Backdoors to Satisfaction: Parameterized Complexity

Serge Gaspers

The University of New South Wales, Sydney, Australia

National ICT Australia, Sydney, Australia

First Symposium on Structure in Hard Combinatorial Problems
16-May-2013
Vienna, Austria

Outline

SAT and #SAT

SAT

Input:	A propositional formula F in conjunctive normal form (CNF)
Question:	Is there an assignment to $\mathrm{var}(F)$ satisfying all clauses of F?

#SAT

Input:	A CNF formula F
Question:	What is the number of assignments to $\mathrm{var}(F)$ satisfying all clauses of F?

Example:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

SAT: theory vs. practice

theory

- NP-complete
- *ETH*: SAT cannot be solved in time $2^{o(n)}$
- *Strong ETH*: SAT cannot be solved in time $(2-\epsilon)^n$ for any $\epsilon > 0$

practice

- Want to solve an NP-complete problem? Just encode into SAT and use a SAT solver
- *Real-world* instances with millions of variables and clauses

Backdoors

- Belief: real world instances have a "hidden structure" that makes them easy to solve
- Challenge: measure and identify this hidden structure
- One way: *Backdoor* = set of "key" variables that make it easy to solve the formula

Backdoors

- CNF formula F
- Set of variables $B \subseteq \text{var}(F)$
- For a truth assignment τ to B, the reduced formula $F[\tau]$ is obtained from F by removing all clauses satisfied by τ and removing all remaining literals on B from the other clauses
- Base class $\mathcal{C}$: class of poly-time solvable CNF formulas

Definition (Weak Backdoor [Williams, Gomes, Selman, 2003])

B is a weak $\mathcal{C}$-backdoor for F if there is a truth assignment τ to B such that $F[\tau] \in \mathcal{C}$ and $F[\tau]$ is satisfiable.

Definition (Strong Backdoor [Williams, Gomes, Selman, 2003])

B is a strong $\mathcal{C}$-backdoor for F if for every truth assignment τ to B we have $F[\tau] \in \mathcal{C}$.

Backdoors

- CNF formula F
- Set of variables $B \subseteq \text{var}(F)$
- For a truth assignment τ to B, the reduced formula $F[\tau]$ is obtained from F by removing all clauses satisfied by τ and removing all remaining literals on B from the other clauses
- Base class $\mathcal{C}$: class of poly-time solvable CNF formulas

Definition (Weak Backdoor [Williams, Gomes, Selman, 2003])

B is a weak $\mathcal{C}$-backdoor for F if there is a truth assignment τ to B such that $F[\tau] \in \mathcal{C}$ and $F[\tau]$ is satisfiable.

Definition (Strong Backdoor [Williams, Gomes, Selman, 2003])

B is a strong $\mathcal{C}$-backdoor for F if for every truth assignment τ to B we have $F[\tau] \in \mathcal{C}$.

Backdoors

- CNF formula F
- Set of variables $B \subseteq \text{var}(F)$
- For a truth assignment τ to B, the reduced formula $F[\tau]$ is obtained from F by removing all clauses satisfied by τ and removing all remaining literals on B from the other clauses
- Base class $\mathcal{C}$: class of poly-time solvable CNF formulas

Definition (Weak Backdoor [Williams, Gomes, Selman, 2003])

B is a weak $\mathcal{C}$-backdoor for F if there is a truth assignment τ to B such that $F[\tau] \in \mathcal{C}$ and $F[\tau]$ is satisfiable.

Definition (Strong Backdoor [Williams, Gomes, Selman, 2003])

B is a strong $\mathcal{C}$-backdoor for F if for every truth assignment τ to B we have $F[\tau] \in \mathcal{C}$.

Experimental results

Table 4. Size, percentage, and number of small backdoors found by the local search algorithms within a cutoff of 3 hours when applied to real-world instances with n variables ($n < 10,000$) and m clauses.

