Relating Proof Complexity Measures and Practical Hardness of SAT

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

18th International Conference on Principles
and Practice of Constraint Programming
Québec City, Canada
October 8–12, 2012

Joint work with Matti Järvisalo, Arie Matsliah, and Stanislav Živný

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

SAT solving

- Enormous progress in performance last 10-15 years
- State-of-the-art solvers can deal with real-world instances with millions of variables
- But best solvers still based on methods from early 1960s
- Tiny formulas known that are totally beyond reach

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

SAT solving

- Enormous progress in performance last 10-15 years
- State-of-the-art solvers can deal with real-world instances with millions of variables
- But best solvers still based on methods from early 1960s
- Tiny formulas known that are totally beyond reach

What makes formulas hard or easy in practice for SAT solvers?

Proof Complexity and SAT Solving

Proof complexity

- Satisifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

SAT solving

- Enormous progress in performance last 10-15 years
- State-of-the-art solvers can deal with real-world instances with millions of variables
- But best solvers still based on methods from early 1960s
- Tiny formulas known that are totally beyond reach

What makes formulas hard or easy in practice for SAT solvers?

What (if anything) can proof complexity say about this?

Resolution Proof System

Refute unsatisfiable formulas in conjunctive normal form (CNF):

$$(x \lor z) \land (y \lor \overline{z}) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u})$$
$$\land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Resolution Proof System

Refute unsatisfiable formulas in conjunctive normal form (CNF):

$$(x \lor z) \land (y \lor \overline{z}) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u})$$
$$\land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Resolution rule:

$$\frac{B \lor x \quad C \lor \overline{x}}{B \lor C}$$

Resolution Proof System

Refute unsatisfiable formulas in conjunctive normal form (CNF):

$$(x \lor z) \land (y \lor \overline{z}) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u})$$
$$\land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Resolution rule:

$$\frac{B \lor x \quad C \lor \overline{x}}{B \lor C}$$

Observation

If F is a satisfiable CNF formula and D is derived from clauses $C_1, C_2 \in F$ by the resolution rule, then $F \land D$ is satisfiable.

Resolution Proof System

Refute unsatisfiable formulas in conjunctive normal form (CNF):

$$(x \lor z) \land (y \lor \overline{z}) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u})$$
$$\land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Resolution rule:

$$\frac{B \lor x \quad C \lor \overline{x}}{B \lor C}$$

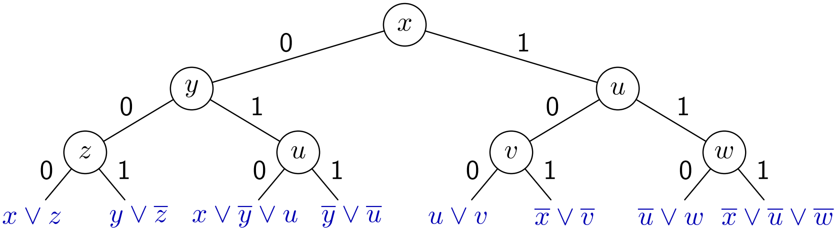
Observation

If F is a satisfiable CNF formula and D is derived from clauses $C_1, C_2 \in F$ by the resolution rule, then $F \land D$ is satisfiable.

So prove CNF formula unsatisfiable by deriving contradiction by resolution

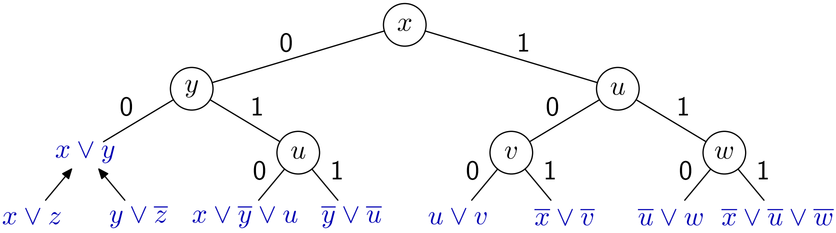
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



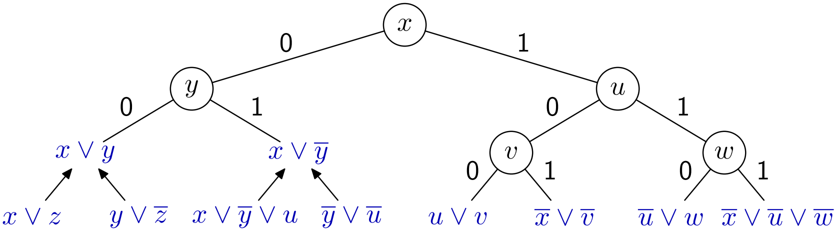
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



