arbitrary literal to true and simplify. Call the approximation procedure and repeat. This finds a satisfying assignment if one exists in polynomial time, contradicting the assumption that P $\neq$ NP.

Corollary to Thm 1:
No sound approximation procedure can return at least one backbone literal.

Backbones are hard to approximate II

Theorem 2:
If P $\neq$ NP then no majority-approximation procedure can be guaranteed to return a fixed fraction α or greater of the literals in the backbone in polynomial time.

Proof

- Σ is the set of clauses
- $k =_{\text{def}} \lceil \frac{1}{\alpha} \rceil$.
- Construct k copies of Σ:
  - Add new var x_i as disjunct in each clause - i.e.
  - $Sigma_i = \{c \vee x_i \mid c \in \Sigma\}$
  - If Σ is unsatisfiable, we can always satisfy Σ_i by setting x_i to true.

Backbones are hard to approximate III

- Consider clauses: $\{z\} \cup (\bigcup_{1 \leq i \leq k} \Sigma_i)$
  - z is again a new variable
  - For fixed α this set is polynomial in the size of Σ
  - If Σ is satisfiable, then z is the unique backbone literal
  - If Σ is unsatisfiable, then the backbone is $\{x_1, \ldots, x_k, z\}$
- Use the approximation procedure to find the backbone of the constructed formula.
- If Σ is satisfiable, the majority approximation procedure must return just z
- If the formula Σ is unsatisfiable then the majority approximation procedure must return at least one literal, x_j from the backbone.
- *Hence, returned literals can be used to decide the satisfiability of* Σ

Algorithm 1: MinWeakBackdoor (F, I)

Input: Formula F, Initial weak backdoor set I - i.e. running *satz* on $F \cup I$ requires no branching.

Output: A set of literals W forming a minimal backdoor, and a model M consistent with the backdoor

1. $W \leftarrow \emptyset$; $M \leftarrow \emptyset$
2. **while** $I \neq \emptyset$
3.   Choose literal $l \in I$ randomly
4. $I \leftarrow I \setminus \{l\}$
5.   Run *satz* on $F \cup W \cup I$
6.   **if** *satz* requires branching,
7.   **then** $W \leftarrow W \cup \{l\}$; $M \leftarrow$ *satz* solution
8. **endwhile**
9. **return** W, M

Algorithm 2: SatzWeak (F)

Input: Formula F

Output: A minimal weak backdoor W and a consistent model M

1. Solve F using *satz* , saving branching literals in B
2. $W, M \leftarrow \textsc{MinWeakBackdoor}\ (F, B)$
3. **return** W, M

Algorithm 3: SatzLS1 (F, W, M)

Input: Formula F, Initial backdoor W, Model M

Output: A set of minimal weak backdoors $\mathcal{S}$

1. $\mathcal{S} \leftarrow \emptyset$; $B \leftarrow W$
   \# W is current backdoor,
   \# B is an example of smallest backdoor seen.
2. Restart-count $\leftarrow 0$
3. **while** Restart-count $<$ Restart-limit
4.   Restart-count $\leftarrow$ Restart-count $+ 1$
5. $W \leftarrow B$
6.   Iteration-count $\leftarrow 0$;
7.   **while** Iteration-count $<$ Iteration-limit
8.   Iteration-count $\leftarrow$ Iteration-count $+ 1$
9. $Z \leftarrow \mid W \mid \times$ Card-mult literals chosen
               randomly from $M \setminus W$
10. $W \leftarrow \textsc{MinWeakBackdoor}\ (F, W \cup Z)$
              (sequential version)
11. $\mathcal{S} \leftarrow \mathcal{S} \cup W$
12.   **if** $\mid W \mid < \mid B \mid$
13.   **then** $B \leftarrow W$; Restart-count $\leftarrow 0$
14.   **endwhile**
15. **endwhile**
16. **return** $\mathcal{S}$

Algorithm 4: SatzLS2(F, L)

Input: Formula F, and a list of models L.

Output: A set of minimal weak backdoors $\mathcal{S}$

$\mathcal{S} \leftarrow$ $\textsc{SatzLS1}$ (F, $\textsc{SatzWeak}(F)$).
while $L \neq \emptyset$
    Choose M from L
    $L \leftarrow L \setminus M$
    $W \leftarrow \emptyset$
    do
        Choose l randomly from $M \setminus W$
        $W \leftarrow W \cup \{l\}$
        Solve $F \cup W$ using *satz*
    until no branching required
    $W \leftarrow$ $\textsc{MinWeakBackdoor}$ (F, W)
    $\mathcal{S} \leftarrow \mathcal{S} \cup$ $\textsc{SatzLS1}$ (F, W, M)
endwhile
return $\mathcal{S}$

Algorithm 5: StrongBackdoor (F, Max-card)

Input: Formula F, the maximum cardinality Max-card.

Output: A set of strong backdoors $\mathcal{S}$, and a set of weak backdoors $\mathcal{W}$.

