Seeking Practical CDCL Insights from Theoretical SAT Benchmarks

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

27th International Joint Conference on Artificial Intelligence
23rd European Conference on Artificial Intelligence
July 17, 2018

Joint work with Jan Elffers, Jesús Giráldez Cru, Stephan Gocht, and Laurent Simon

The Satisfiability Problem (SAT)

$$(x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z})$$

- Variables should be set to true or false
- Constraint $(x \vee \overline{y} \vee z)$: means x or z should be true or y false
- $\wedge$ means all constraints should hold simultaneously
- Is there a truth value assignment satisfying this?

The Satisfiability Problem (SAT)

$$(x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z})$$

- Variables should be set to true or false
- Constraint $(x \vee \overline{y} \vee z)$: means x or z should be true or y false
- $\wedge$ means all constraints should hold simultaneously
- Is there a truth value assignment satisfying this?

Surprisingly rich formalism for expressing many real-world applied problems

The Unreasonable Effectiveness of SAT Solvers

- SAT is NP-complete [Coo71] and so should be exponentially hard

The Unreasonable Effectiveness of SAT Solvers

- SAT is NP-complete [Coo71] and so should be exponentially hard
- But modern SAT solvers using conflict-driven clause learning (CDCL) [MS96, BS97] solve instances with millions of variables

The Unreasonable Effectiveness of SAT Solvers

- SAT is NP-complete [Coo71] and so should be exponentially hard
- But modern SAT solvers using conflict-driven clause learning (CDCL) [MS96, BS97] solve instances with millions of variables
- Lots of smart engineering to make it fly in practice [MMZ$^+$01, ES04, PD07, AS09, ...]

The Unreasonable Effectiveness of SAT Solvers

- SAT is NP-complete [Coo71] and so should be exponentially hard
- But modern SAT solvers using conflict-driven clause learning (CDCL) [MS96, BS97] solve instances with millions of variables
- Lots of smart engineering to make it fly in practice [MMZ$^+$01, ES04, PD07, AS09, ...]
- ... And a somewhat bewildering alphabet soup of heuristics (VSIDS, 1UIP, LBD, BCD, BCE, BVA, ELS, FLP, VE, VMTF, ...)

The Unreasonable Effectiveness of SAT Solvers

- SAT is NP-complete [Coo71] and so should be exponentially hard
- But modern SAT solvers using conflict-driven clause learning (CDCL) [MS96, BS97] solve instances with millions of variables
- Lots of smart engineering to make it fly in practice [MMZ$^+$01, ES04, PD07, AS09, ...]
- ... And a somewhat bewildering alphabet soup of heuristics (VSIDS, 1UIP, LBD, BCD, BCE, BVA, ELS, FLP, VE, VMTF, ...)

Want a **deeper understanding** of how these solvers actually work

How to Analyse Behaviour of CDCL Solvers?

Applied approach

- Instrument solver with combinations of heuristics
- Run on SAT competition benchmarks
- Few and heterogeneous benchmarks
  - Poorly understood properties
  - Isolated data points
- Some work in [LM02, KSM11, SM11] — hard to draw conclusions

How to Analyse Behaviour of CDCL Solvers?

Applied approach

- Instrument solver with combinations of heuristics
- Run on SAT competition benchmarks
- Few and heterogeneous benchmarks
  - Poorly understood properties
  - Isolated data points
- Some work in [LM02, KSM11, SM11] — hard to draw conclusions

Theoretical approach

- CDCL solvers search for proofs in resolution proof system
- Relate CDCL performance to proof complexity measures?
- Only considers existence of proofs, not algorithmic search
- Only asymptotic results (sometimes for gigantic formulas)
- Papers [JMNŽ12, MN14] generated more questions than answers...

Why Not Combine Theory and Practice?

Our combined approach

- Choose interesting formulas from proof complexity literature
  $\Rightarrow$ Well-understood properties

Why Not Combine Theory and Practice?

Our combined approach

- Choose interesting formulas from proof complexity literature
  $\Rightarrow$ Well-understood properties
- Tune to get easy benchmarks (but potentially tricky)
  $\Rightarrow$ Measure CDCL proof search quality

Why Not Combine Theory and Practice?

Our combined approach

- Choose interesting formulas from proof complexity literature
  $\Rightarrow$ Well-understood properties
- Tune to get easy benchmarks (but potentially tricky)
  $\Rightarrow$ Measure CDCL proof search quality
- Generate families with scaled instances of "same problem"
  $\Rightarrow$ Study not isolated data points but asymptotic behaviour

Why Not Combine Theory and Practice?