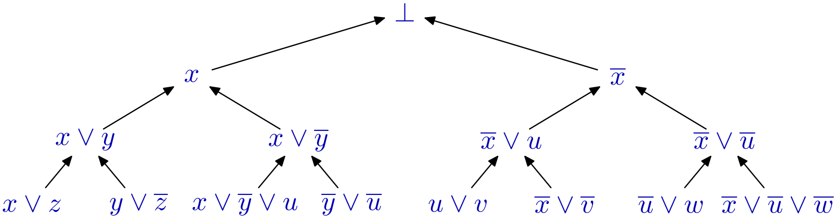
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



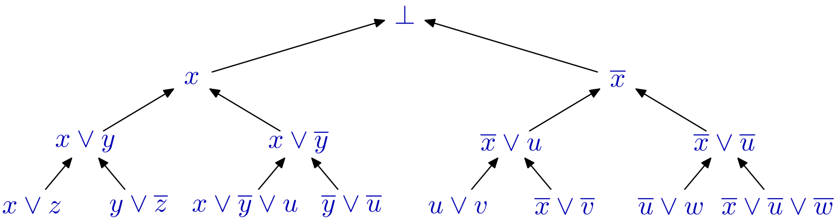
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



- Conflict-driven clause learning adds “shortcut edges” in tree
- But still yields resolution proof
- True also for (most) preprocessing techniques

Complexity Measures for Resolution

Let n = size of formula

Length

clauses in refutation — at most $\exp(n)$

Width

Size of largest clause in refutation — at most n

Space

Max # clauses one needs to remember when "verifying correctness of refutation on blackboard" — at most n (!)

Length

- Clearly lower bound on running time for any CDCL algorithm

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it
- In fact, probably intractable [Aleknovich & Razborov '01]

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it
- In fact, probably intractable [Aleknovich & Razborov '01]
- So small length upper bound might be much too optimistic

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it
- In fact, probably intractable [Aleknovich & Razborov '01]
- So small length upper bound might be much too optimistic
- Not the right measure of "hardness in practice"

Length vs. Width

- Searching for small width refutations known heuristic in AI community

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width $\Rightarrow$ small length (by counting)

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width $\Rightarrow$ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length
- Small width ⇒ CDCL solver will provably be fast
  [Atserias, Ficthe & Thurley '09]
  (but slighly idealized theoretical model)

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width $\Rightarrow$ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length
- Small width $\Rightarrow$ CDCL solver will provably be fast
  [Atserias, Ficthe & Thurley '09]
  (but slighly idealized theoretical model)
- Right hardness measure?

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?
- Space $\geq$ width [Atserias & Dalmau '03]

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?
- Space $\geq$ width [Atserias & Dalmau '03]
- But small width does not say **anything** about space [N. '06], [N. & Håstad '08], [Ben-Sasson & N. '08]

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?
- Space $\geq$ width [Atserias & Dalmau '03]
- But small width does not say **anything** about space
  [N. '06], [N. & Håstad '08], [Ben-Sasson & N. '08]
- So space stricter hardness measure than width
  (but space model even more idealized)

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs
- Proposed as practical measure of hardness of SAT instances in [Ansótegui, Bonet, Levy & Manyà '08]

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs
- Proposed as practical measure of hardness of SAT instances in [Ansótegui, Bonet, Levy & Manyà '08]
- Clearly tree-like space $\geq$ space but not known to be different

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs
- Proposed as practical measure of hardness of SAT instances in [Ansótegui, Bonet, Levy & Manyà '08]
- Clearly tree-like space $\geq$ space but not known to be different

This work can be viewed as implementing program outlined in [ABLM08]

Result 1: Separation of Space and Tree-like Space

We don't believe in tree-like space as hardness measure

- Tree-like space tightly connected with tree-like length
- Corresponds to DPLL without clause learning
- Would suggest CDCL doesn't buy you anything

Result 1: Separation of Space and Tree-like Space

We don't believe in tree-like space as hardness measure

- Tree-like space tightly connected with tree-like length
- Corresponds to DPLL without clause learning
- Would suggest CDCL doesn't buy you anything

We prove first asymptotic separation of space and tree-like space

Theorem

There are formulas requiring space $\mathcal{O}(1)$ for which tree-like space grows like $\Omega(\log n)$

Only constant-factor separation known before [Esteban & Torán '03]