Instance	n	m	Kilby		KilbyImp		Tabu	
			BD size (%)	# BDs	BD size (%)	# BDs	BD size (%)	# BDs
SAT Competition 2002								
apex7_gr_rcs_w5.shuffled	1500	11136	77 (5.13%)	1	**47 (3.13%)**	**4**	53 (3.53%)	42885
dp10s10.shuffled	8372	8557	9 (0.11%)	10520	9 (0.11%)	9573	**9 (0.11%)**	**59399**
bart11.shuffled	162	675	15 (9.26%)	4190	14 (8.64%)	2903	**14 (8.64%)**	**45044**
SAT-Race 2005 and 2008								
grieu-vmpc-s05-24s	576	49478	3 (0.52%)	143	3 (0.52%)	143	3 (0.52%)	143
grieu-vmpc-s05-27r	729	71380	4 (0.55%)	710	4 (0.55%)	660	**4 (0.55%)**	**3271**
simon-mixed-s02bis-01	2424	13793	8 (0.33%)	566	8 (0.33%)	566	**8 (0.33%)**	**10440**
simon-s02b-r4b1k1.2	2424	13811	8 (0.33%)	394	7 (0.29%)	3	**7 (0.29%)**	**16**
Blocks world planning								
bw_large.c	3016	50237	4 (0.13%)	1934	**3 (0.10%)**	**15**	**3 (0.10%)**	**15**
bw_large.d	6325	131607	6 (0.10%)	790	**5 (0.08%)**	**69**	6 (0.10%)	640
Logistics planning								
logistics.a	828	3116	20 (2.42%)	147	**20 (2.42%)**	**6675**	24 (2.90%)	584257
logistics.b	843	3480	16 (1.90%)	1688	**15 (1.78%)**	**9789**	16 (1.90%)	7634
logistics.c	1141	5867	26 (2.28%)	18	**25 (2.19%)**	**387**	28 (2.45%)	424467
logistics.d	4713	16588	25 (0.53%)	39	**22 (0.47%)**	**61**	28 (0.59%)	36610

[Li, van Beek, 2011] weak backdoors to UP+2CNF+1-Val+0-Val

Backdoor Problems

Weak (Strong) $\mathcal{C}$-Backdoor Detection

Input:	A CNF formula F, an integer k
Question:	Does F have a weak (strong) $\mathcal{C}$-backdoor of size at most k?

Weak (Strong) $\mathcal{C}$-Backdoor Evaluation

Input:	A CNF formula F, a weak (strong) $\mathcal{C}$-backdoor B
Question:	Is F satisfiable?

Outline

Parameterized Complexity

"complexity is not governed by the instance size alone"

Definition (Parameterized problem)

A parameterized decision problem is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, x is the main part and k the parameter.

FPT: class of param. pbs that can be solved in time $f(k) \cdot n^{O(1)}$
W[$\cdot$]: parameterized intractability classes
XP: class of param. pbs that can be solved in time $f(k) \cdot n^{g(k)}$

$$\text{FPT} \subseteq \text{W}[1] \subseteq \text{W}[2] \subseteq \ldots \text{XP}.$$

All inclusions believed to be strict.

Parameterized Backdoor Problems

Weak (Strong) $\mathcal{C}$-Backdoor Detection

Input:	A CNF formula F, an integer k
Parameter:	k
Question:	Does F have a weak (strong) $\mathcal{C}$-backdoor of size at most k?

Weak (Strong) $\mathcal{C}$-Backdoor Evaluation

Input:	A CNF formula F, a weak (strong) $\mathcal{C}$-backdoor B
Parameter:	$k = \|B\|$
Question:	Is F satisfiable?

Weak Backdoor Detection

Simple Weak $\mathcal{C}$-Backdoor Detection Algorithm
Input: A CNF formula F and an integer k.
Output: YES if F has a weak $\mathcal{C}$-backdoor of size k, and NO otherwise.

foreach *subset* $B \subseteq \text{var}(F)$ *with* $|B| = k$ **do**
  foreach *assignment* $\tau : B \to \{0, 1\}$ **do**
    if $F[\tau] \in \mathcal{C}$ **then**
      if $F[\tau]$ *is satisfiable* **then**
        return YES

return NO

- run time: $\binom{n}{k} \cdot 2^k \cdot n^{O(1)} = n^{k+O(1)}$
- XP-algorithm

Weak Backdoor Detection

Simple Weak $\mathcal{C}$-Backdoor Detection Algorithm
Input: A CNF formula F and an integer k.
Output: YES if F has a weak $\mathcal{C}$-backdoor of size k, and NO otherwise.