Our combined approach

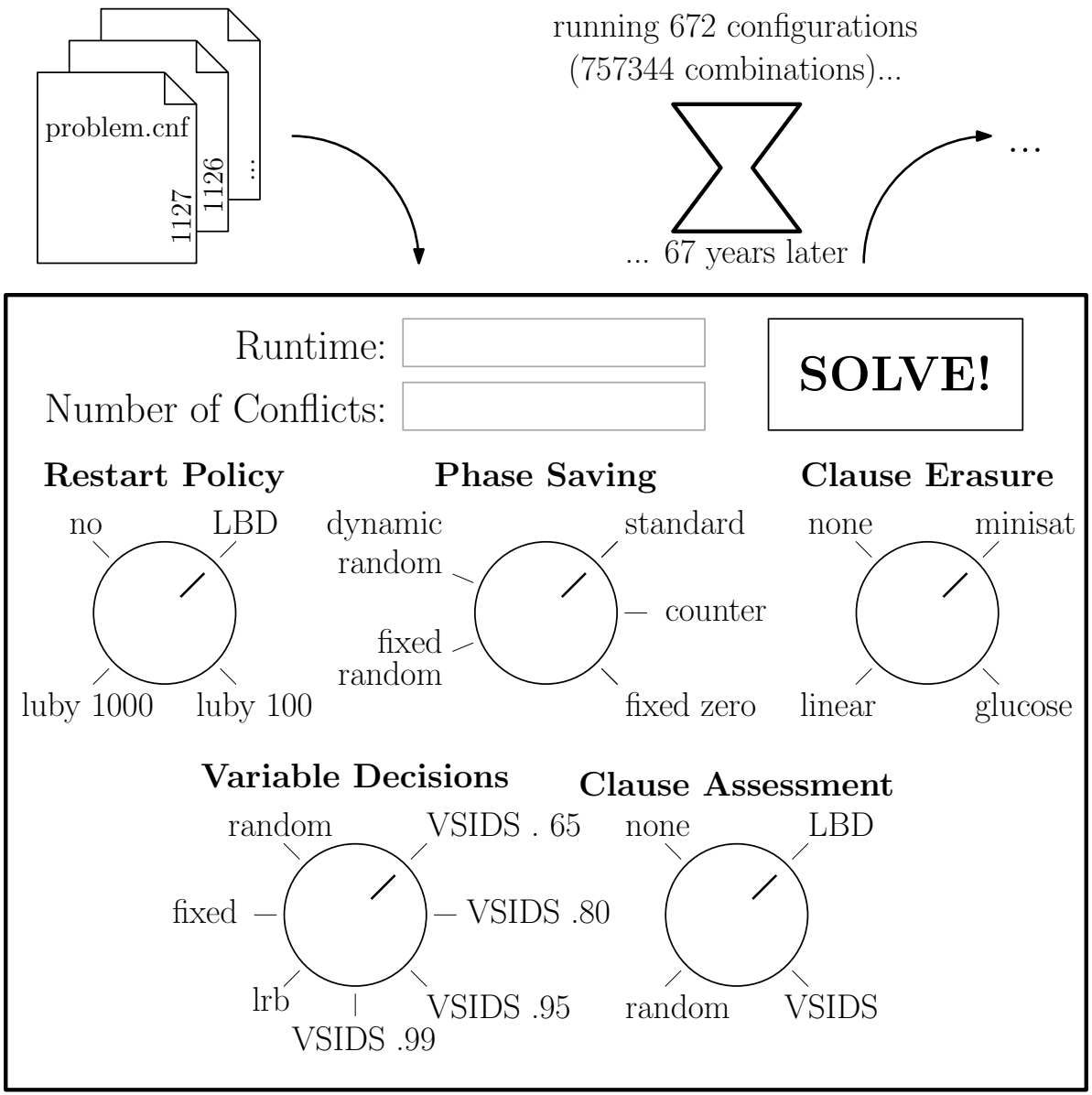
- Choose interesting formulas from proof complexity literature
  $\Rightarrow$ Well-understood properties
- Tune to get easy benchmarks (but potentially tricky)
  $\Rightarrow$ Measure CDCL proof search quality
- Generate families with scaled instances of "same problem"
  $\Rightarrow$ Study not isolated data points but asymptotic behaviour
- Extremal benchmarks w.r.t. different properties
  $\Rightarrow$ Different benchmarks will "stress-test" different heuristics

Why Not Combine Theory and Practice?

