Backdoors to Satisfiability

New Developments in
Exact Algorithms and Lower Bounds

IIT Delhi, 2014

M. S. Ramanujan
University of Bergen, Norway

Outline

- Motivation
- 2 perspectives on backdoors
- Parameterized algorithms for SAT via backdoors

Satisfiability

Satisfiability

- SATISFIABILITY: Is a given propositional formula satisfiable?

Satisfiability

- SATISFIABILITY: Is a given propositional formula satisfiable?

- One of the earliest problems shown to be NP-Complete.

Satisfiability

- SATISFIABILITY: Is a given propositional formula satisfiable?
- One of the earliest problems shown to be NP-Complete.
- Best known algorithm for 3-SAT — 1.308^n (Hertli, FOCS 2011)

Satisfiability

Satisfiability

- Several applications— problems like software model checking, chip verification etc. reduced to SAT and solved using SAT solvers.

Satisfiability

- Several applications— problems like software model checking, chip verification etc. reduced to SAT and solved using SAT solvers.
- Resulting instances often have up to a million variables and several million clauses.

Satisfiability

- Several applications— problems like software model checking, chip verification etc. reduced to SAT and solved using SAT solvers.
- Resulting instances often have up to a million variables and several million clauses.
- Even for 300 variables, worst case bounds exceed age of the universe.

That's all well and good in practice,
but how does it work in theory?

That's all well and good in practice,
but how does it work in theory?

The instances arising in practice must
have some structure!

Modern SAT solvers

Modern SAT solvers

Modern SAT solvers

- ‘Complete’ SAT solvers are variants of the DPLL algorithm.

Modern SAT solvers

- ‘Complete’ SAT solvers are variants of the DPLL algorithm.

DPLL= Davis-Putnam-Logemann-Loveland

The DPLL algorithm

The DPLL algorithm

- Perform Unit Propagation.

The DPLL algorithm

- Perform Unit Propagation.
- If there is a unit clause then the literal is set accordingly and the formula reduced.

The DPLL algorithm

- Perform Unit Propagation.
- If there is a unit clause then the literal is set accordingly and the formula reduced.

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

The DPLL algorithm

- Perform Unit Propagation.
- If there is a unit clause then the literal is set accordingly and the formula reduced.

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

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

The DPLL algorithm

- Perform Unit Propagation.
- If there is a unit clause then the literal is set accordingly and the formula reduced.

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

$$(x_3) \wedge (x_2 \vee \neg x_3) \longrightarrow$$

The DPLL algorithm

- Perform Unit Propagation.
- If there is a unit clause then the literal is set accordingly and the formula reduced.

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

$$(x_3)\wedge(x_2 \vee \neg x_3) \longrightarrow (x_2)$$

The DPLL algorithm

- Perform Unit Propagation.
- If there is a unit clause then the literal is set accordingly and the formula reduced.

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

$$\swarrow$$

$$(x_3) \wedge (x_2 \vee \neg x_3) \longrightarrow (x_2)$$

Horn formulas : formulas with at most one positive literal in every clause, solved just by unit propagation.

The DPLL algorithm

The DPLL algorithm

- Perform Pure Literal Elimination.

The DPLL algorithm

- Perform Pure Literal Elimination.
- If a variable occurs with a single polarity, then assign it and reduce formula.

The DPLL algorithm

- Perform Pure Literal Elimination.
- If a variable occurs with a single polarity, then assign it and reduce formula.

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

The DPLL algorithm

- Perform Pure Literal Elimination.
- If a variable occurs with a single polarity, then assign it and reduce formula.

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

$$\downarrow$$

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

The DPLL algorithm

- Perform Pure Literal Elimination.
- If a variable occurs with a single polarity, then assign it and reduce formula.

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

$$\downarrow$$

$$(x_1)\wedge(\neg x_1 \vee x_3) \longrightarrow$$

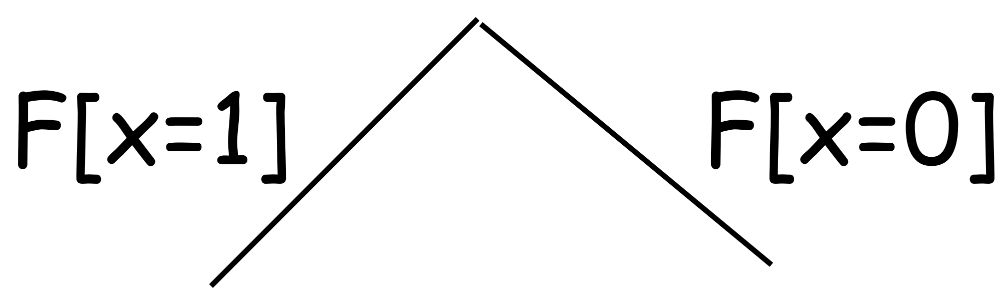
The DPLL algorithm

- Perform Pure Literal Elimination.
- If a variable occurs with a single polarity, then assign it and reduce formula.

$$\begin{array}{l} (x_1) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3) \\ \qquad \downarrow \\ (x_1) \wedge (\neg x_1 \vee x_3) \longrightarrow (x_1) \end{array}$$

The DPLL algorithm

- Select a variable (based on some heuristic) and explore both assignments.



Modern Sat solvers

Modern Sat solvers

- Build on DPLL by better variable selection heuristics.

Modern Sat solvers

- Build on DPLL by better variable selection heuristics.
- Better backtracking strategies.

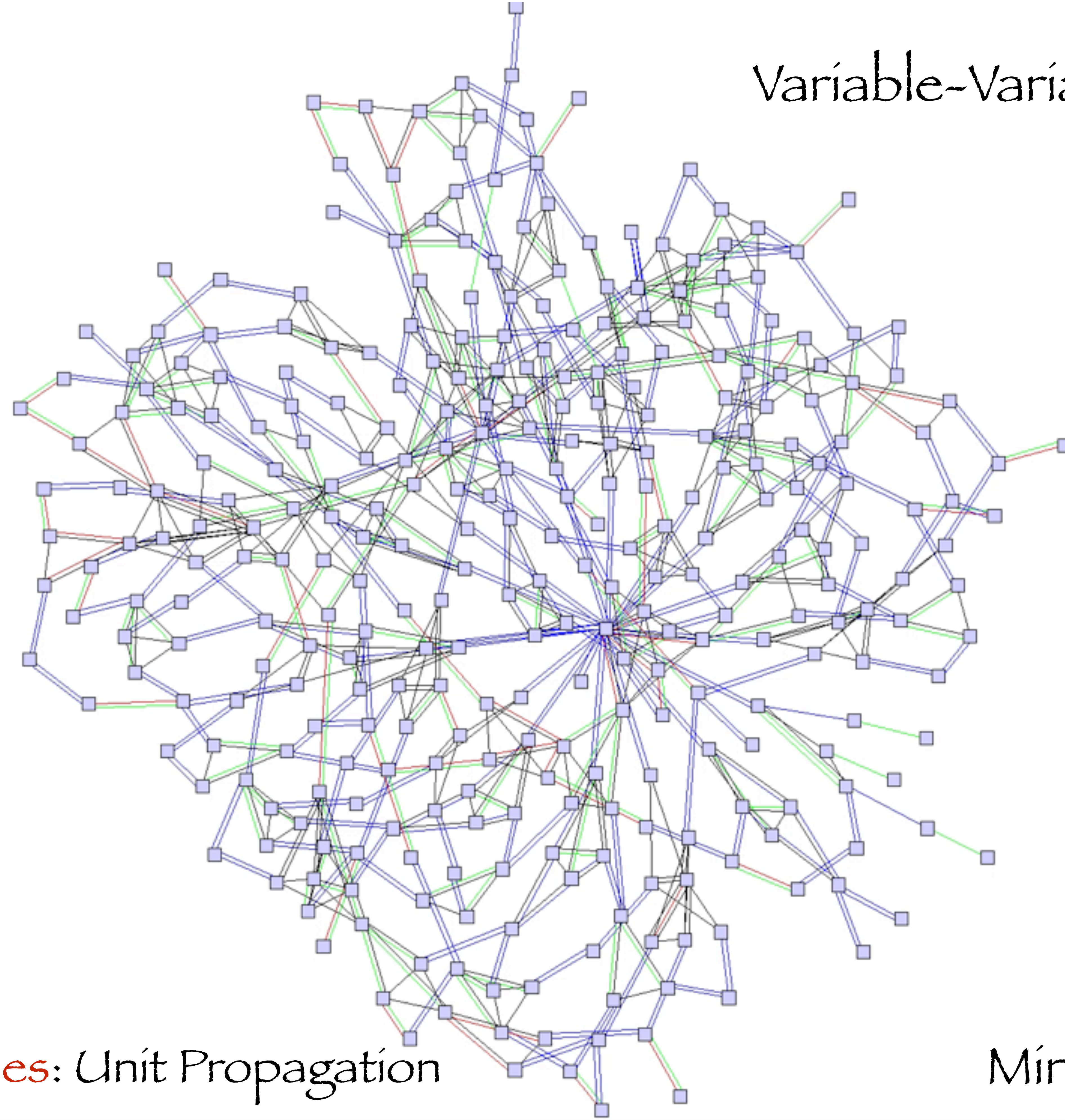
Modern Sat solvers

- Build on DPLL by better variable selection heuristics.
- Better backtracking strategies.
- ‘Learning’ clauses.

Modern Sat solvers

- Build on DPLL by better variable selection heuristics.
- Better backtracking strategies.
- ‘Learning’ clauses.
- ‘watching’ literals for fast unit propagations.

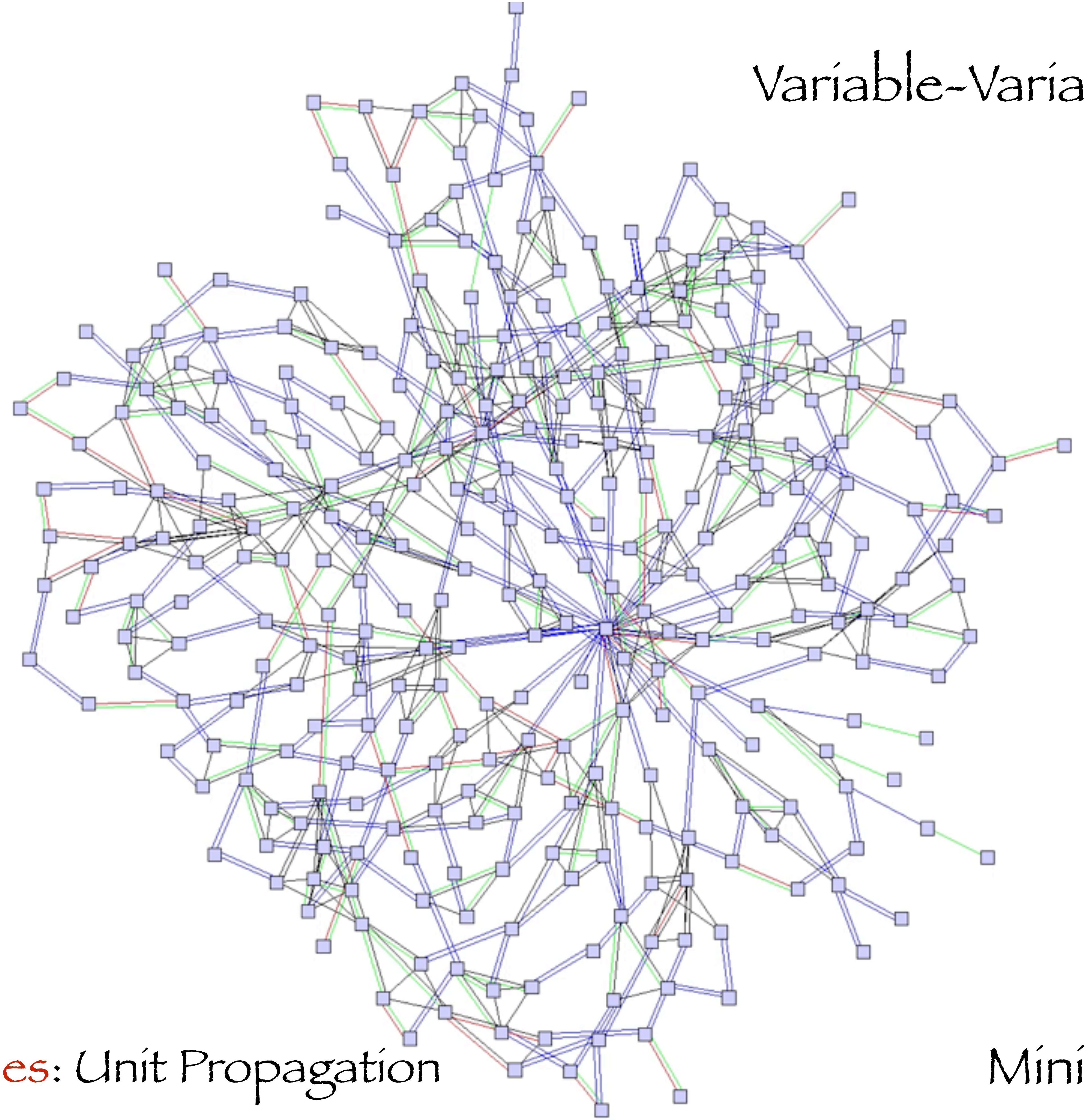
Variable-Variable Graph



Red nodes: Unit Propagation

Mini-SAT

Variable-Variable Graph



Red nodes: Unit Propagation

Mini-SAT

Variable dependencies

Variable dependencies

- Most of the variables seem ‘dependent’ on a few variables.

Variable dependencies

- Most of the variables seem 'dependent' on a few variables.
- Fixing an assignment to this set propagates to the rest of the variables.

Variable dependencies

- Most of the variables seem 'dependent' on a few variables.
- Fixing an assignment to this set propagates to the rest of the variables.
- Lots of real world instances seem to have a small set on which the remaining variables are dependent.

Variable dependencies

- Most of the variables seem 'dependent' on a few variables.
- Fixing an assignment to this set propagates to the rest of the variables.
- Lots of real world instances seem to have a small set on which the remaining variables are dependent.
- Can we capture the structure of an instance through this small set of variables ?

Backdoor sets

Introduced by
Williams, Gomes, Selman (IJCAI 2003)
and
Crama, Ekin, Hammer (D. A. M. 1997)

Informally, a set of variables whose instantiation results in a significantly simplified formula.

Subsolvers

A sub-solver is an algorithm with some nice properties.

Subsolvers

A sub-solver is an algorithm with some nice properties.

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).

Subsolvers

A sub-solver is an algorithm with some nice properties.

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).
- (Efficiency) runs in polynomial time.

Subsolvers

A sub-solver is an algorithm with some nice properties.

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).
- (Efficiency) runs in polynomial time.
- (Self-reducibility) solves reduced instances.

Subsolvers

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).
- (Efficiency) runs in polynomial time.
- (Self-reducibility) solves reduced instances.

Subsolvers

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).
- (Efficiency) runs in polynomial time.
- (Self-reducibility) solves reduced instances.

- Subsolver1: Solve all 2cnf formulas and reject the rest.

Subsolvers

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).
- (Efficiency) runs in polynomial time.
- (Self-reducibility) solves reduced instances.

- Subsolver1: Solve all 2cnf formulas and reject the rest.
- Subsolver2: Solve all Horn formulas and reject the rest.

Subsolvers

- (Trichotomy) input is either rejected or correctly solved (sat or unsat).
- (Efficiency) runs in polynomial time.
- (Self-reducibility) solves reduced instances.