foreach *subset* $B \subseteq \text{var}(F)$ *with* $|B| = k$ **do**
  foreach *assignment* $\tau : B \to \{0, 1\}$ **do**
    if $F[\tau] \in \mathcal{C}$ **then**
      if $F[\tau]$ *is satisfiable* **then**
        return YES

return NO

- run time: $\binom{n}{k} \cdot 2^k \cdot n^{O(1)} = n^{k+O(1)}$
- XP-algorithm

Strong Backdoor Detection

Simple Strong $\mathcal{C}$-Backdoor Detection Algorithm
Input: A CNF formula F and an integer k.
Output: YES if F has a strong $\mathcal{C}$-backdoor of size k, and NO otherwise.

```
foreach subset B ⊆ var(F) with |B| = k do
    valid ← true
    foreach assignment τ : B → {0,1} do
        if F[τ] ∉ C then
            valid ← false
    if valid then
        return YES
return NO
```

- run time: $\binom{n}{k} \cdot 2^k \cdot n^{O(1)} = n^{k+O(1)}$
- XP-algorithm

Strong Backdoor Detection

Simple Strong $\mathcal{C}$-Backdoor Detection Algorithm
Input: A CNF formula F and an integer k.
Output: YES if F has a strong $\mathcal{C}$-backdoor of size k, and NO otherwise.

foreach *subset* $B \subseteq \text{var}(F)$ *with* $|B| = k$ **do**
  valid $\leftarrow$ true
  foreach *assignment* $\tau : B \rightarrow \{0, 1\}$ **do**
    if $F[\tau] \notin \mathcal{C}$ **then**
      valid $\leftarrow$ false
  if valid **then**
    return YES
return NO

- run time: $\binom{n}{k} \cdot 2^k \cdot n^{O(1)} = n^{k+O(1)}$
- XP-algorithm

Backdoor Evaluation

Simple $\mathcal{C}$-Backdoor Evaluation Algorithm
Input: A CNF formula F and a weak or strong $\mathcal{C}$-backdoor B of size k.
Output: YES if F is satisfiable, and
NO otherwise.

foreach *assignment* $\tau : B \to \{0, 1\}$ **do**
    if $F[\tau] \in \mathcal{C}$ **then** `/* not necessary for strong */`
        if $F[\tau]$ *is satisfiable* **then**
            return YES

return NO `/* not possible for weak */`

- run time: $2^k \cdot n^{O(1)}$
- FPT-algorithm

Backdoor Evaluation

Simple $\mathcal{C}$-Backdoor Evaluation Algorithm
Input: A CNF formula F and a weak or strong $\mathcal{C}$-backdoor B of size k.
Output: YES if F is satisfiable, and
NO otherwise.

foreach *assignment* $\tau : B \to \{0,1\}$ **do**
    if $F[\tau] \in \mathcal{C}$ **then**   /* not necessary for strong */
        if $F[\tau]$ *is satisfiable* **then**
            return YES

return NO   /* not possible for weak */

- run time: $2^k \cdot n^{O(1)}$
- FPT-algorithm

Consequences for SAT

- The challenging part is Backdoor Detection.
- If Weak (Strong) $\mathcal{C}$-Backdoor Detection is FPT, then SAT is FPT parameterized by the size of a smallest weak (strong) $\mathcal{C}$-backdoor.

Outline

Detecting backdoors to some base classes

	Weak		*Strong*	
Base Class	CNF	r-CNF	CNF	r-CNF
HORN	W[2]-h	FPT	FPT	FPT
2CNF	W[2]-h	FPT	FPT	FPT
UP	W[P]-c	W[P]-c	W[P]-c	W[P]-c
RHORN	W[2]-h	W[2]-h	W[2]-h	open
CLU	W[2]-h	FPT	W[2]-h	FPT

The parameterized complexity of finding weak and strong backdoor sets of CNF formulas and r-CNF formulas, where $r \geq 3$ is a fixed integer.

Results by: [Nishimura, Ragde, Szeider, 2004] [Szeider, 2005] [Nishimura, Ragde, Szeider, 2007] [Gaspers, Szeider, 2012]
See [Gaspers, Szeider, 2012] for a survey.

What does this tell us?

- FPT cases
  - There is an algorithm with running time $f(k) \cdot n^{O(1)}$ that either finds a backdoor of size k, or determines that no such backdoor exists
  - If the instance has a small backdoor, there is at least one efficient way to find it (maybe many efficient ways)
- W[$\cdot$]-hard cases
  - There is probably no algorithm with running time $f(k) \cdot n^{O(1)}$ that either finds a backdoor of size k, or determines that no such backdoor exists
  - There are instances with small backdoors of size k, but probably no efficient way to find these backdoors
  - Maybe a backdoor of size $k+1$ can still be found efficiently $\ldots$ or one of size 2^k?