Our combined approach

- Choose interesting formulas from proof complexity literature
  $\Rightarrow$ Well-understood properties
- Tune to get easy benchmarks (but potentially tricky)
  $\Rightarrow$ Measure CDCL proof search quality
- Generate families with scaled instances of "same problem"
  $\Rightarrow$ Study not isolated data points but asymptotic behaviour
- Extremal benchmarks w.r.t. different properties
  $\Rightarrow$ Different benchmarks will "stress-test" different heuristics
- Code up instrumented solver with wide selection of heuristics
  - More complicated than it sounds — some heuristics tightly integrated
  - Run on (essentially full) cross product of heuristics
  - Which heuristics are important when?
  - How do heuristics interact?

Our Set-up


problem.cnf
1127
1126
...
running 672 configurations
(757344 combinations)...
... 67 years later
...
Runtime:
Number of Conflicts:
SOLVE!
Restart Policy
no
LBD
luby 1000
luby 100
Phase Saving
dynamic
standard
random
counter
fixed
random
fixed zero
Clause Erasure
none
minisat
linear
glucose
Variable Decisions
random
VSIDS . 65
fixed
VSIDS .80
lrb
VSIDS .95
VSIDS .99
Clause Assessment
none
LBD
random
VSIDS

. . . And the Results. . .

Experiments

- 1127 instances in 27 formula families
- 672 solver configurations
- More than 500,000 hours (67 years) of computations

. . . And the Results. . .

Experiments

- 1127 instances in 27 formula families
- 672 solver configurations
- More than 500,000 hours (67 years) of computations

Analysis?

- Huge amounts of data. . . How to even get an overview?
- How to make sure results are significant?
- Solvers deterministic — standard statistic tools don't seem to apply

. . . And the Results. . .

Experiments

- 1127 instances in 27 formula families
- 672 solver configurations
- More than 500,000 hours (67 years) of computations

Analysis? (Topic for separate talk)

- Huge amounts of data. . . How to even get an overview?
- How to make sure results are significant?
- Solvers deterministic — standard statistic tools don't seem to apply

. . . And the Results. . .

Experiments

- 1127 instances in 27 formula families
- 672 solver configurations
- More than 500,000 hours (67 years) of computations

Analysis? (Topic for separate talk)

- Huge amounts of data. . . How to even get an overview?
- How to make sure results are significant?
- Solvers deterministic — standard statistic tools don't seem to apply

Rest of this talk: some example findings

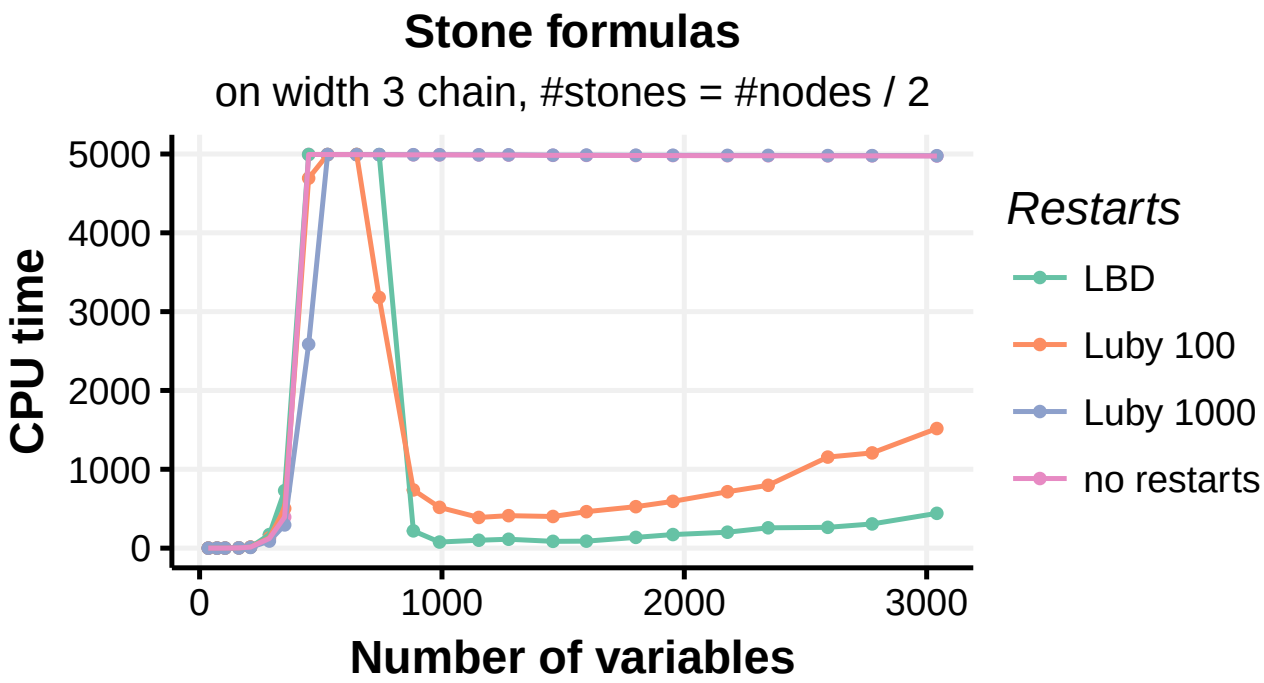
- Restarts
- Memory management
- Branching heuristic