- Subsolver1: Solve all 2cnf formulas and reject the rest.
- Subsolver2: Solve all Horn formulas and reject the rest.

Subsolvers

- Subsolver1: Solve all 2cnf formulas and reject the rest.
- Subsolver2: Solve all Horn formulas and reject the rest.

Subsolvers

The instances NOT rejected by a sub-solver can be treated as a tractable base class for SAT.

- Subsolver1: Solve all 2cnf formulas and reject the rest.
- Subsolver2: Solve all Horn formulas and reject the rest.

Subsolvers

The instances NOT rejected by a sub-solver can be treated as a tractable base class for SAT.

For every tractable base class for SAT, we have a sub-solver that solves instances in this class and rejects the rest.

- Subsolver1: Solve all 2cnf formulas and reject the rest.
- Subsolver2: Solve all Horn formulas and reject the rest.

Subsolvers

Subsolvers

Base class = sub-solver

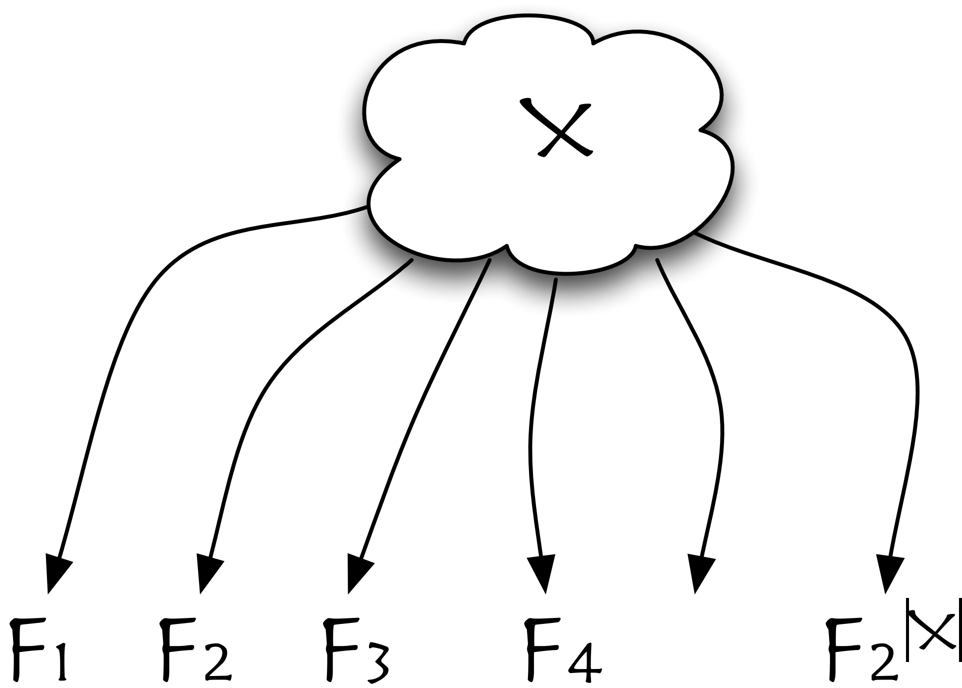
Backdoors to SAT

Backdoors to SAT

- Weak Backdoor to C

Backdoors to SAT

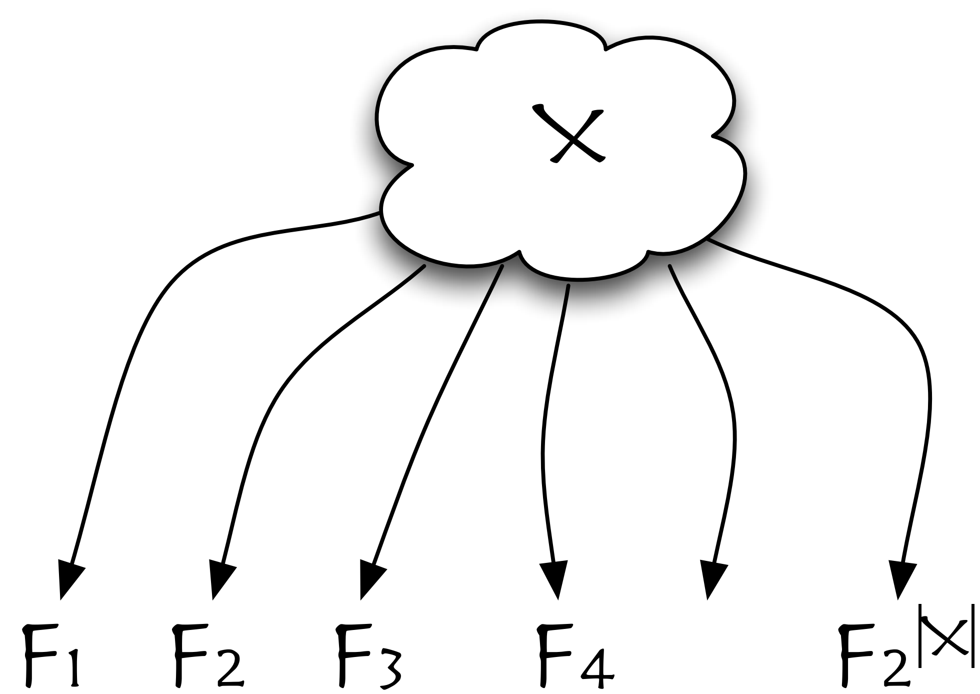
- Weak Backdoor to C



Some assignment leads to a satisfiable instance in the base class C

Backdoors to SAT

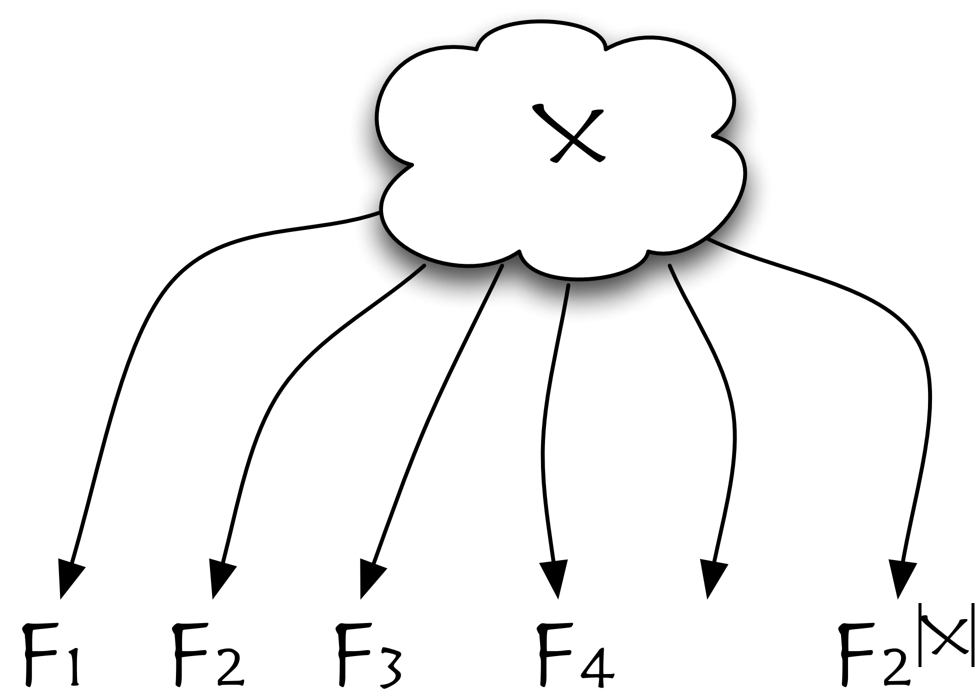
- Weak Backdoor to C
- Strong Backdoor to C



Some assignment leads to a satisfiable instance in the base class C

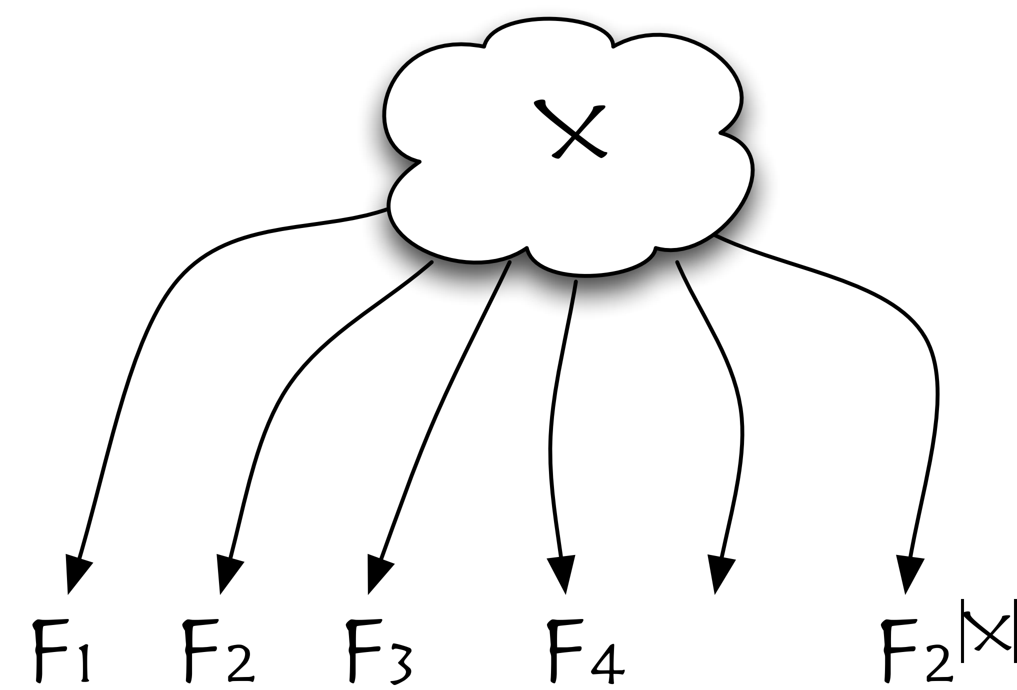
Backdoors to SAT

- Weak Backdoor to C



Some assignment leads to a satisfiable instance in the base class C

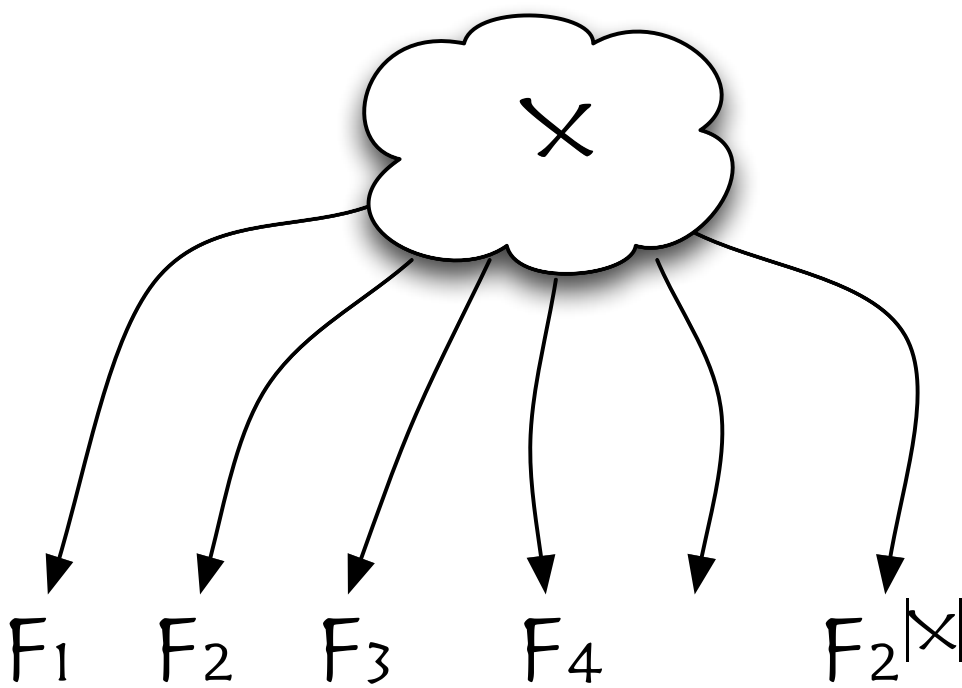
- Strong Backdoor to C



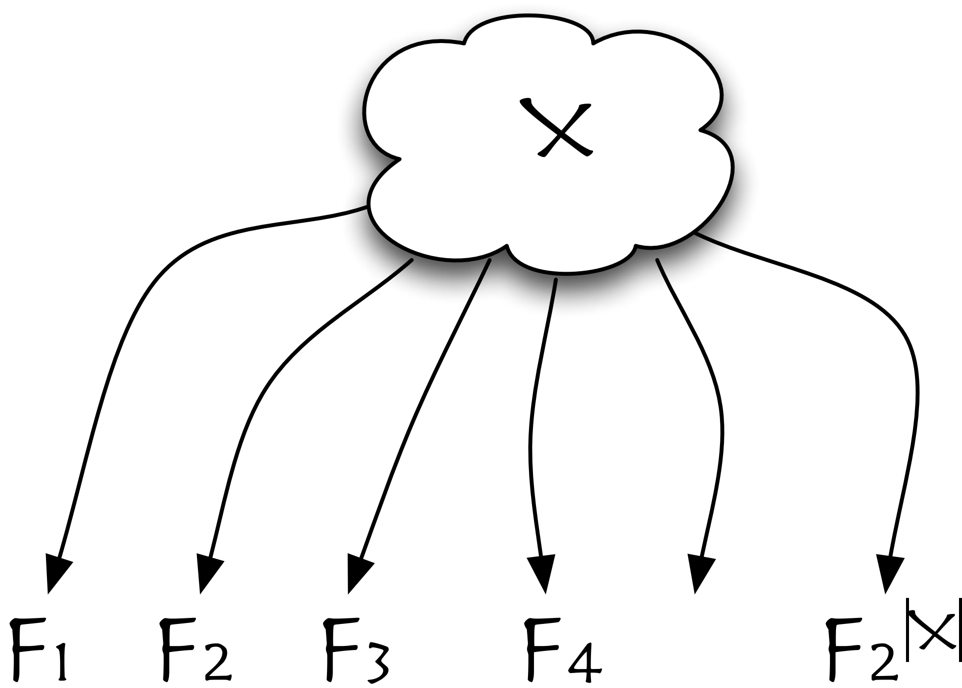
All assignments lead to an instance in the base class C.