FPT Approximation

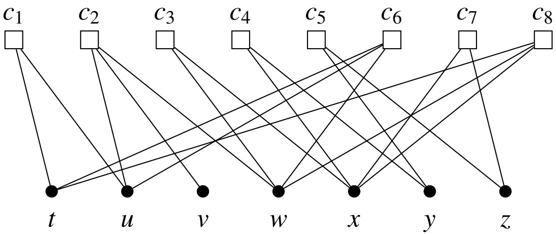
Definition ([Downey, Fellows, McCartin, 2006])

A parameterized algorithm is an FPT-approximation algorithm for a minimization problem if there exist functions f, g such that on input (x, k), the algorithm has running time $f(k) \cdot n^{O(1)}$ and it either

- determines that (x, k) is a NO-instance, or
- determines that (x, k') is a YES-instance for some $k' \leq g(k)$

Outline

Incidence graph



Incidence graph of the formula $F = \bigwedge_{i=1}^{8} c_i$ with

$$c_1 = t \vee \neg u, \quad c_2 = u \vee v \vee w, \quad c_3 = w \vee x, \quad c_4 = x \vee \neg y,$$
$$c_5 = y \vee \neg z, \quad c_6 = t \vee u \vee \neg w, \quad c_7 = \neg x \vee z, \quad c_8 = \neg t \vee w \vee x$$

Acyclic SAT formulas

Definition

A SAT formula is acyclic if its incidence graph has no cycle.

Definition

FOREST denotes the class of all acyclic SAT formulas

Results for FOREST-backdoors

Theorem ([Gaspers, Szeider, ICALP 2012])

Weak FOREST*-Backdoor Detection is* W[2]*-hard.*

Theorem ([Gaspers, Szeider, ICALP 2012])

For every constant $r \geq 3$, Weak FOREST*-Backdoor Detection is* FPT *for r-CNF formulas.*

Theorem ([Gaspers, Szeider, ICALP 2012])

There is an FPT*-approximation algorithm for Strong* FOREST*-Backdoor Detection.*

Consequences for SAT

Corollary ([Gaspers, Szeider, ICALP 2012])

r-SAT and r-#SAT are FPT parameterized by the size of a smallest weak FOREST*-backdoor.*

Corollary ([Gaspers, Szeider, ICALP 2012])

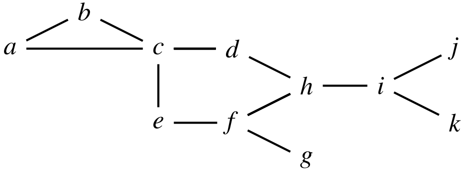
SAT and #SAT are FPT parameterized by the size of a smallest strong FOREST*-backdoor.*

More general?

- Are there larger base classes with an FPT-approximation for Strong Backdoor Detection?

Tree decompositions (by example)

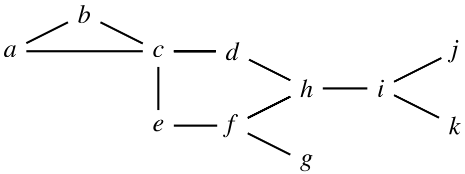
- A graph G



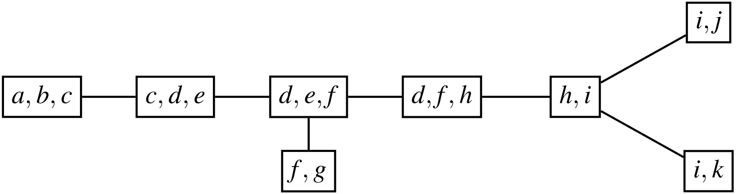
- A *tree decomposition* of G

Tree decompositions (by example)

- A graph G

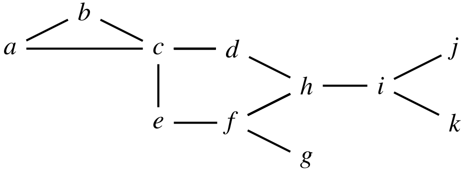


- A *tree decomposition* of G

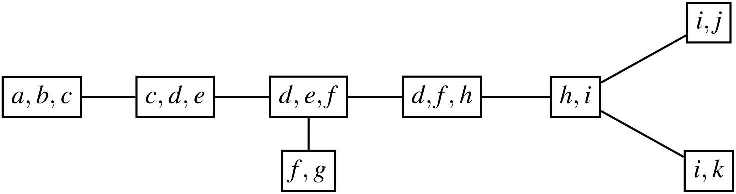


Tree decompositions (by example)