Restarts

- Fast restarts crucial for CDCL solvers in practice
- Also appear in proofs that CDCL searches efficiently (kind of) in resolution [AFT11, PD11]
- But do restarts increase theoretical reasoning power? Open...
- Run experiments on benchmarks where "full power of resolution" needed [AJPU07] to gather "circumstantial evidence"?

Restarts

- Fast restarts crucial for CDCL solvers in practice
- Also appear in proofs that CDCL searches efficiently (kind of) in resolution [AFT11, PD11]
- But do restarts increase theoretical reasoning power? Open...
- Run experiments on benchmarks where "full power of resolution" needed [AJPU07] to gather "circumstantial evidence"?

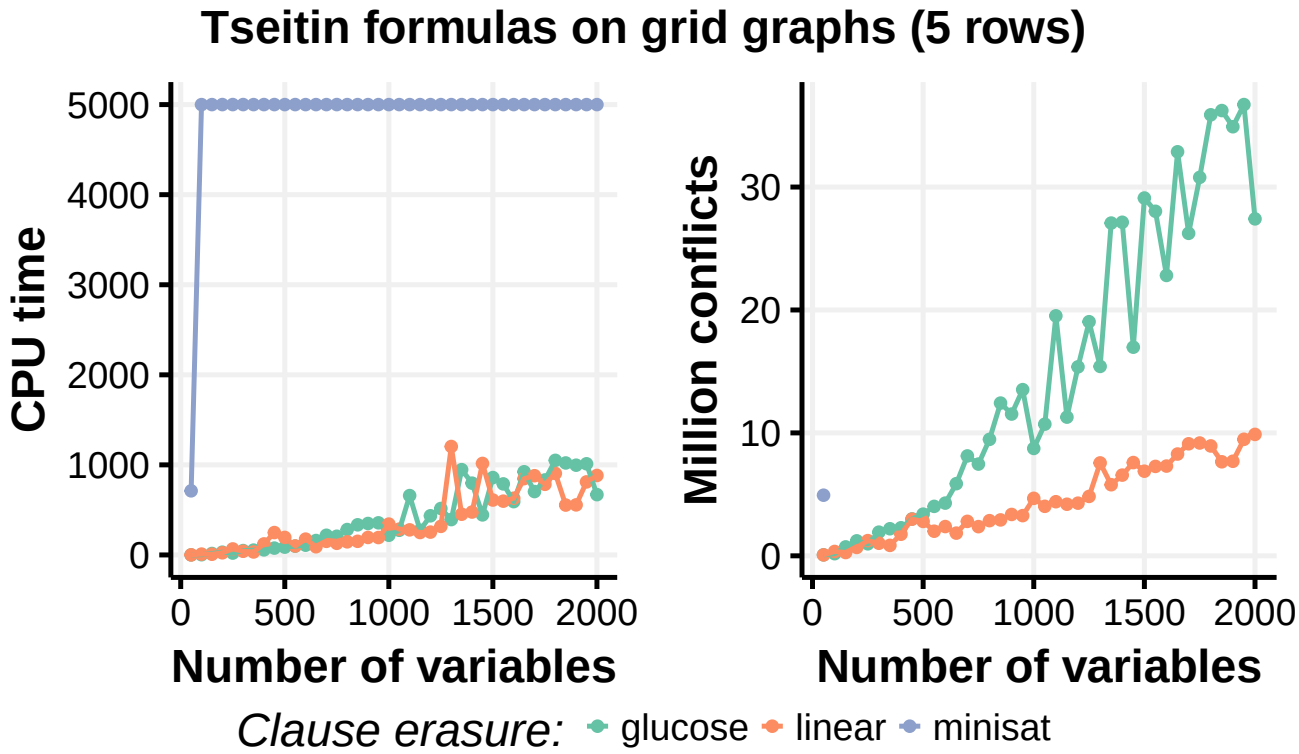


Memory Management: Time-Space Trade-offs

- CDCL solvers very aggressively minimize memory usage
- Dramatic time-space trade-offs in theory [BN11, BBI12, BNT13]
- Could this happen in practice?