Strong vs weak Backdoors

Strong vs weak Backdoors

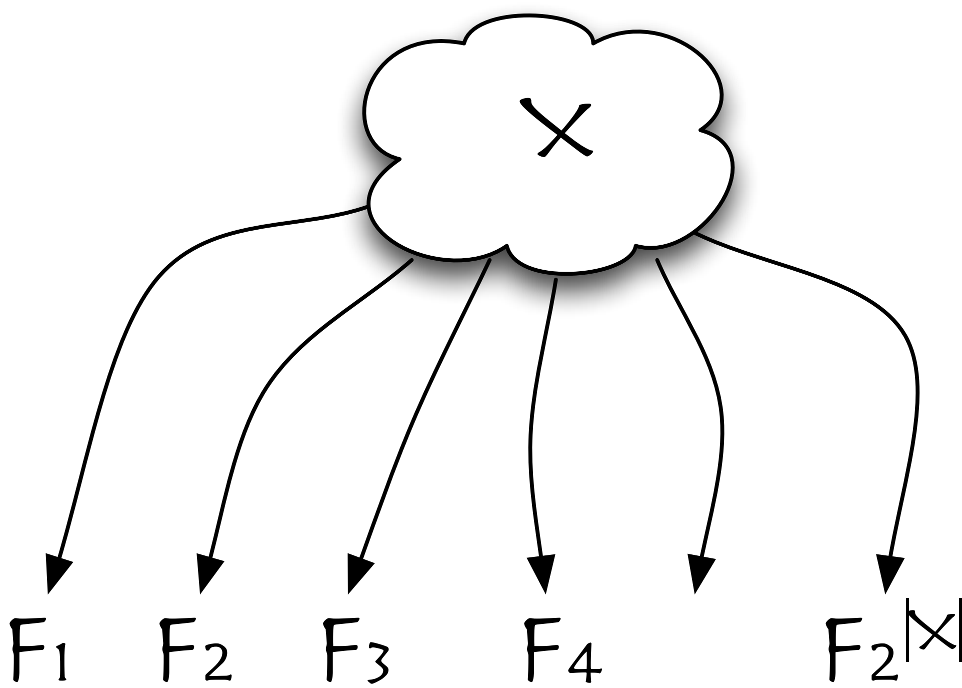


Strong vs weak Backdoors



If the instance is satisfiable then every strong backdoor is also a weak backdoor!

Strong vs weak Backdoors



If the instance is satisfiable then every strong backdoor is also a weak backdoor!

wbd ≤ sbd

2 Perspectives on backdoor sets

2 Perspectives on backdoor sets

- (a) (Williams, Gomes, Selman) the presence of small backdoor sets provides a good explanation for the performance of SAT solvers, the success of random restarts etc.

2 Perspectives on backdoor sets

- (a) (Williams, Gomes, Selman) the presence of small backdoor sets provides a good explanation for the performance of SAT solvers, the success of random restarts etc.

- (b) (Crama, Ekin, Hammer) backdoor sets provide an excellent framework to extend tractability results for SAT.

2 Perspectives on backdoor sets

- (a) (Williams, Gomes, Selman) the presence of small backdoor sets provides a good explanation for the performance of SAT solvers, the success of random restarts etc.
- (b) (Crama, Ekin, Hammer) backdoor sets provide an excellent framework to extend tractability results for SAT.

eg. SAT is in P for 2-cnf formulas—> SAT is in P for formulas with a strong backdoor of size 10 to 2-cnf.

Islands of Tractability

Islands of Tractability

- Think of the base classes as `Islands of tractability'.

Islands of Tractability

- Think of the base classes as `Islands of tractability'.

- An instance with a `small' backdoor to one of these base classes is `close' to an island of tractability.

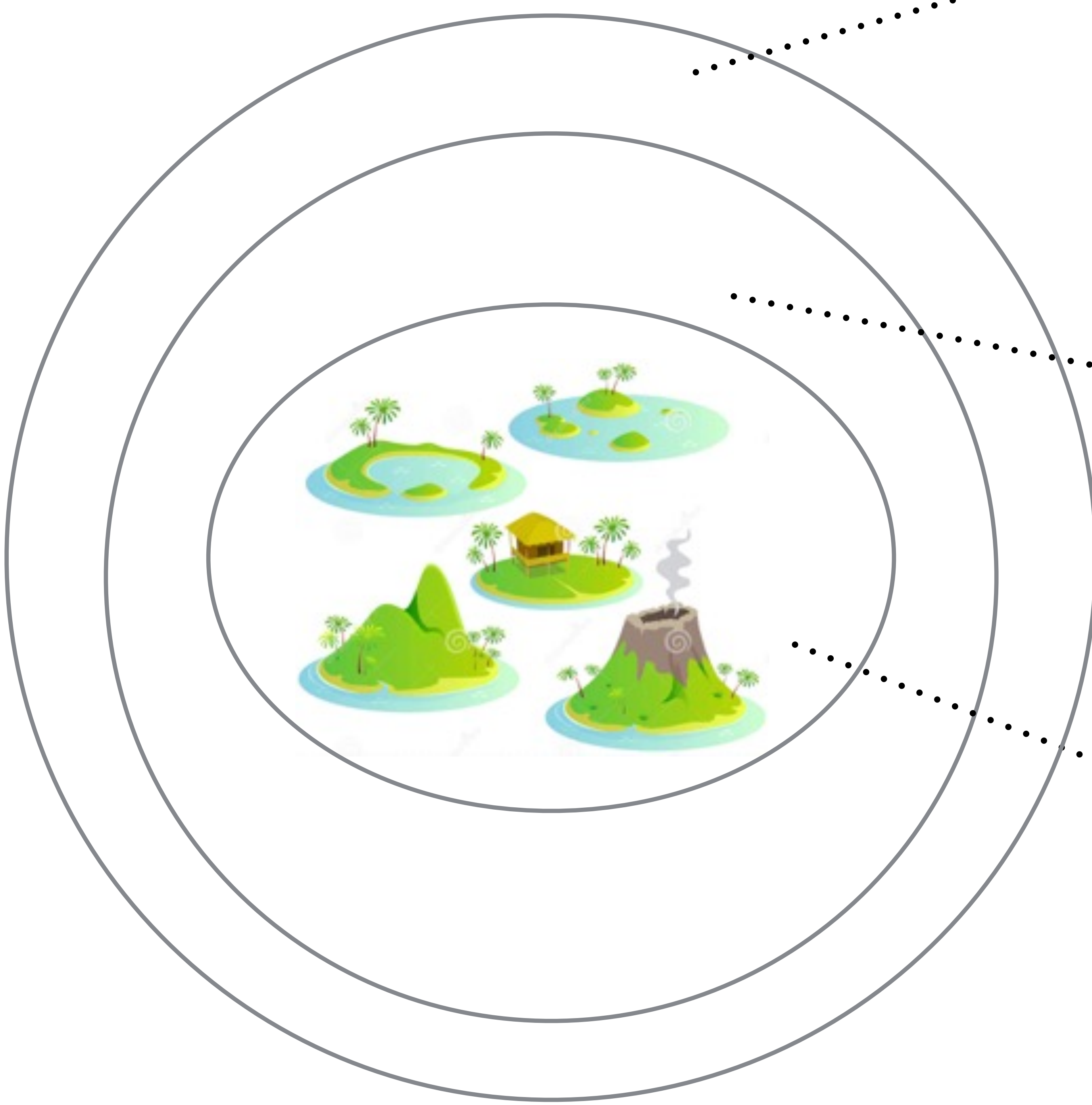
Islands of Tractability

- Think of the base classes as `Islands of tractability'.

- An instance with a `small' backdoor to one of these base classes is `close' to an island of tractability.

- Objective: If an instance is close to an island of tractability, then we can solve it efficiently.

Research Agenda



Instances with a backdoor of size $\log^2 n$ to an island

Instances with a backdoor of size $\log n$ to an island

Instances with a backdoor of size c to an island

How to extend tractability results?

How to extend tractability results?

- One approach: Find a weak/strong backdoor to a base class, explore all assignments to the backdoor variables.

How to extend tractability results?

- One approach: Find a weak/strong backdoor to a base class, explore all assignments to the backdoor variables.
- For each reduced formula, run the sub-solver for this base class.

How to extend tractability results?

- One approach: Find a weak/strong backdoor to a base class, explore all assignments to the backdoor variables.
- For each reduced formula, run the sub-solver for this base class.
- Running time is $T(D)+2^{|X|}.T(A)$; X is the backdoor set, D is the detection algorithm and A is the sub-solver.

How to extend tractability results?

- One approach: $\boxed{\text{Find a weak/strong backdoor}}$ to a base class, explore all assignments to the backdoor variables.
- For each reduced formula, run the sub-solver for this base class.
- Running time is $T(D)+2^{|X|}.T(A)$; X is the backdoor set, D is the detection algorithm and A is the sub-solver.

Finding Backdoor sets

Finding Backdoor sets

- How do we detect that an instance is `close' to an island of tractability?

Finding Backdoor sets

- How do we detect that an instance is `close' to an island of tractability?

- For which islands can we do this detection efficiently (in polynomial time)?

Finding Backdoor sets

Finding Backdoor sets

- For any reasonable island of tractability, detecting if an instance is `close' to this island is NP-complete.

Finding Backdoor sets

- For any reasonable island of tractability, detecting if an instance is `close' to this island is NP-complete.

- For which islands can we do this detection efficiently (for a relaxed notion of efficient)?

Finding Backdoors

Finding Backdoors

- How to develop and analyze `efficient' algorithms to detect small backdoors?

Finding Backdoors

- How to develop and analyze `efficient' algorithms to detect small backdoors?

Fixed-Parameter Algorithms!

FPT algorithms

FPT algorithms

- Parameterized problems - (x,k); k is the parameter.

FPT algorithms

- Parameterized problems - (x,k); k is the parameter.

- 2-dimensional analysis of algorithms.

FPT algorithms

- Parameterized problems - (x,k); k is the parameter.
- 2-dimensional analysis of algorithms.
- Aim for running times of the form $f(k)\ |x|^c$

FPT algorithms

- Parameterized problems - (x,k); k is the parameter.
- 2-dimensional analysis of algorithms.
- Aim for running times of the form $f(k)\ |x|^c$

FPT

FPT algorithms

FPT algorithms

- Running time $f(k)\,|x|^c$ implies that for k bounded by $f^{-1}(\mathrm{poly}(n))$, we have a poly time algorithm.

FPT algorithms

- Running time $f(k)\,|x|^c$ implies that for k bounded by $f^{-1}(poly(n))$, we have a poly time algorithm.

- Corresponding hardness theory.

FPT algorithms

- Running time $f(k)\,|x|^c$ implies that for k bounded by $f^{-1}(\mathrm{poly}(n))$, we have a poly time algorithm.

- Corresponding hardness theory.

- W-hierarchy: $FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq XP$

Rest of this talk

Rest of this talk

- Recent advances in FPT algorithms for computing backdoors to some base classes (q-Horn, tw-SAT, composite classes)

Rest of this talk

- Recent advances in FPT algorithms for computing backdoors to some base classes (q-Horn, tw-SAT, composite classes)

- Discuss some interesting connections between the 2 perspectives.

Classical sub-solvers

Classical sub-solvers

Schaefer Classes

Classical sub-solvers

Schaefer Classes

- Horn (at most one positive lit in each clause)

Classical sub-solvers

Schaefer Classes

- Horn (at most one positive lit in each clause)
- Dual-Horn (at most one negative lit in each clause)

Classical sub-solvers

Schaefer Classes

- Horn (at most one positive lit in each clause)
- Dual-Horn (at most one negative lit in each clause)
- 2-cnf (at most 2 literals in each clause)

Classical sub-solvers

Schaefer Classes

- Horn (at most one positive lit in each clause)
- Dual-Horn (at most one negative lit in each clause)
- 2-cnf (at most 2 literals in each clause)
- 0/1-valid (satisfied by the all-0/all-1 assignment)

Classical sub-solvers

Schaefer Classes

- Horn (at most one positive lit in each clause)
- Dual-Horn (at most one negative lit in each clause)
- 2-cnf (at most 2 literals in each clause)
- 0/1-valid (satisfied by the all-0/all-1 assignment)

Nishimura, Ragde, Szeider SAT 2004

Classical sub-solvers

Class Backdoor	Schaefer [Nishimura, Ragde, Szeider 2004]	Unit Prop + Pure Lit. Elim [Szeider 2005]
Weak	W[2]-hard (FPT for 3-cnf)	W[2]-hard (FPT for 3-cnf)
Strong	FPT	W[2]-hard

Why is weak backdoor detection hard?

Why is weak backdoor detection hard?

- Existence of a small weak backdoor —-> the formula is satisfiable!

Why is weak backdoor detection hard?

- Existence of a small weak backdoor —-> the formula is satisfiable!

- Allows fairly straightforward encodings from Hitting Set/Set Cover, both W[2]-hard parameterized by the size of the solution (the hitting set or the set cover).

Why is weak backdoor detection hard?

- Existence of a small weak backdoor —-> the formula is satisfiable!

- Allows fairly straightforward encodings from Hitting Set/Set Cover, both W[2]-hard parameterized by the size of the solution (the hitting set or the set cover).

- Can change if restricted to 3-cnf formulas.

Backdoors to q-Horn

Backdoors to q-Horn

Backdoors to q-Horn

- quadratic-Horn (q-Horn) (Boros, Crama, Hammer 1990)

Backdoors to q-Horn

- quadratic-Horn (q-Horn) (Boros, Crama, Hammer 1990)

- F is q-Horn if there is a weight function
  w: lit(F)->{0,1/2,1} s.t

w(x)+w(¬x)=1 and for every clause C, w(C)≤ 1.

Backdoors to q-Horn

- quadratic-Horn (q-Horn) (Boros, Crama, Hammer 1990)

- F is q-Horn if there is a weight function
  w: lit(F)->{0,1/2,1} s.t

  $w(x)+w(\neg x)=1$ and for every clause C, $w(C) \leq 1$.

- q-Horn generalizes Horn and 2-cnf.

Backdoors to q-Horn

- quadratic-Horn (q-Horn) (Boros, Crama, Hammer 1990)

- F is q-Horn if there is a weight function
  w: lit(F)->{0,1/2,1} s.t

  w(x)+w(¬x)=1 and for every clause C, w(C)≤ 1.

- q-Horn generalizes Horn and 2-cnf.

w(x)=1 and w(¬x)=0 for all x   w(x)=w(¬x)=1/2 for all x

Backdoors to q-Horn

- quadratic-Horn (q-Horn) (Boros, Crama, Hammer 1990)

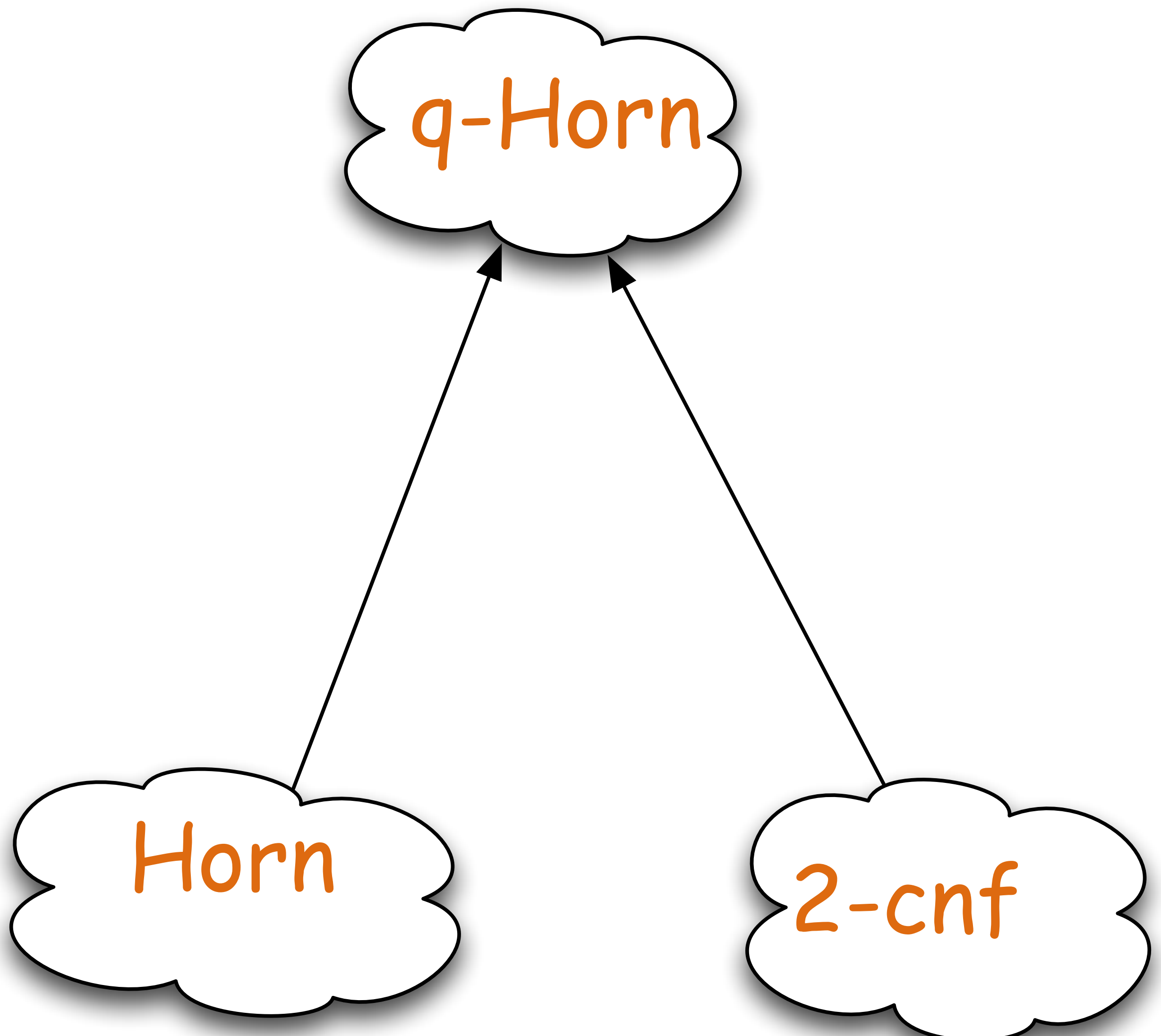
- F is q-Horn if there is a weight function
  w: lit(F)->{0,1/2,1} s.t

  $w(x)+w(\neg x)=1$ and for every clause C, $w(C) \leq 1$.