- A graph G



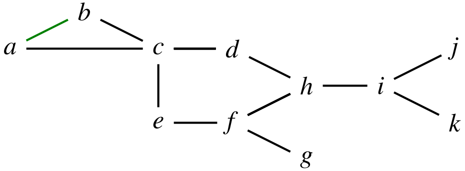
- A *tree decomposition* of G



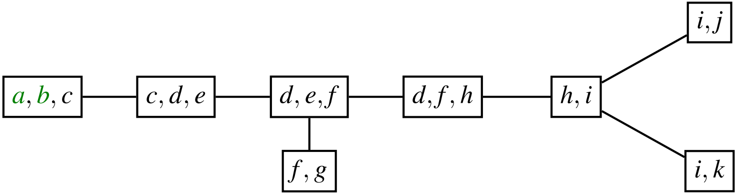
Conditions:

Tree decompositions (by example)

- A graph G



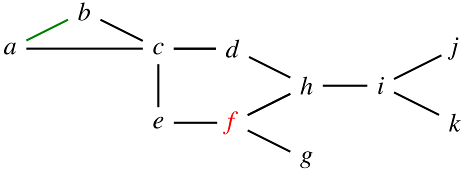
- A *tree decomposition* of G



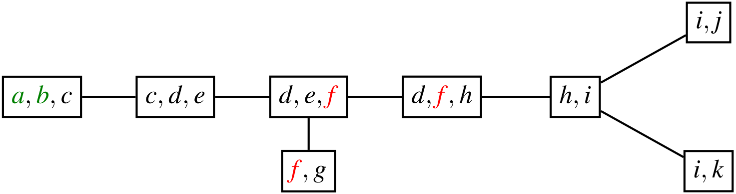
Conditions: covering

Tree decompositions (by example)

- A graph G



- A *tree decomposition* of G



Conditions: covering and connectedness.

Tree decomposition (more formally)

- Let G be a graph, T a tree, and χ a labeling of the nodes of T by subsets of $V(G)$.
- We refer to the sets $\chi(t)$ as “bags”.
- The pair (T, χ) is a tree decomposition of G if the following two conditions hold:
  - For every edge $vw \in E(G)$ there exists a node t of T such that $v, w \in \chi(t)$ (“covering”).
  - For every vertex v of G, the graph $T[t \in V(T) : v \in \chi(t)]$ is a non-empty (connected) tree (“connectedness”).

Treewidth

- The width of a tree decomposition (T, χ) is defined as the maximum $|\chi(t)| - 1$ over all nodes t of T.
- The treewidth $\mathrm{tw}(G)$ of a graph G is the minimum width over all its tree decompositions.

Treewidth of some graphs

- Trees have treewidth 1.
- Cycles have treewidth 2.
- The complete graph on n vertices has treewidth $n - 1$.

Treewidth of SAT formulas

- A CNF formula has treewidth t if its incidence graph has treewidth t.
- $\mathcal{W}_t$ denotes the class of all CNF formulas with treewidth at most t.

Outline

Outline of the FPT approximation algorithm

Algorithm TW-backdoor
Input: A CNF formula F and integers $k, t \geq 0$.
Output: A strong $\mathcal{W}_t$-backdoor of F of size $\leq 2^k$, or
No if F has no strong $\mathcal{W}_t$-backdoor of size k.

if *F has "small" treewidth [Bodlaender, 1996]* **then**
Express the problem in MSO_2 using [Adler, Grohe, Kreutzer, 2008] [Lagergren, 1998]
Use Courcelle's theorem [Courcelle, 1990] [Arnborg, Lagergren, Seese, 1991]

else
...

Outline of the FPT approximation algorithm

Algorithm TW-backdoor
Input: A CNF formula F and integers $k, t \geq 0$.
Output: A strong $\mathcal{W}_t$-backdoor of F of size $\leq 2^k$, or
No if F has no strong $\mathcal{W}_t$-backdoor of size k.