Result 2: Small Backdoor Sets Imply Small Space

- Backdoor sets: practically motivated hardness measure
- First studied in [Williams, Gomes & Selman '03]
- Real-world SAT instances often have small backdoors

Result 2: Small Backdoor Sets Imply Small Space

- Backdoor sets: practically motivated hardness measure
- First studied in [Williams, Gomes & Selman '03]
- Real-world SAT instances often have small backdoors

We show connections between backdoors and space complexity (elaborating on [ABLM08])

Theorem (Informal)

If a formula has a small backdoor set, then it requires small space

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Run experiments on formulas with fixed complexity w.r.t. width (and length) but varying space*

- Is running time essentially the same?
- Or does it increase with increasing space?

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Run experiments on formulas with fixed complexity w.r.t. width (and length) but varying space*

- Is running time essentially the same?
- Or does it increase with increasing space?

Experimental results

Running times seem to correlate with space complexity**

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Run experiments on formulas with fixed complexity w.r.t. width (and length) but varying space*

- Is running time essentially the same?
- Or does it increase with increasing space?

Experimental results

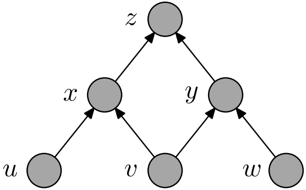
Running times seem to correlate with space complexity**

(*) But such formulas are nontrivial to find
(**) With some caveats to be discussed later

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

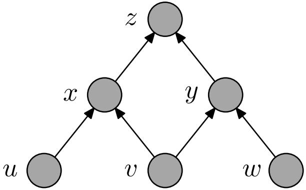


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

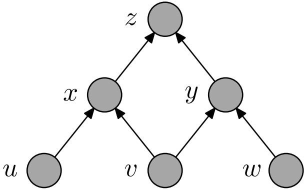


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

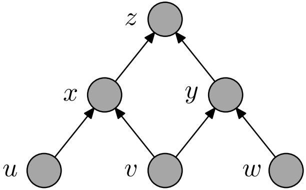


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas. . .

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

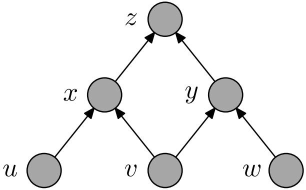


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas. . .

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$



- sources are true
- truth propagates upwards
- but sink is false

Extensive literature on pebbling time-space trade-offs from 1970s and 80s

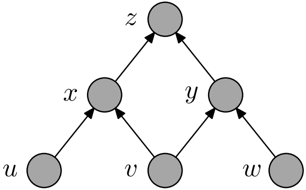
Pebbling formulas studied by [Bonet et al. '98, Raz & McKenzie '99, Ben-Sasson & Wigderson '99] and others

Hope that pebbling properties of DAG somehow carry over to resolution refutations of pebbling formulas.

Use Pebbling Formulas. . .

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$



- sources are true
- truth propagates upwards
- but sink is false

Extensive literature on pebbling time-space trade-offs from 1970s and 80s

Pebbling formulas studied by [Bonet et al. '98, Raz & McKenzie '99, Ben-Sasson & Wigderson '99] and others

Hope that pebbling properties of DAG somehow carry over to resolution refutations of pebbling formulas. **Except. . .**

...with Functions Substituted for Variables

Won't work — pebbling formulas solved by unit propagation, so supereasy

Make formula harder by substituting $x_1 \oplus x_2$ for every variable x (also works for other Boolean functions with "right" properties):

$$\overline{x} \lor y$$

$$\Downarrow$$

$$\neg(x_1 \oplus x_2) \lor (y_1 \oplus y_2)$$

$$\Downarrow$$

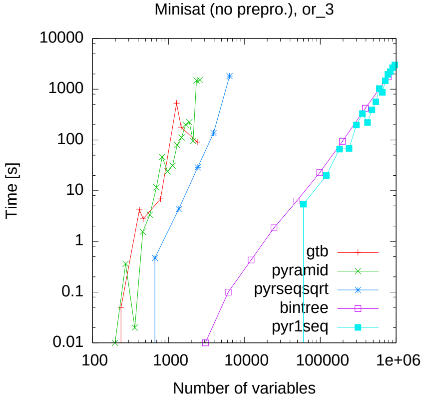
$$\begin{aligned} & (x_1 \lor \overline{x}_2 \lor y_1 \lor y_2) \\ \land\ & (x_1 \lor \overline{x}_2 \lor \overline{y}_1 \lor \overline{y}_2) \\ \land\ & (\overline{x}_1 \lor x_2 \lor y_1 \lor y_2) \\ \land\ & (\overline{x}_1 \lor x_2 \lor \overline{y}_1 \lor \overline{y}_2) \end{aligned}$$