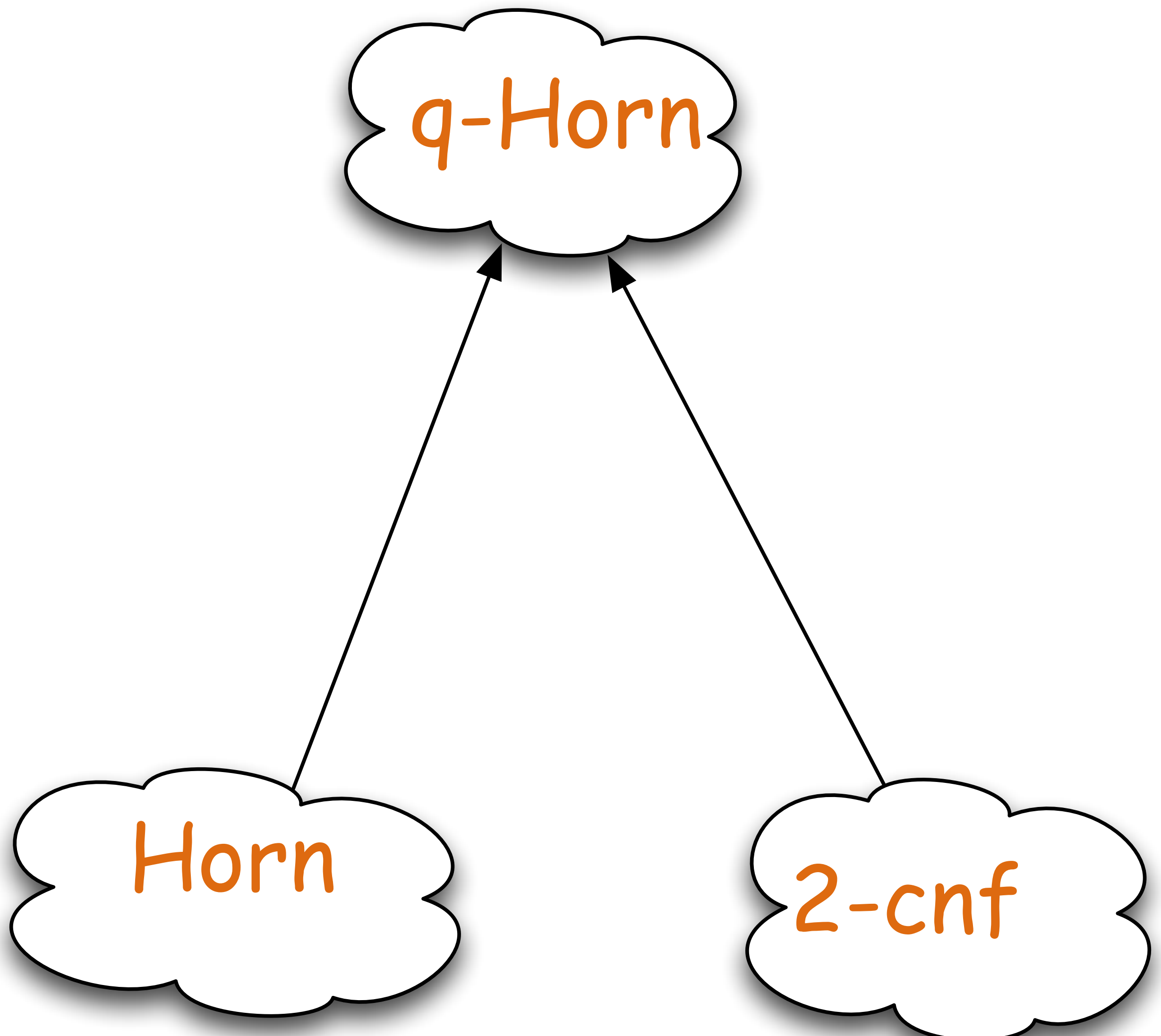
- q-Horn generalizes Horn and 2-cnf.

SAT is in P for
q-Horn forulas!

Backdoors to q-Horn

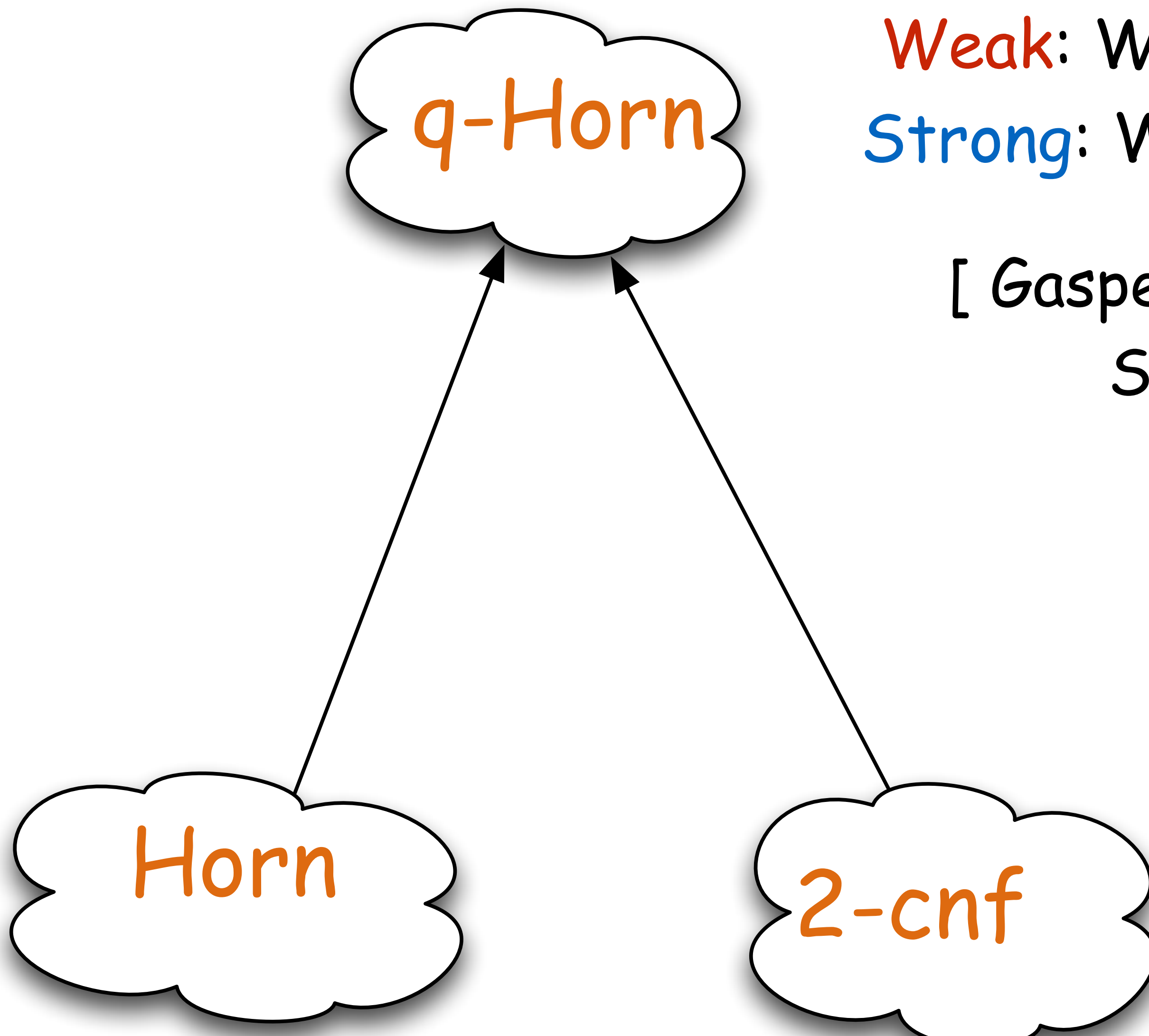


Backdoors to q-Horn



Weak: W[2]-hard
Strong: FPT

Backdoors to q-Horn



Weak: W[2]-hard
Strong: W[2]-hard

[Gaspers, Ordyniak, R., Saurabh, Szeider STACS 2013]

Weak: W[2]-hard
Strong: FPT

Backdoors to q-Horn

Backdoors to q-Horn

- How can we extend tractability results if strong backdoor detection is also W-hard?

Backdoors to q-Horn

- How can we extend tractability results if strong backdoor detection is also W-hard?
- What other notions of `distance' can we have?

Backdoors to q-Horn

- How can we extend tractability results if strong backdoor detection is also W-hard?
- What other notions of `distance' can we have?
- What about distance through deletion instead of instantiation?

Deletion Backdoors

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

$$(x_1) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (\neg x_3 \vee x_4)$$

Deleting x_2

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

$$(x_1) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (\neg x_3 \vee x_4)$$

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5) \wedge (x_2 \vee x_4)$$

Deleting x_2

Deleting x_3

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5) \wedge (x_2 \vee x_4)$$

Deleting x_3

x_3 is a deletion backdoor into 2-cnf.

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

$x_3 = 0$

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5) \wedge (x_2 \vee x_4)$$

Deleting x_3

x_3 is a deletion backdoor into 2-cnf.

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

$x_3 = 0$ $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5)$

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5) \wedge (x_2 \vee x_4)$$

Deleting x_3

x_3 is a deletion backdoor into 2-cnf.

Deletion Backdoors

- A deletion backdoor set of F to the base class C is a set of variables S such that F-S is in C.

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_3 \vee x_5) \wedge (x_2 \vee \neg x_3 \vee x_4)$$

$x_3 = 0$ $(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5)$

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_5) \wedge (x_2 \vee x_4)$$

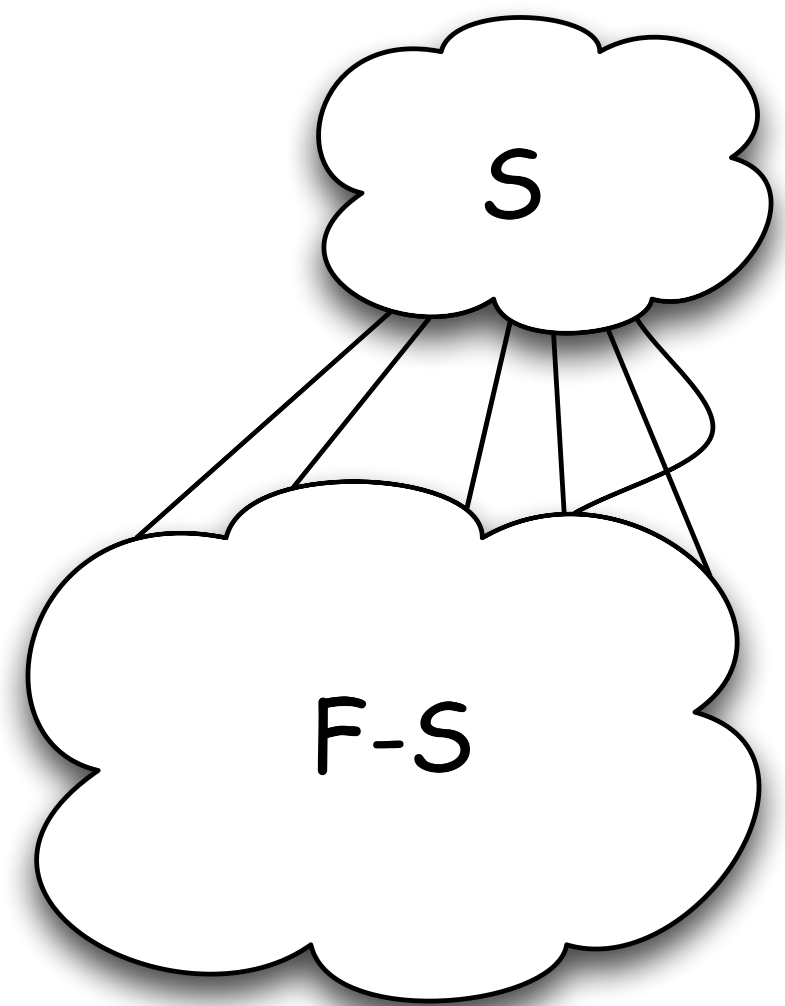
Deleting x_3

$x_3 = 1$ $(x_1 \vee x_2) \wedge (x_2 \vee x_4)$

x_3 is a deletion backdoor into 2-cnf.