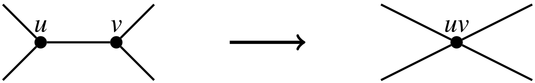
...
else
  Compute a large wall as a *topological minor* [Robertson, Seymour, Thomas '94] [Grohe, Kawarabayashi, Marx, Wollan '11]
  Compute a set S of $f(k,t)$ variables such that every strong $\mathcal{W}_t$-backdoor contains at least one of these variables
  foreach $x \in S$ **do**
    $B_x \leftarrow$ **TW-backdoor**$(F[x=1], k-1, t)$
    $B_{\neg x} \leftarrow$ **TW-backdoor**$(F[x=0], k-1, t)$
    if $(B_x \neq \text{No}) \wedge (B_{\neg x} \neq \text{No})$ **then**
      return $B_x \cup B_{\neg x} \cup \{x\}$
  return No

(Topological) Minors

Definition ((Topological) Minor)

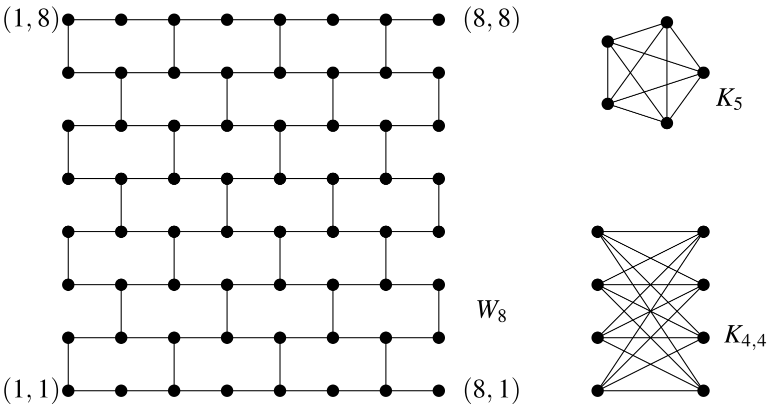
Let H, G be two graphs.
H is a (topological) minor of G if a graph isomorphic to H can be obtained from G by a sequence of the following operations:

- delete a vertex
- delete an edge
- contract an edge (incident to a vertex of degree 2)



Contract the edge uv

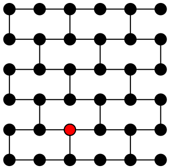
Obstructions for $\mathcal{W}_3$



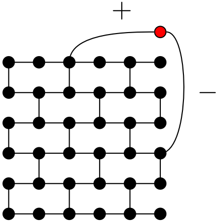
Using the Topological Wall Minor

- Large wall as a topological minor $\rightarrow$ many disjoint wall obstructions
- Each obstruction needs to be killed

Internal and External Killers



An internal killer



An external killer

- At most k wall obstructions are killed internally.
- $\Rightarrow$ "Guess" them and discard them
- All remaining obstructions are killed externally

Recall: we have many disjoint wall obstructions, and all of them need to be killed externally.

- $\geq 1/2^k$-th of all wall obstructions are killed externally by the same backdoor variables
- $\Rightarrow$ "Guess" this subset $\mathcal{O}$ of wall obstructions and the number ℓ of backdoor variables that kill them externally
- Denote by Z the set of common external killers of the wall obstructions in $\mathcal{O}$

Aim: Find a small subset $S \subseteq Z$ such that every valid (i.e., respecting our guesses) strong $\mathcal{W}_t$-backdoor contains a vertex from S. Then, S can be used for branching.

Recall: we have many disjoint wall obstructions, and all of them need to be killed externally.

- $\geq 1/2^k$-th of all wall obstructions are killed externally by the same backdoor variables
- $\Rightarrow$ "Guess" this subset $\mathcal{O}$ of wall obstructions and the number ℓ of backdoor variables that kill them externally
- Denote by Z the set of common external killers of the wall obstructions in $\mathcal{O}$

Aim: Find a small subset $S \subseteq Z$ such that every valid (i.e., respecting our guesses) strong $\mathcal{W}_t$-backdoor contains a vertex from S. Then, S can be used for branching.

We have 3 rules to construct S:

Rule 1 (Few Common Killers). If $|Z| \leq 6k\text{nb}(t)$, then set $S := Z$. ($\text{nb}(t) = \lceil 16(t+2)log(t+2) \rceil$)

Rule 2 (Multiple Neighborhoods). If there is a subset $L \subseteq Z$ such that L is the neighborhood of at least $t2^{\ell} + 1$ vertices in $\mathcal{B}_m(\mathcal{O})$, then set $S := L$.

Rule 3 (No Multiple Neighborhoods). Set S to be the $6k\text{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O})$ (ties are broken arbitrarily).

But what are $\mathcal{B}_m(\mathcal{O})$ and $\mathcal{B}(\mathcal{O})$?