Memory Management: Time-Space Trade-offs

- CDCL solvers very aggressively minimize memory usage
- Dramatic time-space trade-offs in theory [BN11, BBI12, BNT13]
- Could this happen in practice?



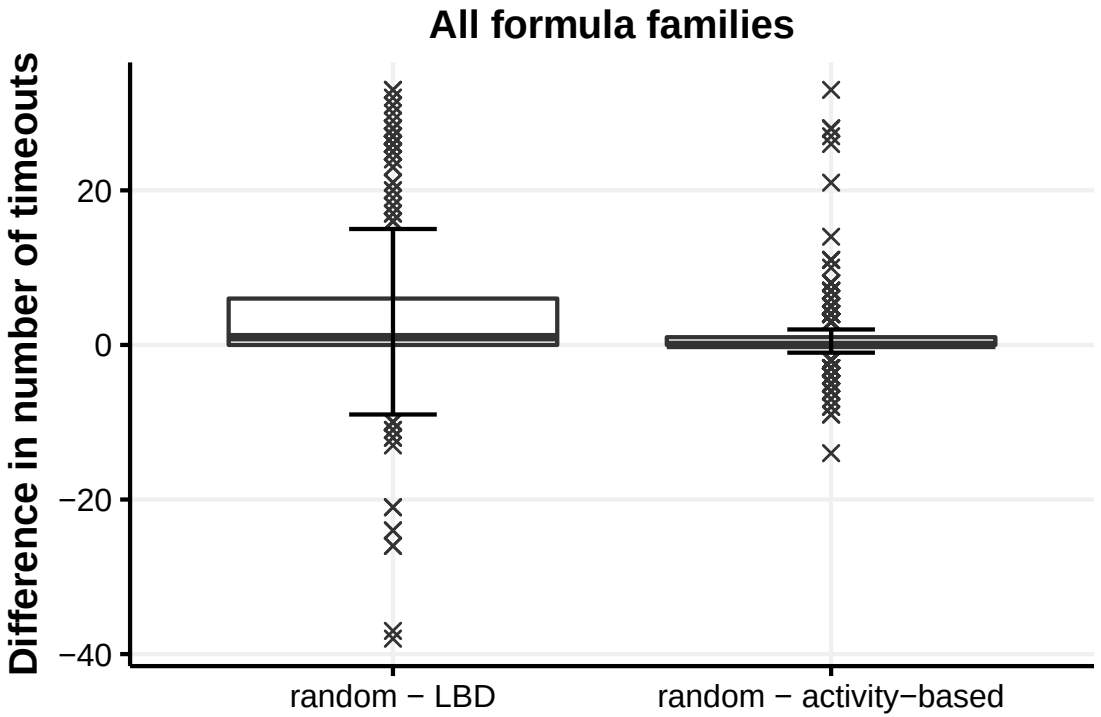
Database size: minisat $\left(\sim N^{0.25}\right)$ < glucose $\left(\sim N^{0.5}\right)$ < linear $\left(\sim N\right)$ [$N = \#\text{conflicts}$]

Memory Management: Clause Assessment

- When time to purge database, how to identify useful clauses to keep?
  - Activity-based [ES04]
  - Literal block distance (LBD) [AS09]
  - Random (control)

Memory Management: Clause Assessment

- When time to purge database, how to identify useful clauses to keep?
  - Activity-based [ES04]
  - Literal block distance (LBD) [AS09]
  - Random (control)
- LBD quite successful, especially when space is getting tight
- Activity-based indistinguishable from random (& random OK-ish)