Deletion Backdoors

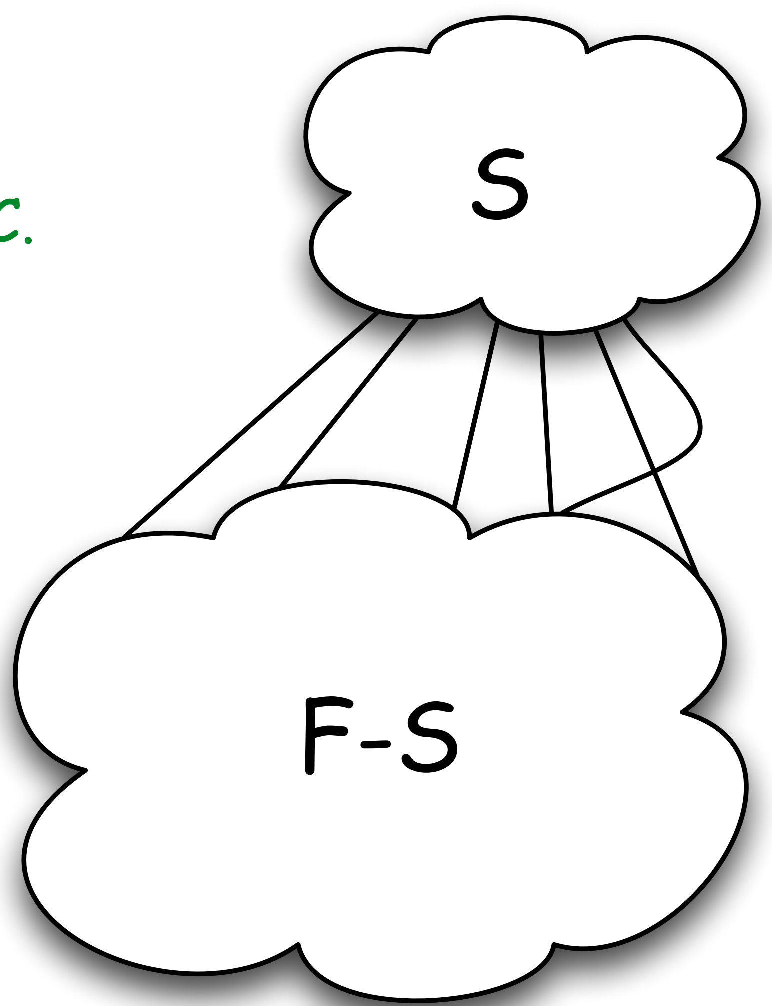
Deletion Backdoors



Deletion Backdoors

If F is in C, every subformula of F
induced by a subset of clauses is in C.

If the base class C is 'clause induced'
then S is also a strong backdoor to C.

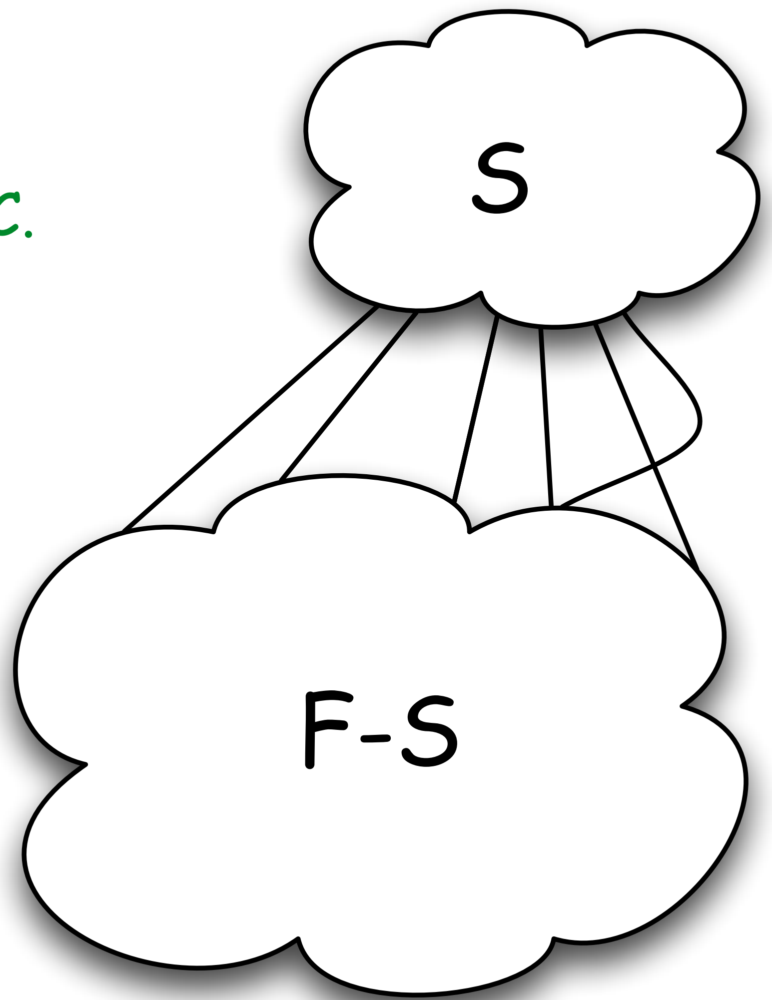


Deletion Backdoors

wbd ≤ sbd ≤ del. bd

If F is in C, every subformula of F induced by a subset of clauses is in C.

If the base class C is 'clause induced' then S is also a strong backdoor to C.



Deletion Backdoors

Deletion Backdoors

Deletion Backdoors

- The detection of strong backdoors being W-hard is not a dead end.

Deletion Backdoors

- The detection of strong backdoors being W-hard is not a dead end.

- If we can detect deletion backdoors then we can still extend the tractable region for SAT.

Backdoors to q-Horn

Backdoors to q-Horn

Backdoors to q-Horn

- q-Horn is clause induced. Can we find a deletion backdoor to q-Horn in FPT time?

Backdoors to q-Horn

- q-Horn is clause induced. Can we find a deletion backdoor to q-Horn in FPT time?
- In FPT time, we can approximate it.
  [Gaspers, Ordyniak, R., Saurabh, Szeider STACS 2013]

Backdoors to q-Horn

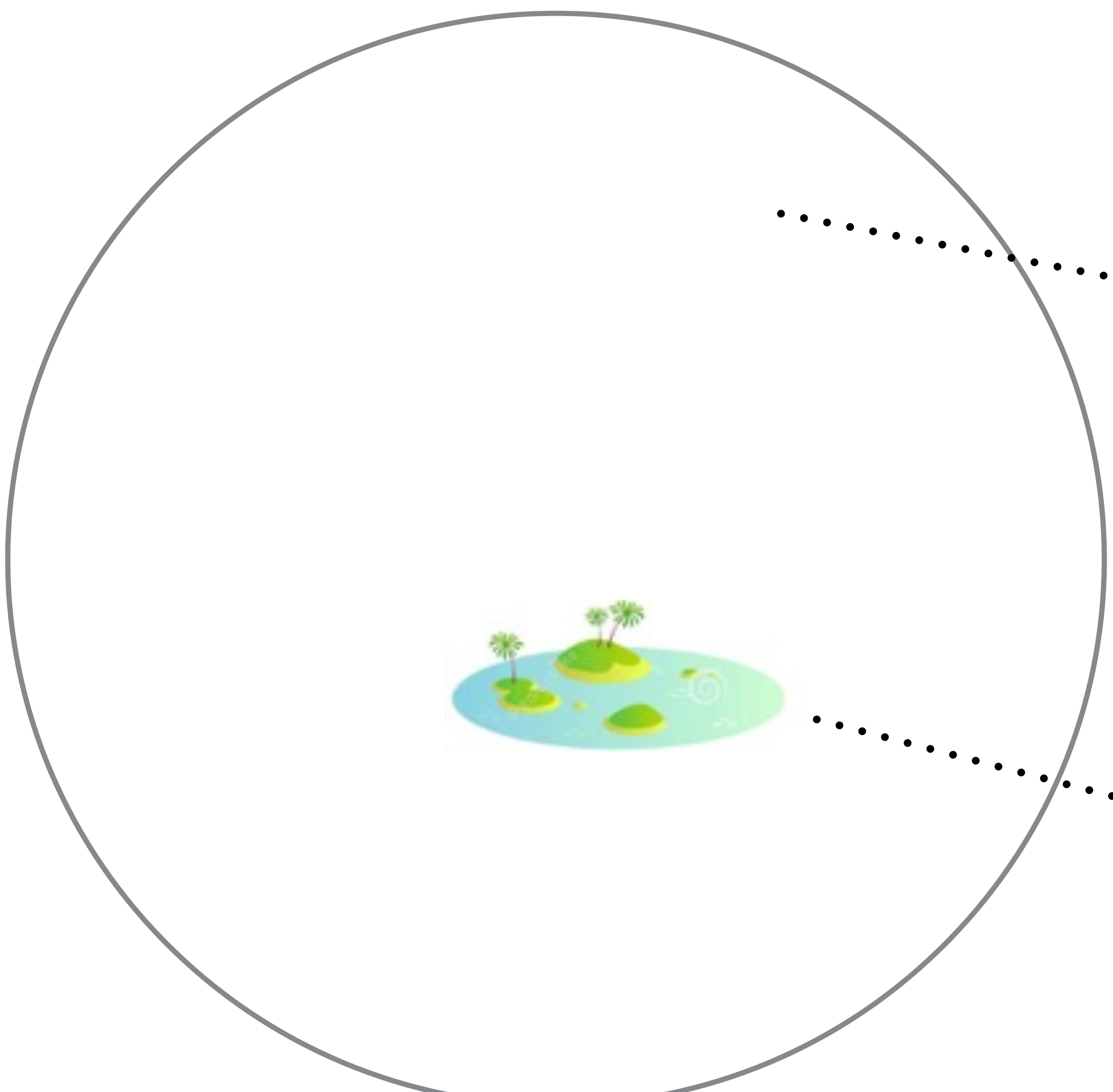
- q-Horn is clause induced. Can we find a deletion backdoor to q-Horn in FPT time?
- In FPT time, we can approximate it.
  [Gaspers, Ordyniak, R., Saurabh, Szeider STACS 2013]
- In $O(6^k mn)$ time, we can either conclude no del backdoor of size k or compute a deletion backdoor of size at most $2k^2$.

Backdoors to q-Horn

- q-Horn is clause induced. Can we find a deletion backdoor to q-Horn in FPT time?
- In FPT time, we can approximate it. [Gaspers, Ordyniak, R., Saurabh, Szeider STACS 2013]
- In $O(6^k mn)$ time, we can either conclude no del backdoor of size k or compute a deletion backdoor of size at most $2k^2$.

SAT parameterized by size of deletion backdoor to q-Horn can be solved in time $2^{O(k^2)} mn$.

Backdoors to q-Horn



SAT is in P for instances with a del. backdoor of size $O(\sqrt{\log n})$ to q-Horn.

SAT is in P for q-Horn

Backdoors to q-Horn

[R. and Saurabh, SODA 2014]

Deletion backdoor set detection to q-Horn can be solved in time $O(12^k\, m)$.

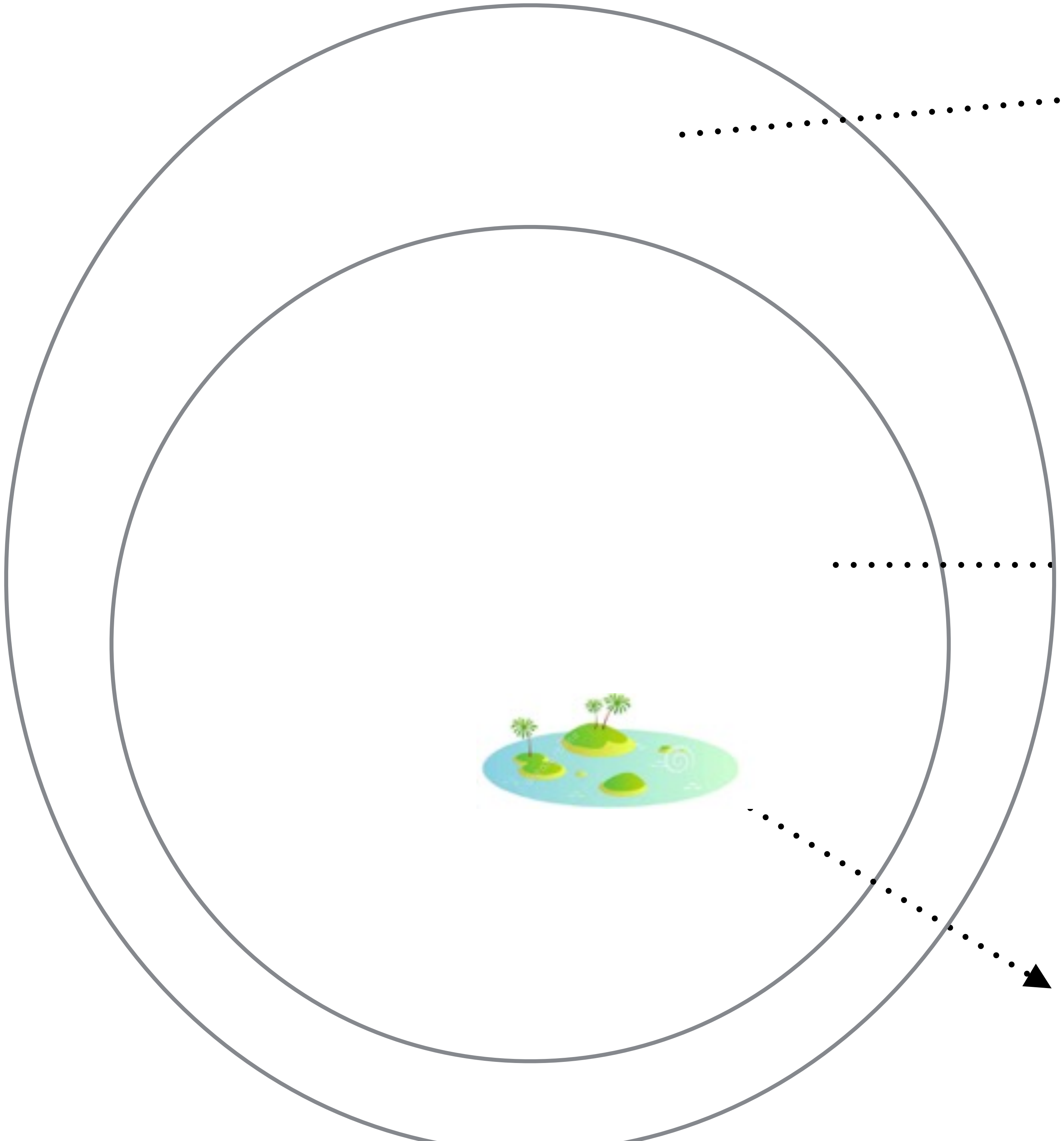
Backdoors to q-Horn

[R. and Saurabh, SODA 2014]

Deletion backdoor set detection to q-Horn can be solved in time $O(12^k m)$.

SAT parameterized by size of deletion backdoor to q-Horn can be solved in time $2^{O(k)} m$.

Backdoors to q-Horn



SAT is in P for instances with a del. backdoor of size O(log n) to q-Horn [R. and Saurabh 2014].

SAT is in P for instances with a del. backdoor of size O(√log n) to q-Horn [Gaspers, Ordyniak, R., Saurabh, Szeider 2014].

SAT is in P for q-Horn

Backdoors to q-Horn

Backdoors to q-Horn

- A linear time algorithm for SAT instances \`close' to being q-Horn.

Backdoors to q-Horn

- A linear time algorithm for SAT instances `close' to being q-Horn.
- Corollary: Deletion backdoor detection for RHorn can be done in time $O(4^k m)$.