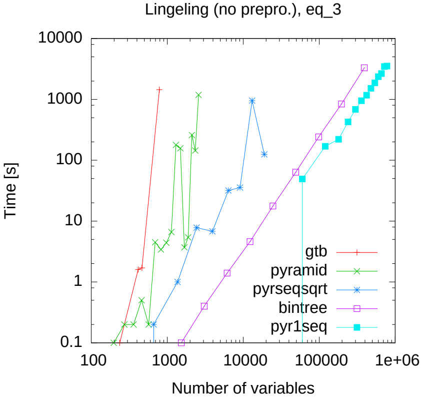
Now CNF formula inherits pebbling graph properties!

About the Experiments

- 12 graph families with varying space complexity
- 8 different substitution functions
- Total of 96 formula families with around 50 instances per family
- CDCL solvers Minisat 2.2.0 and Lingeling version 774
- Experiments
  - with and without preprocessing
  - with and without random shuffling of clauses and variables
- Intel Core i5-2500 3.3-GHz quad-core CPU with 8 GB of memory
- Time-out 1 hour per instance
- Massive amounts of data...

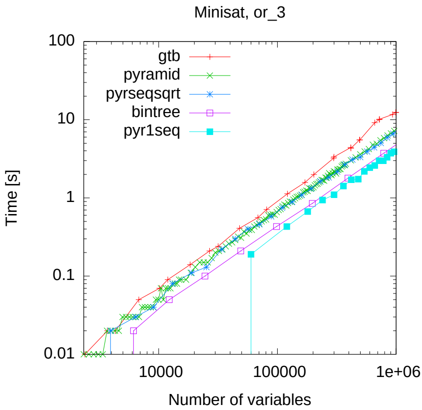
Example Results Without Preprocessing

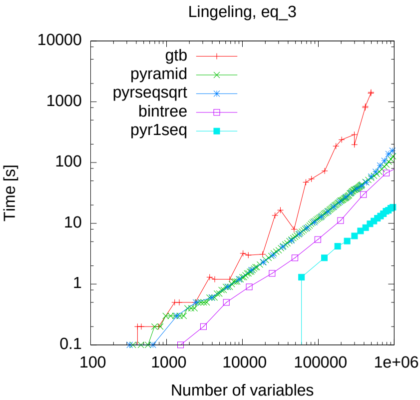




Looks nice. . . Easy formulas solved fast and hard formulas take longer time

Example Results with Preprocessing





Less nice... Which is not surprising

Caveats and Issues

Preprocessing dampens correlations

- To be expected — space of proof not captured during preprocessing
- By construction formulas amenable to preprocessing

Caveats and Issues

Preprocessing dampens correlations

- To be expected — space of proof not captured during preprocessing
- By construction formulas amenable to preprocessing

Artificial benchmarks

- True, but the only formulas where we know how to control space
- In general, computing space complexity probably PSPACE-complete

Caveats and Issues

Preprocessing dampens correlations

- To be expected — space of proof not captured during preprocessing
- By construction formulas amenable to preprocessing

Artificial benchmarks

- True, but the only formulas where we know how to control space
- In general, computing space complexity probably PSPACE-complete

Theory vs. practice

- In theory all substitution functions equal — not so in practice
- In theory graph pebbling space all that matters — but many source vertices make binary tree formulas "too easy"

Caveats and Issues

Preprocessing dampens correlations

- To be expected — space of proof not captured during preprocessing
- By construction formulas amenable to preprocessing

Artificial benchmarks

- True, but the only formulas where we know how to control space
- In general, computing space complexity probably PSPACE-complete

Theory vs. practice

- In theory all substitution functions equal — not so in practice
- In theory graph pebbling space all that matters — but many source vertices make binary tree formulas "too easy"

Varying width and space independently would be more convincing

- Very true, but provably impossible since space $\geq$ width
- Want to see if space is "more fine-grained" hardness indicator

Summing up

- Modern CDCL SAT solvers amazingly successful in practice
- But poorly understood which formulas are easy or hard
- We propose space complexity as a measure of hardness in practice
- Don't claim conclusive evidence, but nontrivial correlations
- Would like to get similar results also with preprocessing
- Would like to study if theoretical time-space trade-offs show up in practice
- Believe there are more connections between proof complexity and SAT solving worth exploring

Thank you for your attention!