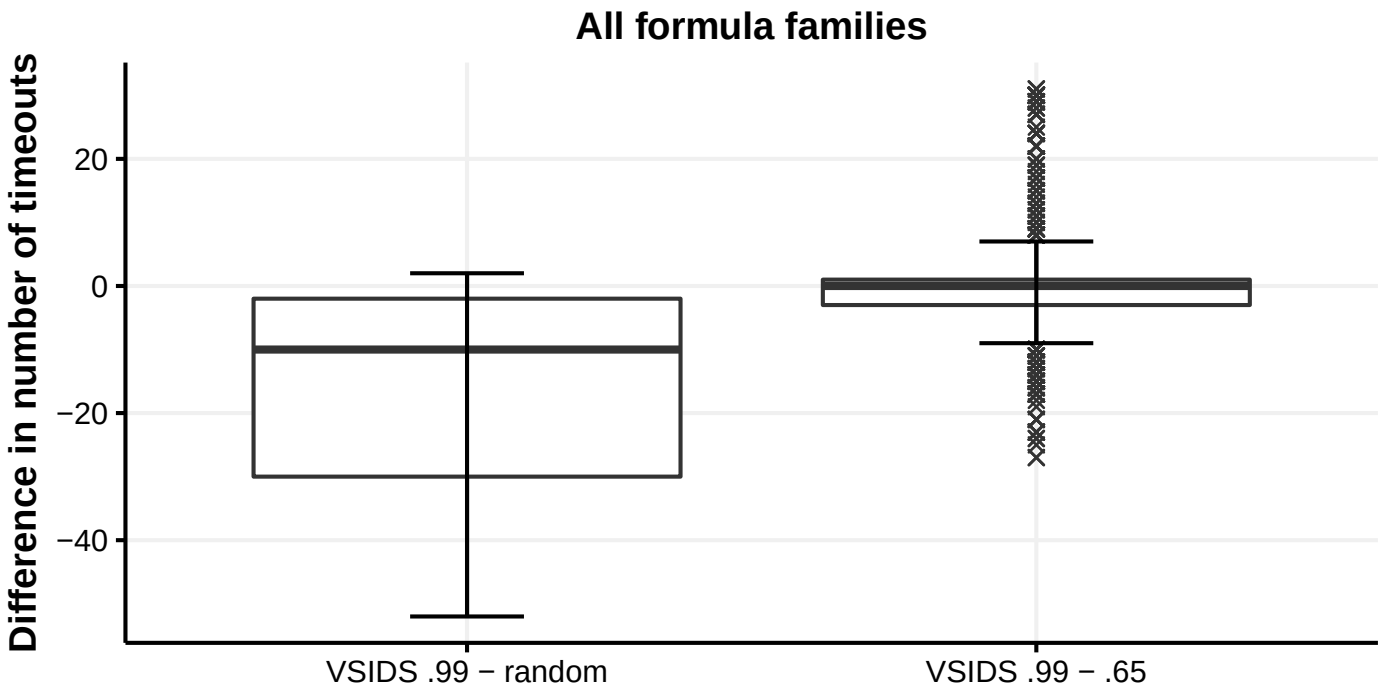


Branching Heuristics: Importance of Memory Horizon

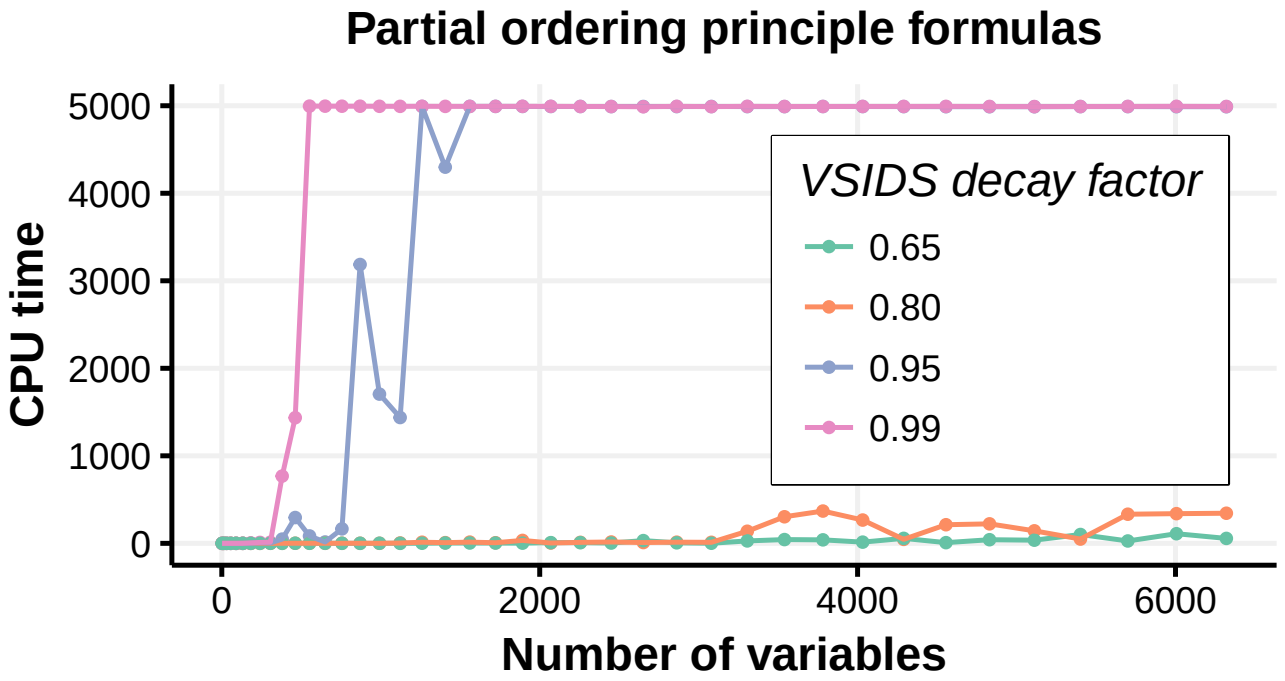
- Branch on variable appearing most often in recent conflicts
  + exponential decay to reward recent conflicts: VSIDS [MMZ$^+$01]
    - Low VSIDS factor = short memory
    - High VSIDS factor = long memory
- Choice high/low depends on instance (but random choice always bad)