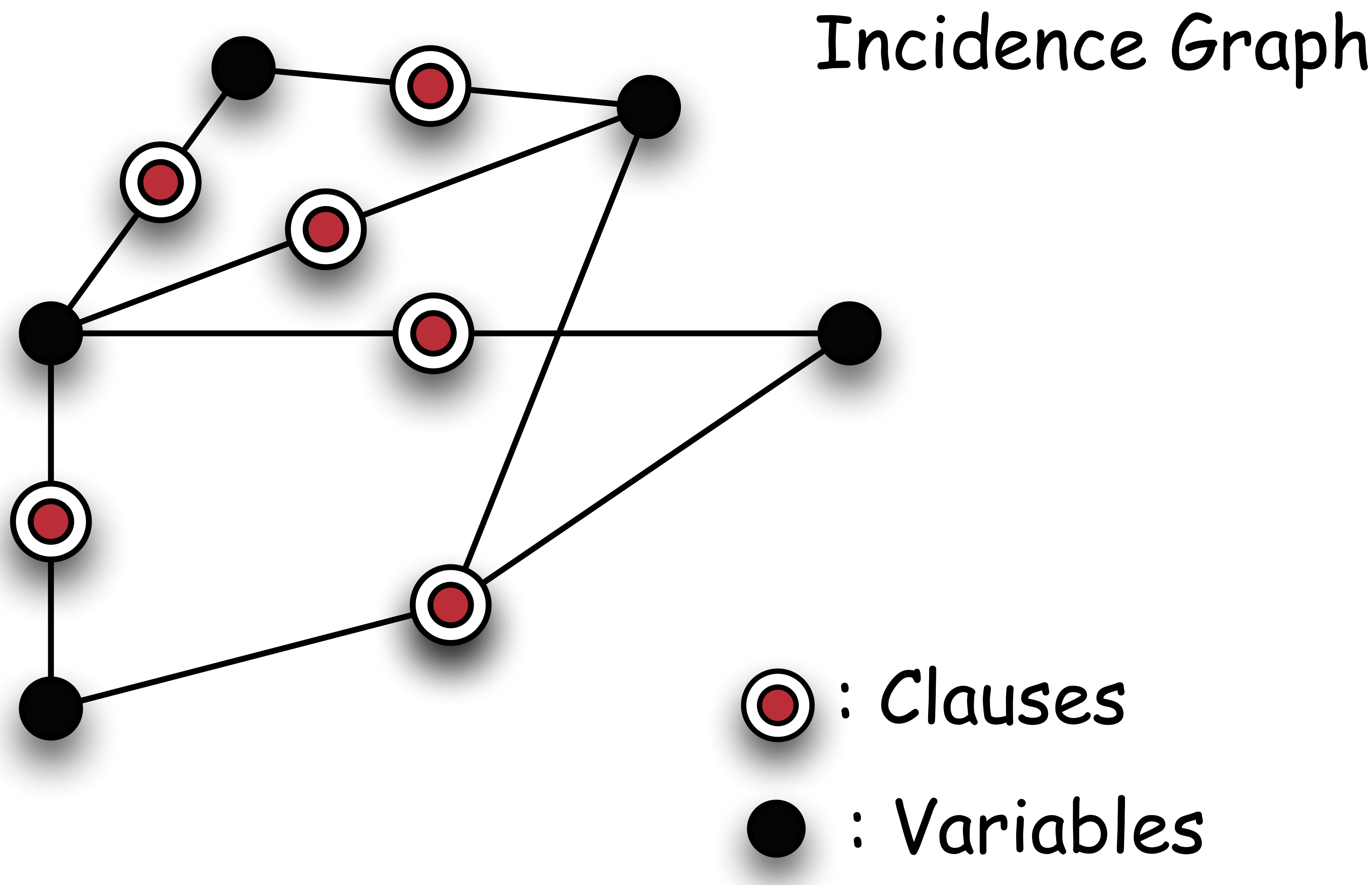
Backdoors to q-Horn

- A linear time algorithm for SAT instances `close' to being q-Horn.
- Corollary: Deletion backdoor detection for RHorn can be done in time $O(4^k m)$.
- A further consequence of this algorithm: the first linear time FPT algorithm for Odd Cycle Transversal (open problem of Reed, Smith and Vetta, 2003).

Backdoors to Bounded Treewidth SAT

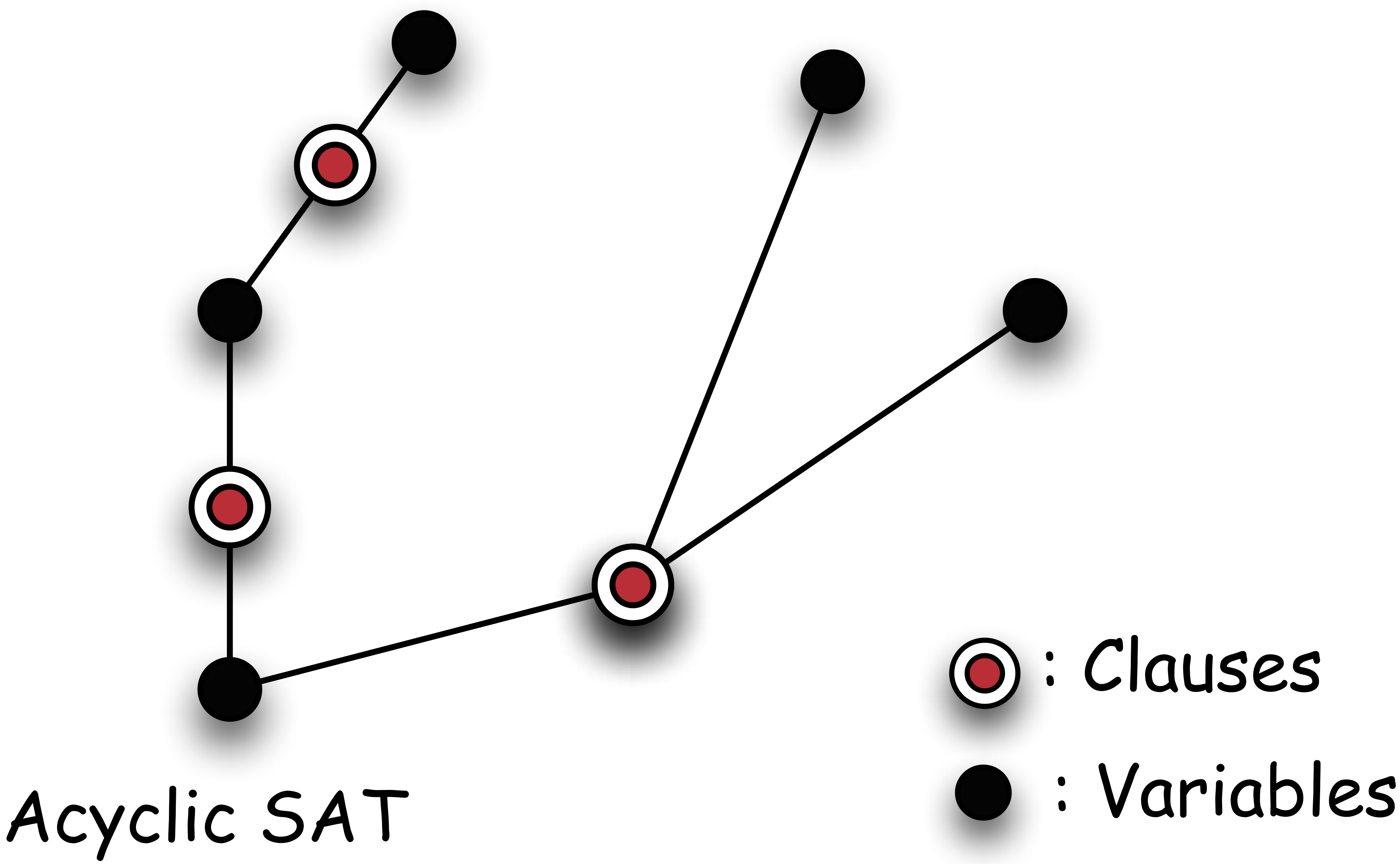
Backdoors to acyclic SAT

Modeling CNF-formulas as graphs



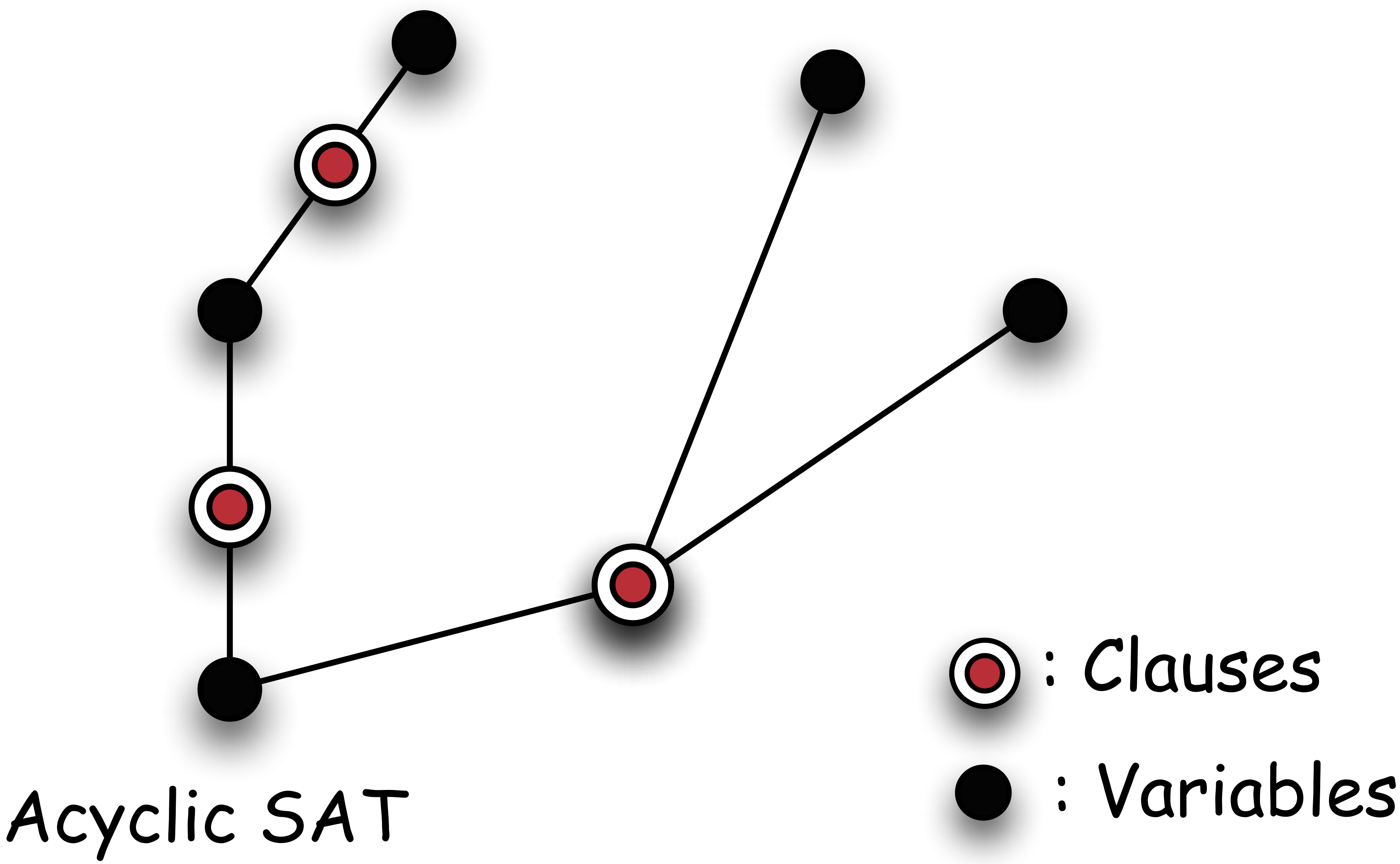
Backdoors to acyclic SAT

If the Incidence graph is a forest then SAT is in P (Fischer, Makowsky, Ravve 2008).



Backdoors to acyclic SAT

What about formulas with small backdoors to Acyclic SAT?
Is SAT tractable on these formulas?



Backdoors to acyclic SAT

Gaspers and Szeider (ICALP 2012) :

- weak backdoor detection to acyclic SAT is W[2]-hard.
- weak backdoor detection to acyclic 3-SAT is FPT.
- strong backdoor detection to acyclic SAT is FPT-approximable.

Backdoors to acyclic SAT

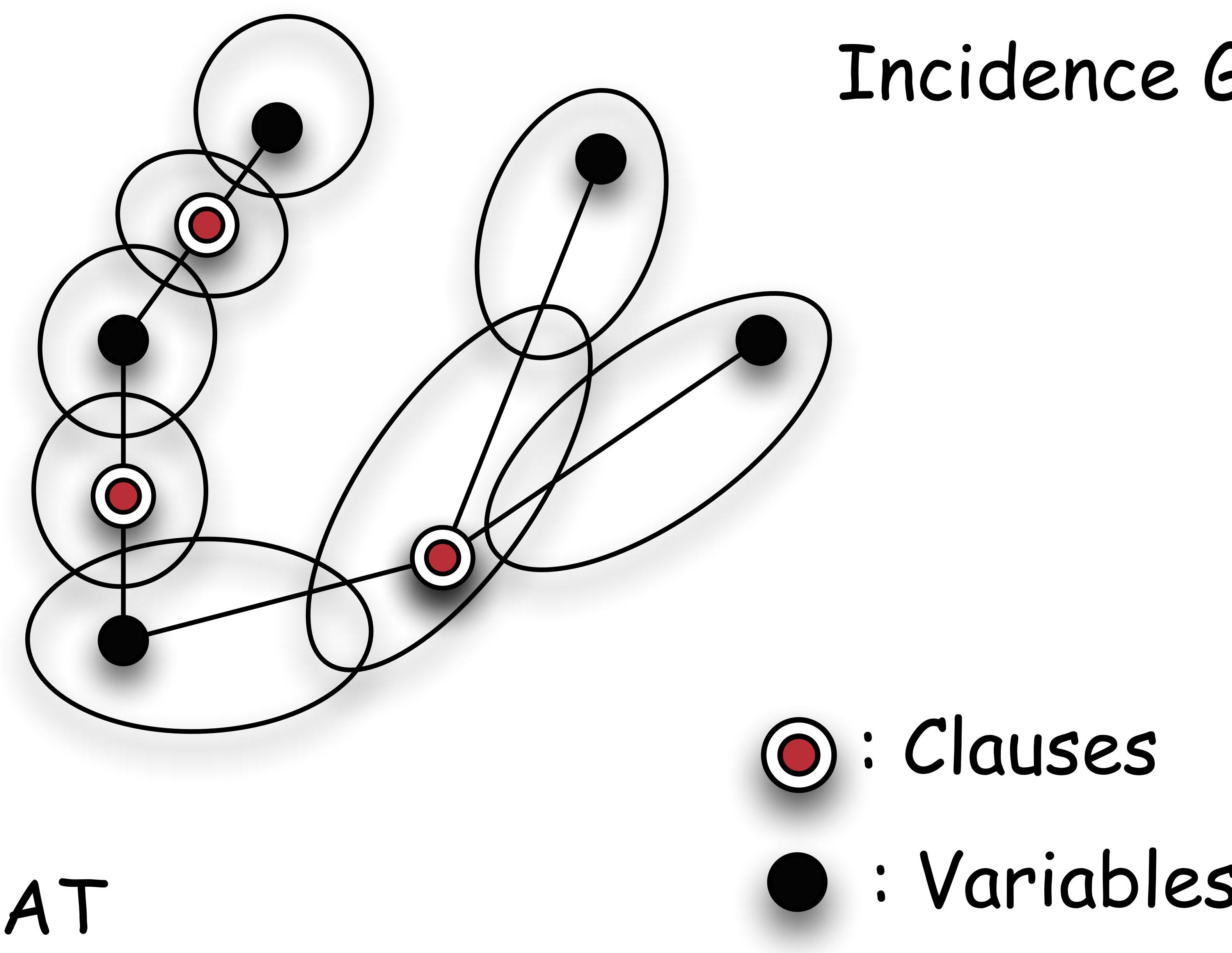
Gaspers and Szeider (ICALP 2012) :

- weak backdoor detection to acyclic SAT is W[2]-hard.
- weak backdoor detection to acyclic 3-SAT is FPT.
- strong backdoor detection to acyclic SAT is FPT-approximable.