We have 3 rules to construct S:

Rule 1 (Few Common Killers). If $|Z| \leq 6k\text{nb}(t)$, then set $S := Z$. ($\text{nb}(t) = \lceil 16(t+2)log(t+2) \rceil$)

Rule 2 (Multiple Neighborhoods). If there is a subset $L \subseteq Z$ such that L is the neighborhood of at least $t2^{\ell} + 1$ vertices in $\mathcal{B}_m(\mathcal{O})$, then set $S := L$.

Rule 3 (No Multiple Neighborhoods). Set S to be the $6k\text{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O})$ (ties are broken arbitrarily).

But what are $\mathcal{B}_m(\mathcal{O})$ and $\mathcal{B}(\mathcal{O})$?

We have 3 rules to construct S:

Rule 1 (Few Common Killers). If $|Z| \leq 6k\text{nb}(t)$, then set $S := Z$. ($\text{nb}(t) = \lceil 16(t+2) log(t+2) \rceil$)

Rule 2 (Multiple Neighborhoods). If there is a subset $L \subseteq Z$ such that L is the neighborhood of at least $t2^\ell + 1$ vertices in $\mathcal{B}_m(\mathcal{O})$, then set $S := L$.

Rule 3 (No Multiple Neighborhoods). Set S to be the $6k\text{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O})$ (ties are broken arbitrarily).

But what are $\mathcal{B}_m(\mathcal{O})$ and $\mathcal{B}(\mathcal{O})$?

We have 3 rules to construct S:

Rule 1 (Few Common Killers). If $|Z| \leq 6k\text{nb}(t)$, then set $S := Z$. ($\text{nb}(t) = \lceil 16(t+2) log(t+2) \rceil$)

Rule 2 (Multiple Neighborhoods). If there is a subset $L \subsetneq Z$ such that L is the neighborhood of at least $t2^{\ell} + 1$ vertices in $\mathcal{B}_m(\mathcal{O})$, then set $S := L$.

Rule 3 (No Multiple Neighborhoods). Set S to be the $6k\text{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O})$ (ties are broken arbitrarily).

But what are $\mathcal{B}_m(\mathcal{O})$ and $\mathcal{B}(\mathcal{O})$?

We have 3 rules to construct S:

Rule 1 (Few Common Killers). If $|Z| \leq 6k\text{nb}(t)$, then set $S := Z$. ($\text{nb}(t) = \lceil 16(t+2)log(t+2) \rceil$)

Rule 2 (Multiple Neighborhoods). If there is a subset $L \subseteq Z$ such that L is the neighborhood of at least $t2^\ell + 1$ vertices in $\mathcal{B}_m(\mathcal{O})$, then set $S := L$.

Rule 3 (No Multiple Neighborhoods). Set S to be the $6k\text{nb}(t)$ vertices from Z of highest degree in $\mathcal{B}(\mathcal{O})$ (ties are broken arbitrarily).

But what are $\mathcal{B}_m(\mathcal{O})$ and $\mathcal{B}(\mathcal{O})$?

The Beast

Definition (obstruction-template)

An obstruction-template $\mathrm{OT}(W)$ of a wall-obstruction $W \in \mathcal{O}$ is a triple $(\mathcal{B}(W), P, R)$, where

- $\mathcal{B}(W)$ is a bipartite graph whose vertex set is bipartitioned into the two independent sets Z and Q_W, where Q_W is a set of new vertices,
- P is a partition of $V(W)$ into *regions* such that for each region $A \in P$, we have that $W[A]$ is connected, and
- $R : Q_W \to P$ is a function associating a region of P with each vertex in Q_W.

The Beast (2)

Definition (valid obstruction-template)

An obstruction-template $\text{OT}(W) = (\mathcal{B}(W), P, R)$ of a wall-obstruction $W \in \mathcal{O}_s$ is *valid* if it satisfies the following properties:

(1) only existing edges: for each $q \in Q_W$ we have that $N_{\mathcal{B}(W)}(q) \subseteq N_G(R(q))$,

(2) private neighbor: for each $q \in Q_W$, there is a vertex $z \in N_{\mathcal{B}(W)}(q)$, called q's *private neighbor*, such that there is no other $q' \in N_{\mathcal{B}(W)}(z)$ with $R(q') = R(q)$,

(3) degree-Z: for each $z \in Z$ we have that $d_{\mathcal{B}(W)}(z) \geq 1$,

(4) degree-Q_W: for each $q \in Q_W$ we have that $\text{nb}(t) \leq d_{\mathcal{B}(W)}(q) \leq 3\text{nb}(t)$, and

(5) vulnerable vertex: for each $q \in Q_W$, there is at most one vertex $v \in R(q)$, called q's *vulnerable vertex*, such that $N_G(v) \cap Z \not\subseteq N_{\mathcal{B}(W)}(q)$.