Branching Heuristics: Importance of Memory Horizon

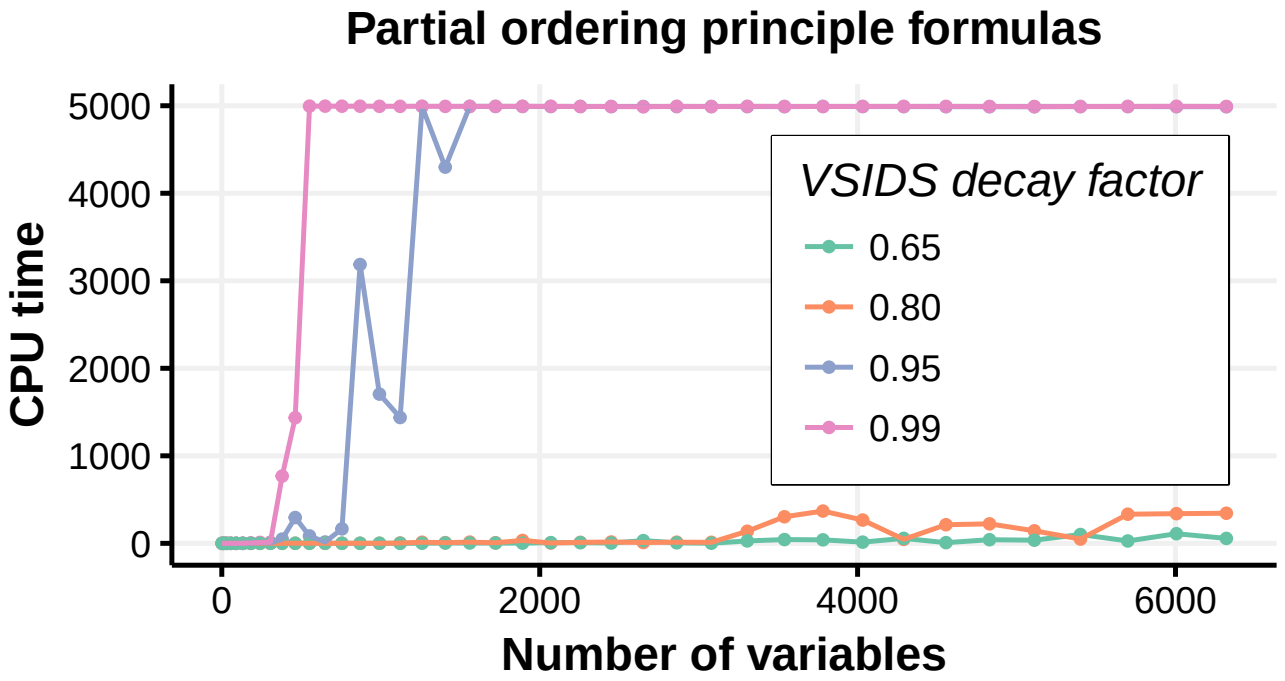
- Branch on variable appearing most often in recent conflicts
  + exponential decay to reward recent conflicts: VSIDS [MMZ+01]
    - Low VSIDS factor = short memory
    - High VSIDS factor = long memory
- Choice high/low depends on instance (but random choice always bad)



Branching Heuristics: A Particularly Dramatic Example



Branching Heuristics: A Particularly Dramatic Example



- What's going on? We're not sure...
- Instances very hard for tree-like resolution ($\approx$ DPLL) [BG01, BW01]
- Low VSIDS factor $\Rightarrow$ search focus locally on latest conflicts
- Maybe high VSIDS factor $\Rightarrow$ closer to DPLL-style search?!
  (Consistent with our experiments, but more data & insights needed)

Take-Away Message

This work

Goal: Not better solvers, but **understand** how best solvers work

Approach: Harness theory results for empirical study of heuristics

Conclusion: Yes, can get practical insights from theory benchmarks!