In FPT time, either conclude there is no strong backdoor of size k or compute a strong backdoor of size 2^k

Backdoors to bounded tw SAT

If the Incidence graph is tree-like then SAT is in P (Fischer, Makowsky, Ravve 2008).



tw-SAT

Backdoors to bounded tw SAT

Gaspers and Szeider (FOCS 2013) :

Strong backdoor detection to tw SAT is FPT-approximable.

Backdoors to bounded tw SAT

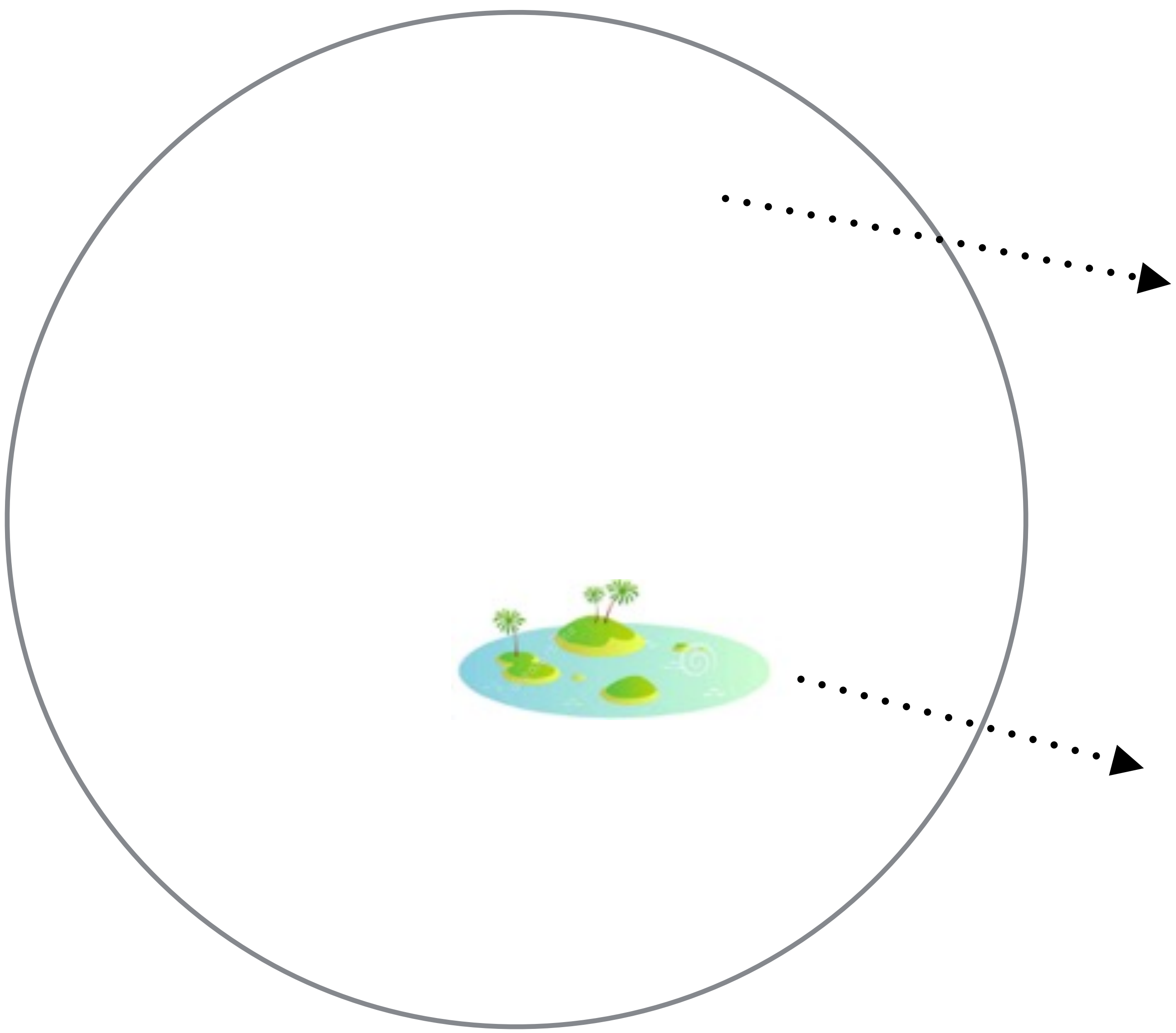
Gaspers and Szeider (FOCS 2013) :

Strong backdoor detection to tw SAT is FPT-approximable.

SAT parameterized by size of sbd to tw SAT is FPT.

Running time : $2^{2^{\wedge k}} n^3$

Backdoors to bounded tw SAT



SAT is in P for instances
with sbd of size
O(log log n) to tw SAT.

tw SAT is in P.

Backdoors to bounded tw SAT

Fomin, Lokshtanov, Misra, R., Saurabh (SODA 2015)

Backdoors to bounded tw SAT

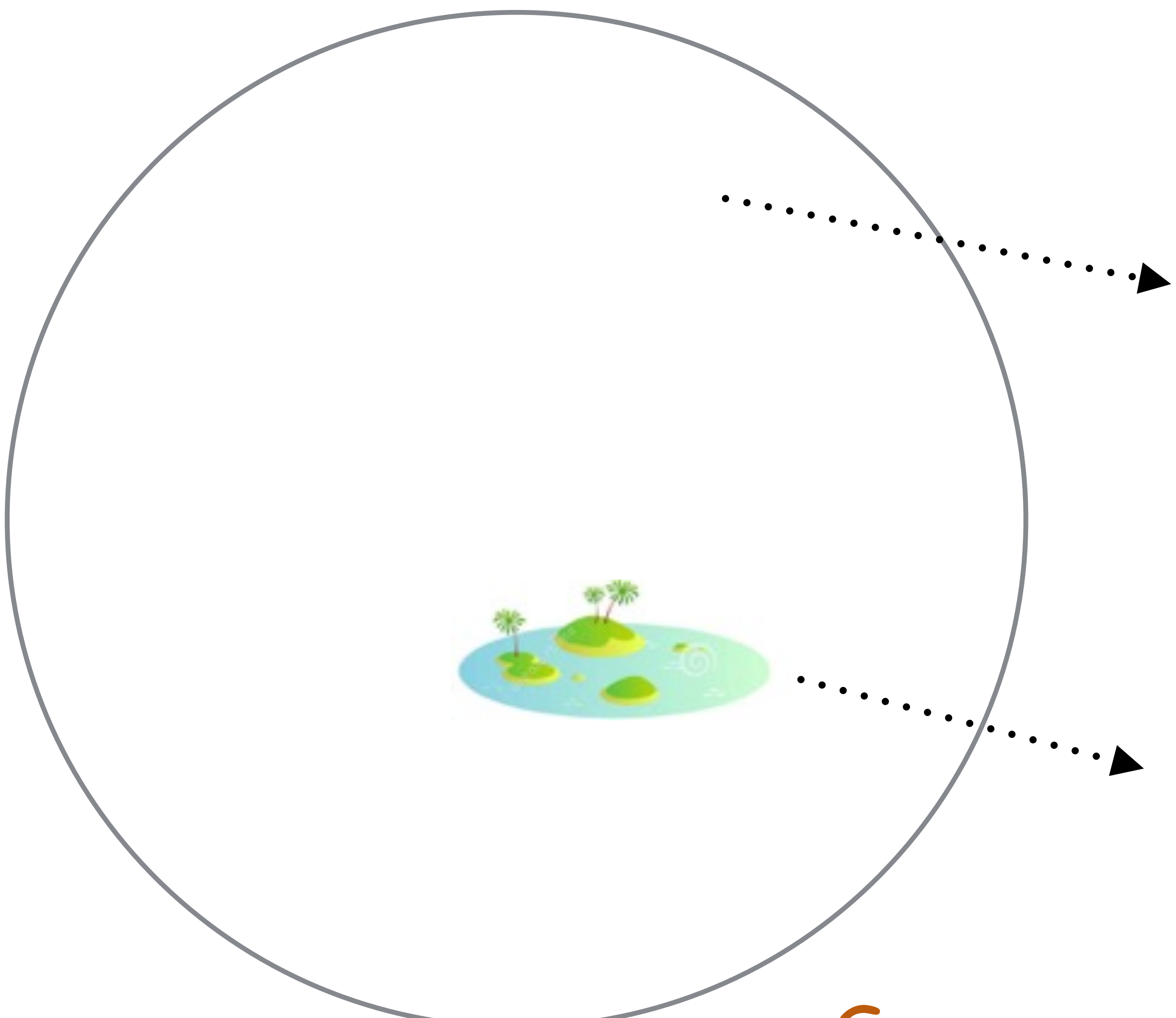
Fomin, Lokshtanov, Misra, R., Saurabh (SODA 2015)

3-SAT parameterized by k=min{sbd,wbd} to tw 3-SAT can be solved in time $2^{O(k)}$ m.

This running time is optimal both w.r.t parameter and input-size.

Backdoors to bounded tw SAT

This region cannot be extended.



3-SAT is in P for instances with a s/w backdoor of size O(log n) to tw 3-SAT.

tw 3-SAT is in P.

Some new features in this algorithm!

Combining the perspectives

Combining the perspectives

- Williams et al. proposed that SAT solvers encounter backdoor sets without actually searching for them.

Combining the perspectives

- Williams et al. proposed that SAT solvers encounter backdoor sets without actually searching for them.

- This algorithm: revisit this perspective.

Backdoors to bounded tw SAT

DPLL

Backdoors to bounded tw SAT

DPLL

- Apply UP and PLE

Backdoors to bounded tw SAT

DPLL

- Apply UP and PLE
- Select a variable x u.a.r

Backdoors to bounded tw SAT

DPLL

- Apply UP and PLE
- Select a variable x u.a.r
- Branch on x

Backdoors to bounded tw SAT

DPLL DPLL'

- Apply UP and PLE
- Select a variable x u.a.r
- Branch on x

Backdoors to bounded tw SAT

DPLL

- Apply UP and PLE
- Select a variable x u.a.r
- Branch on x

DPLL'

- If formula has constant tw, then solve in poly time.

Backdoors to bounded tw SAT

DPLL	DPLL'
	• If formula has constant tw, then solve in poly time.
• Apply UP and PLE	• Reduce all \`protrusions'.
• Select a variable x u.a.r	
• Branch on x	

Backdoors to bounded tw SAT

DPLL

- Apply UP and PLE
- Select a variable x u.a.r
- Branch on x

DPLL'

- If formula has constant tw, then solve in poly time.
- Reduce all `protrusions'.
- Select a variable x u.a.r

Backdoors to bounded tw SAT

DPLL	DPLL'
	• If formula has constant tw, then solve in poly time.
• Apply UP and PLE	• Reduce all `protrusions'.
• Select a variable x u.a.r	• Select a variable x u.a.r
• Branch on x	• Branch on x

Backdoors to bounded tw SAT

DPLL	DPLL′
	• If formula has constant tw, then solve in poly time.
• Apply UP and PLE	• Reduce all \`protrusions′.
• Select a variable x u.a.r	• Select a variable x u.a.r
• Branch on x	• Branch on x

DPLL′ is an FPT algorithm for 3-SAT par by min{sbd,wbd} to tw 3-SAT.

Backdoors to bounded tw SAT

Backdoors to bounded tw SAT

- Protrusion replacement takes the place of UP and PLE.

Backdoors to bounded tw SAT

- Protrusion replacement takes the place of UP and PLE.
- Since the base class is more complex, the preprocessing is also involved.

Backdoors to bounded tw SAT

- Protrusion replacement takes the place of UP and PLE.
- Since the base class is more complex, the preprocessing is also involved.
- But intuition remains the same: Remove 'irrelevant' parts of the formula or at the very least replace them with a `small' equivalent formula.

Backdoors to bounded tw SAT

Backdoors to bounded tw SAT

- First FPT algorithm for SAT which does not depend on computing a backdoor set first.

Backdoors to bounded tw SAT

- First FPT algorithm for SAT which does not depend on computing a backdoor set first.

- Optimal running time (parameter and i/p size)

Backdoors to bounded tw SAT

- First FPT algorithm for SAT which does not depend on computing a backdoor set first.

- Optimal running time (parameter and i/p size)

- Again, techniques developed here have other applications: improving several kernelization and FPT algorithms to linear time.

Composite Base Classes

Heterogenous backdoors

Heterogenous backdoors

Consider the following formula.

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2 \ldots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2\ldots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

What is the size of a smallest strong backdoor set into Horn?

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2 \ldots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

What is the size of a smallest strong backdoor set into Horn?

at least n

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

What is the size of a smallest strong backdoor set into Horn?

at least n

What is the size of a smallest strong backdoor set into 2-cnf?

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2\ldots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

What is the size of a smallest strong backdoor set into Horn?

at least n

What is the size of a smallest strong backdoor set into 2-cnf?

at least n-1

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2\ldots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2 \dots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

Consider F[x=0] $\quad (\neg a_1 \vee \neg a_{2 \dots} \vee \neg a_n)$

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

Consider F[x=0] $(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n)$

Consider F[x=1] $(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$

Heterogenous backdoors

Consider the following formula.

$$(x \vee \neg a_1 \vee \neg a_{2 \ldots} \vee \neg a_n) \wedge$$
$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

Consider F[x=0] $(\neg a_1 \vee \neg a_{2 \ldots} \vee \neg a_n)$ Horn

Consider F[x=1] $(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$

Heterogenous backdoors

Consider the following formula.