The Beast (3)

$\mathcal{B}_m(\mathcal{O})$ is obtained by taking the union of all $\mathcal{B}(W)$, $W \in \mathcal{O}$.

$\mathcal{B}(\mathcal{O})$ is obtained from $\mathcal{B}_m(\mathcal{O})$ by merging vertices from $V(\mathcal{B}_m(\mathcal{O})) \setminus Z$ with identical neighborhoods.

Computing and Taming the Beast

- To identify a small $S \subseteq Z$ intersecting every valid strong $\mathcal{W}_t$-backdoor, we need to find obstructions involving S and $\mathcal{O}$ for at least one assignment to every candidate backdoor of size k avoiding S.
- Valid obstruction-templates model various ways to assemble such obstructions.
- A valid obstruction-template can be computed in $O(n^2)$ time.
- We prove that for a set S constructed by our rules, a valid $\mathcal{W}_t$-backdoor contains a variable from S, otherwise at least one assignment to the backdoor produces a formula whose incidence graph has treewidth at least $t+1$.

Computing and Taming the Beast

- To identify a small $S \subseteq Z$ intersecting every valid strong $\mathcal{W}_t$-backdoor, we need to find obstructions involving S and $\mathcal{O}$ for at least one assignment to every candidate backdoor of size k avoiding S.
- Valid obstruction-templates model various ways to assemble such obstructions.
- A valid obstruction-template can be computed in $O(n^2)$ time.
- We prove that for a set S constructed by our rules, a valid $\mathcal{W}_t$-backdoor contains a variable from S, otherwise at least one assignment to the backdoor produces a formula whose incidence graph has treewidth at least $t+1$.

Computing and Taming the Beast

- To identify a small $S \subseteq Z$ intersecting every valid strong $\mathcal{W}_t$-backdoor, we need to find obstructions involving S and $\mathcal{O}$ for at least one assignment to every candidate backdoor of size k avoiding S.
- Valid obstruction-templates model various ways to assemble such obstructions.
- A valid obstruction-template can be computed in $O(n^2)$ time.
- We prove that for a set S constructed by our rules, a valid $\mathcal{W}_t$-backdoor contains a variable from S, otherwise at least one assignment to the backdoor produces a formula whose incidence graph has treewidth at least $t+1$.

Computing and Taming the Beast

- To identify a small $S \subseteq Z$ intersecting every valid strong $\mathcal{W}_t$-backdoor, we need to find obstructions involving S and $\mathcal{O}$ for at least one assignment to every candidate backdoor of size k avoiding S.
- Valid obstruction-templates model various ways to assemble such obstructions.
- A valid obstruction-template can be computed in $O(n^2)$ time.
- We prove that for a set S constructed by our rules, a valid $\mathcal{W}_t$-backdoor contains a variable from S, otherwise at least one assignment to the backdoor produces a formula whose incidence graph has treewidth at least $t+1$.

Results for Bounded Treewidth

Theorem ([Gaspers, Szeider, 2012])

There is an FPT algorithm with parameter $k + t$ that either concludes that F has no strong $\mathcal{W}_t$-backdoor of size at most k or finds a strong $\mathcal{W}_t$-backdoor of F of size at most 2^k.

Corollary ([Gaspers, Szeider, 2012])

There is a cubic-time algorithm that, given a CNF formula F, computes the number of satisfying assignments of F or concludes that the smallest strong $\mathcal{W}_t$-backdoor of F is larger than k, for any pair of constants $k, t \geq 0$.

Related Results

- Faster and simpler randomized FPT algorithm for detecting weak FOREST-backdoors for r-CNF formulas (based on [Fomin, Lokshtanov, Misra, Saurabh, FOCS 2012])
- Also extends to the base class $\mathcal{W}_t \cap r$-CNF

Conclusion

- Aim at explaining the good running times of SAT solvers
- Is there a strong correlation between
  "the problem is FPT w.r.t. parameter k"
  and
  "heuristics work well if k is small"?
- Need simpler algorithms (randomization?)
- Is Strong FOREST/ $\mathcal{W}_t$-backdoor detection FPT?
- Combination of base classes

Thank you!

Questions? Comments?