- Sometimes confirms conventional wisdom — nice to get evidence
- Sometimes more surprising results — raise questions for further study

Take-Away Message

This work

Goal: Not better solvers, but **understand** how best solvers work

Approach: Harness theory results for empirical study of heuristics

Conclusion: Yes, can get practical insights from theory benchmarks!

- Sometimes confirms conventional wisdom — nice to get evidence
- Sometimes more surprising results — raise questions for further study

Directions for future research

More in-depth study of intriguing questions raised

- Restarts: Only frequency important or also exact timing?
- Memory management: Trade-off speed/quality of proof search
- Branching: VSIDS factor crucial — how to get it right?
- Phase saving: Super-important also for unsatisfiable instances — why?

Take-Away Message

This work

Goal: Not better solvers, but **understand** how best solvers work

Approach: Harness theory results for empirical study of heuristics

Conclusion: Yes, can get practical insights from theory benchmarks!

- Sometimes confirms conventional wisdom — nice to get evidence
- Sometimes more surprising results — raise questions for further study

Directions for future research

More in-depth study of intriguing questions raised

- Restarts: Only frequency important or also exact timing?
- Memory management: Trade-off speed/quality of proof search
- Branching: VSIDS factor crucial — how to get it right?
- Phase saving: Super-important also for unsatisfiable instances — why?

Analogous study on industrial benchmarks

Take-Away Message

This work

Goal: Not better solvers, but **understand** how best solvers work

Approach: Harness theory results for empirical study of heuristics

Conclusion: Yes, can get practical insights from theory benchmarks!

- Sometimes confirms conventional wisdom — nice to get evidence
- Sometimes more surprising results — raise questions for further study

Directions for future research

More in-depth study of intriguing questions raised

- Restarts: Only frequency important or also exact timing?
- Memory management: Trade-off speed/quality of proof search
- Branching: VSIDS factor crucial — how to get it right?
- Phase saving: Super-important also for unsatisfiable instances — why?

Analogous study on industrial benchmarks

. . . And maybe better solvers after all, thanks to better understanding

Take-Away Message

This work

Goal: Not better solvers, but **understand** how best solvers work

Approach: Harness theory results for empirical study of heuristics

Conclusion: Yes, can get practical insights from theory benchmarks!

- Sometimes confirms conventional wisdom — nice to get evidence
- Sometimes more surprising results — raise questions for further study

Directions for future research

More in-depth study of intriguing questions raised

- Restarts: Only frequency important or also exact timing?
- Memory management: Trade-off speed/quality of proof search
- Branching: VSIDS factor crucial — how to get it right?
- Phase saving: Super-important also for unsatisfiable instances — why?

Analogous study on industrial benchmarks

. . . And maybe better solvers after all, thanks to better understanding

Thank you for your attention!

References I

[AFT11] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, January 2011. Preliminary version in *SAT '09*.

[AJPU07] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3(5):81–102, May 2007. Preliminary version in *STOC '02*.

[AS09] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pages 399–404, July 2009.

[BBI12] Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, pages 213–232, May 2012.

[BG01] María Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, December 2001. Preliminary version in *FOCS '99*.

References II

[BN11] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011.

[BNT13] Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13)*, pages 813–822, May 2013.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC '71)*, pages 151–158, 1971.

References III

[ES04] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *6th International Conference on Theory and Applications of Satisfiability Testing (SAT '03), Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2004.

[JMNŽ12] Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný. Relating proof complexity measures and practical hardness of SAT. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 316–331. Springer, October 2012.

[KSM11] Hadi Katebi, Karem A. Sakallah, and João P. Marques-Silva. Empirical study of the anatomy of modern SAT solvers. In *Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT '11)*, volume 6695 of *Lecture Notes in Computer Science*, pages 343–356. Springer, June 2011.

[LM02] Inês Lynce and João P. Marques-Silva. Building state-of-the-art SAT solvers. In *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI '02)*, pages 166–170. IOS Press, May 2002.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

References IV

[MN14] Mladen Mikša and Jakob Nordström. Long proofs of (seemingly) simple formulas. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 121–137. Springer, July 2014.

[MS96] João P. Marques-Silva and Karem A. Sakallah. GRASP—a new search algorithm for satisfiability. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD '96)*, pages 220–227, November 1996.

[PD07] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing (SAT '07)*, volume 4501 of *Lecture Notes in Computer Science*, pages 294–299. Springer, May 2007.

[PD11] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, February 2011. Preliminary version in *CP '09*.

[SM11] Karem A. Sakallah and João Marques-Silva. Anatomy and empirical evaluation of modern SAT solvers. *Bulletin of the European Association for Theoretical Computer Science*, 103:96–121, February 2011.