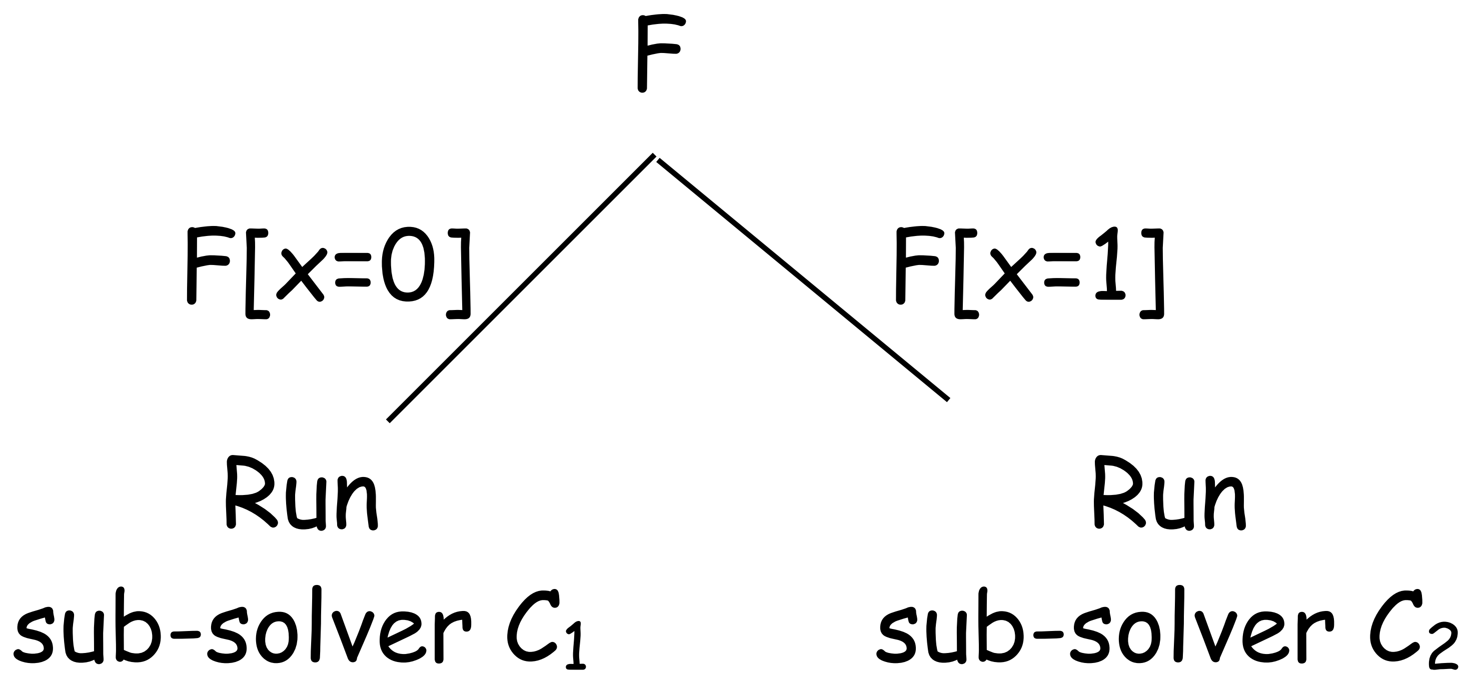
$$(x \lor \neg a_1 \lor \neg a_2 \ldots \lor \neg a_n) \land$$
$$(\neg x \lor b_1 \lor c_1) \land (\neg x \lor b_2 \lor c_2) .. \land (\neg x \lor b_n \lor c_n)$$

Consider F[x=0] $(\neg a_1 \lor \neg a_2 \ldots \lor \neg a_n)$ Horn

Consider F[x=1] $(b_1 \lor c_1) \land (b_2 \lor c_2) .. \land (b_n \lor c_n)$ 2-cnf

Heterogenous backdoors

C_1, C_2: Horn, 2-cnf



Heterogenous backdoors

Let $C_1, \ldots, C_r$ be islands of tractability.

X is a heterogenous backdoor into $C_1, \ldots, C_r$ if for every assignment of X, the reduced formula is in some C_i.

Heterogenous backdoors

Heterogenous backdoors

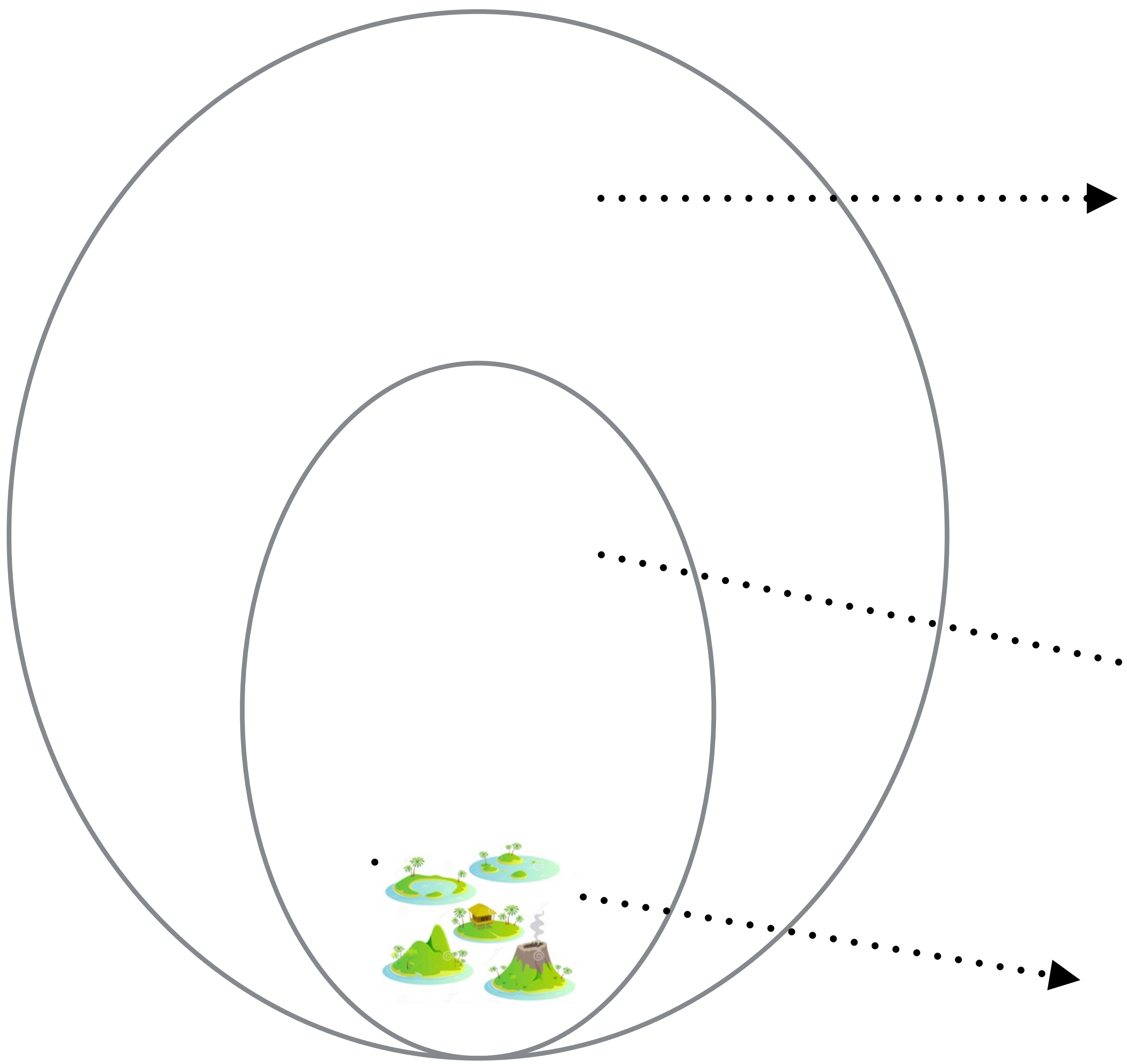
- Heterogenous backdoors can be arbitrarily smaller than normal strong backdoors.

Heterogenous backdoors

- Heterogenous backdoors can be arbitrarily smaller than normal strong backdoors.

- Class of instances with small heterogenous backdoors is a much larger class than instances with small strong backdoor.

Heterogenous backdoors



Heterogenous backdoor of size k to some Islands

Strong backdoor of size k to some Island of Tractability

Islands of Tractability

Heterogenous backdoors

Heterogenous backdoors

Gaspers, Ordyniak, Misra, Szeider, Zivny (AAAI 2014)

Heterogenous backdoors

Gaspers, Ordyniak, Misra, Szeider, Zivny (AAAI 2014)

1. If H =Horn/dual-Horn ∪ 2CNF then detecting heterogenous backdoors to H is FPT

2. For every other combination of Schaefer classes, detecting heterogenous backdoors to H is W[2]-hard.

but FPT for 3-cnf formulas

Archipelagos of tractability

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_{2\ldots} \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_{2\ldots} \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

What is the size of a smallest heterogenous backdoor set into Horn $\cup$ 2-cnf?

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

What is the size of a smallest heterogenous backdoor set into Horn $\cup$ 2-cnf?

at least 2n

Archipelagos of tractability

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_{2 \ldots} \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_{2 \ldots} \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider $F[x=0]$

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$

$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider $F[x=0]$

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$
$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider $F[x=1]$

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$
$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$
$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$
$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

Horn

$$\boxed{(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n)} \wedge$$

$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$

$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider $F[x=0]$

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$

$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider $F[x=1]$

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$

$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$

$$\boxed{(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)}$$ 2-cnf

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$

$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$
$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$
$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$

$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Horn

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$

$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$

$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$

$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$

$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$

$$\boxed{(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)}$$ 2-cnf

Archipelagos of tractability

$$(x \vee \neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge (\neg x \vee \neg p_1 \vee \neg p_2 \ldots \vee \neg p_n)$$

$$\wedge$$

$$(\neg x \vee b_1 \vee c_1) \wedge (\neg x \vee b_2 \vee c_2) .. \wedge (\neg x \vee b_n \vee c_n)$$

$$\wedge$$

$$(x \vee q_1 \vee r_1) \wedge (x \vee q_2 \vee r_2) .. \wedge (x \vee q_n \vee r_n)$$

Consider F[x=0]

$$(\neg a_1 \vee \neg a_2 \ldots \vee \neg a_n) \wedge$$
$$(q_1 \vee r_1) \wedge (q_2 \vee r_2) .. \wedge (q_n \vee r_n)$$

Consider F[x=1]

$$(\neg p_1 \vee \neg p_2 \ldots \vee \neg p_n) \wedge$$
$$(b_1 \vee c_1) \wedge (b_2 \vee c_2) .. \wedge (b_n \vee c_n)$$

Archipelagos of tractability

C_1, C_2 :   Horn,2-cnf

F

F[x=0]   F[x=1]

Run appropriate sub-solver C_i on each part variable-disjoint from the rest

Run appropriate sub-solver C_i on each part variable-disjoint from the rest

Split backdoors

Let $C_1, \ldots, C_r$ be islands of tractability.

X is a **split backdoor** into $C_1, \ldots, C_r$ if for every assignment of X, every connected component of the reduced formula is in some C_i.

Split backdoors

Let $C_1,\dots,C_r$ be islands of tractability.

X is a split backdoor into $C_1,\dots,C_r$ if for every assignment of X, every connected component of the reduced formula is in some C_i.

A minimal set of clauses which is variable-disjoint from the remaining clauses.

Split backdoors

Split backdoors

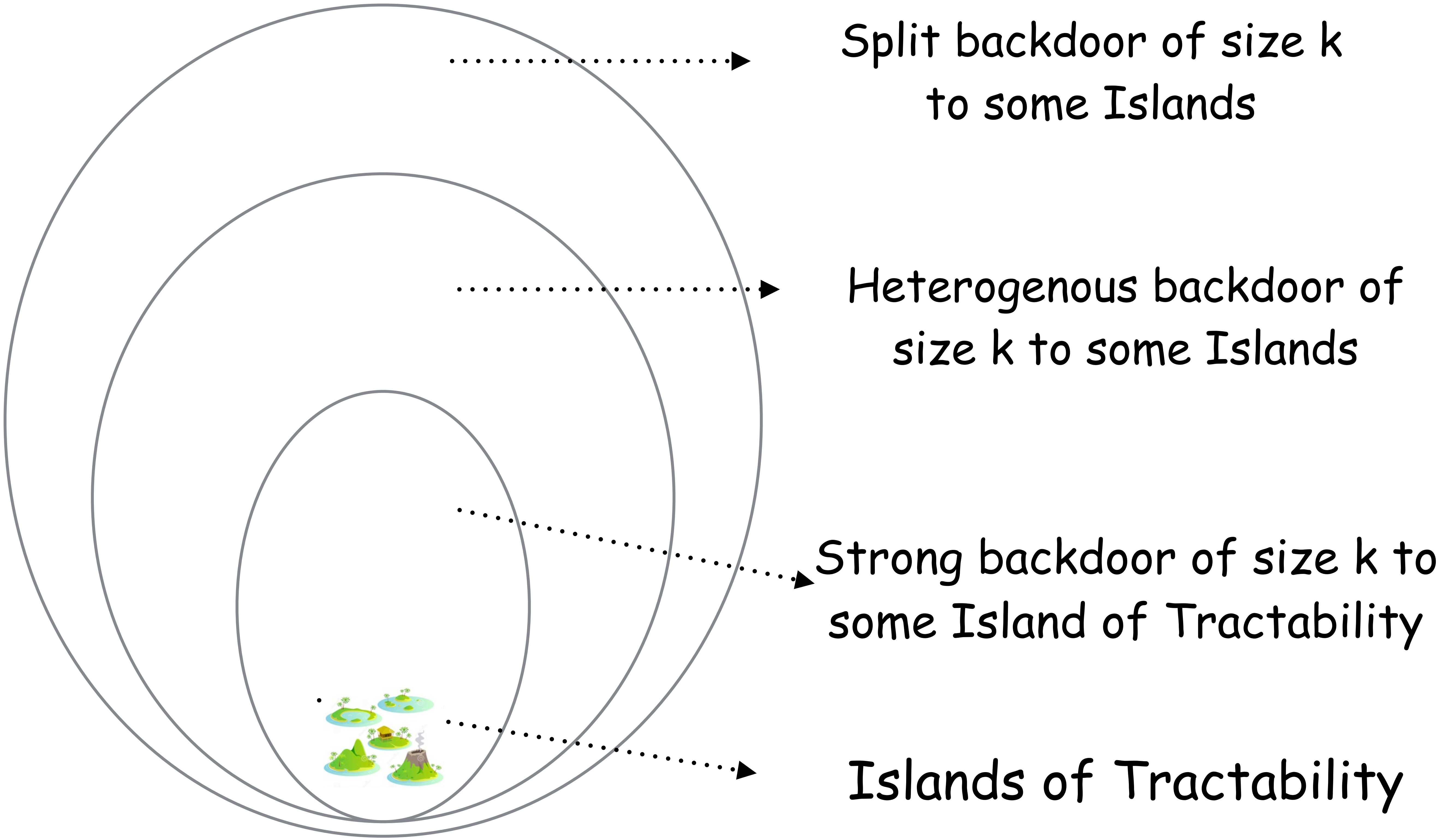
- Split backdoors can be arbitrarily smaller than heterogenous backdoors.

Split backdoors

- Split backdoors can be arbitrarily smaller than heterogenous backdoors.

- Class of instances with small split backdoors is a much larger class than class of instances with small heterogenous backdoor.

Split backdoors



Split backdoors

Ganian, R., Szeider (2014):

If H is a finite set of finite constraint languages, then detecting split-backdoors of the given CSP to H is FPT.

Builds on a combination of traditional FPT tools and new graph separation tools like important separators, sequences and CSP based pattern replacements.

Summing up

Summing up

- We have seen how backdoors and fixed parameter tractability provide a framework to extend tractability results for SAT based on the `distance' of instances to islands of tractability.

Summing up

- We have seen how backdoors and fixed parameter tractability provide a framework to extend tractability results for SAT based on the `distance' of instances to islands of tractability.

- Stronger definitions of sub-solvers and base classes allowing us to prove tractability for larger classes of instances.

Summing up

- We have seen how backdoors and fixed parameter tractability provide a framework to extend tractability results for SAT based on the `distance' of instances to islands of tractability.

- Stronger definitions of sub-solvers and base classes allowing us to prove tractability for larger classes of instances.

- Several other variants of backdoors have been proposed, eg. backdoor trees (Samer and Szeider AAAI 2008), learning sensitive backdoors (Dilkina, Gomes, Sabharwal SAT 2009).

Future research

Future research

- So far backdoor sets and variants have provided the best and theoretically most robust explanation for the performances of SAT solvers.

Future research

- So far backdoor sets and variants have provided the best and theoretically most robust explanation for the performances of SAT solvers.
- What other structural properties of instances are correlated to the computation time and can be effectively formalized in theory?

Future research

Future research

- So far, `small' backdoors treated as certificates for closeness.
- Better measures than size?
- i.e. backdoors of potentially unbounded size but with some structure.

Future research

Future research

- Analysis of existing SAT algorithms in terms of FPT parameterized by backdoors.

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!

Thank you for your attention!