1. $\mathcal{S} \leftarrow \mathcal{W} \leftarrow \emptyset$
2. **for each** subset X of the problem variables of size
   up to Max-card
3.   **for each** distinct set of literals L corresponding
         to the variables in X
4.   Run *satz* on $F \cup L$
5.   **if** branching is not required,
               and the formula is satisfiable
6.   **then** $\mathcal{W} \leftarrow \mathcal{W} \cup L$
7.   **if** no literal set required branching
8.   **then** $\mathcal{S} \leftarrow \mathcal{S} \cup X$
9. **return** $\mathcal{S}, \mathcal{W}$

THE AUSTRALIAN NATIONAL UNIVERSITY

NATIONAL ICT AUSTRALIA LIMITED

Tests - data

Abbrev	Description	n	m	Number of Inst.
RTI	Random 3-sat	100	429	500
uf20	Random 3-sat	20	91	1000
uf50	Random 3-sat	50	218	1000
uuf50	Random 3-sat (unsatisfiable)	50	218	1000
uf100	Random 3-sat	100	430	1000
uf125	Random 3-sat	125	538	100
uf150	Random 3-sat	150	645	100
uf175	Random 3-sat	175	753	100
uf200	Random 3-sat	200	860	100
uf225	Random 3-sat	225	960	100
flat30	SAT-encoded graph colouring	90	300	100
flat50	SAT-encoded graph colouring	150	545	100

Used *satz* *branch-free* backdoors – includes unit resolution and 2-SAT as polynomial classes.

Backdoors

Size	Num weak backdoors	Num strong backdoors
0	275	275
1	687	238
2	38	171
3	0	102
>3	0	214

Size of strong and weak backdoors - Uniform random 3-SAT with 20 variables and 91 constraints (i.e. at the phase transition)

THE AUSTRALIAN NATIONAL UNIVERSITY

NATIONAL ICT AUSTRALIA LIMITED

Backbones and Backdoors

- Each formula F can have a large number of backdoors, both strong and weak, of various sizes.
- $w_F =_{\text{def}}$ cardinality of the smallest weak backdoor of F
- $s_F =_{\text{def}}$ cardinality of the smallest strong backdoor of F
- $W^*_F =_{\text{def}}$ set of literals that are in at least one weak backdoor of size w_F
- $S^*_F =_{\text{def}}$ set of variables that are in at least one strong backdoor of size s_F

- for $n = 20$ (1000 instances)
  - mean $w_F = 0.76$
  - mean $s_F = 1.1$

THE AUSTRALIAN NATIONAL UNIVERSITY

NATIONAL ICT AUSTRALIA LIMITED

Backbone *versus* backdoor

In 1000 problems of size $n = 20$, **no** backbone variables appeared in any strong or weak backdoor

	Problem set	Mean backbone size	Mean w_F	Mean $\mid W_F^* \mid$	Mean Overlap
Weak	uf20[1]	13.7	0.76	4.4	0
Backdoors	uf50[1]	30.8	1.6	18.8	6.5
	uf100[2]	53.6	4.6	10.2	2.7
	RTI[2]	53.8	4.5	9.8	2.5

	Problem set	Mean backbone size	Mean s_F	Mean $\mid S_F^* \mid$	Mean Overlap
Strong Backdoors	uf20[1,3]	15.5	1.1	3.1	0
	uf50[1,3]	43.4	2.0	16.4	13.0

[1] These entries are based on systematic search

[2] These entries are based on local search

[3] Only strong backdoors of size < 4 were considered

Problem Hardness

Problem set	Statistic	r-value	c.o.d.
uf20	s_F	**0.71**	0.50
uf20	s_F + guess	**0.78**	0.61
uf20	w_F	0.58	0.34
uf20	Backbone size	-**0.88**	0.78
uuf50	s_F	**0.74**	0.54
uf50	s_F	0.37	0.14
uf50	s_F + guess	0.46	0.21
uf50	w_F	0.26	0.07
uf50	Backbone size	-**0.46**	0.21
RTI	w_F	0.03	0.00
uf100	w_F	-0.01	0.00
uf125	w_F	-0.03	0.00
uf150	w_F	0.07	0.01
uf175	w_F	0.01	0.00
uf200	w_F	-0.13	0.02
uf225	w_F	-0.05	0.00

THE AUSTRALIAN NATIONAL UNIVERSITY

NATIONAL ICT AUSTRALIA LIMITED

Problem Hardness II

Problem set	Statistic	r-value	c.o.d.
RTI	Backbone size	0.16	0.03
uf100	Backbone size	0.25	0.06
uf125	Backbone size	0.32	0.10
uf150	Backbone size	0.28	0.08
uf175	Backbone size	0.25	0.06
uf200	Backbone size	0.34	0.12
uf225	Backbone size	0.48	0.23
flat30	Num weak backdoors overall	-0.31	0.09
flat50	Num weak backdoors overall	-0.37	0.14
flat30	Num models	0.42	0.18
flat50	Num models	0.54	0.29

THE AUSTRALIAN NATIONAL UNIVERSITY

NATIONAL ICT AUSTRALIA LIMITED

Conclusions

- Most interesting: Little overlap between (weak) backdoor and backbone
- Implication: A *backdoor*-guided search will choose different variables to a *backbone*-guided search
- (Others have shown that backbone-guided search can be successful)
- Open question: Will a backdoor-guided search perform better than a backbone-guided search?
- Open question: How to guide an algorithm toward a backdoor?
- At least: There are lots of distinct backdoors out there, with many literals involved (but only one backbone)