Boolean Satisfiability: Theory and Engineering

Moshe Y. Vardi

Rice University

Why Are We Here?

V., March 2014:

"The workshop brought together leading SAT-complexity theorists and SAT-solving engineers, two groups that rarely meet together and barely speak the same technical language. The hope was that exposing theoreticians and engineers to state-of-the-art developments in SAT theory and engineering would narrow the chasm between these communities."

Role of Theory

Personal Perspective: Computer science is ultimately an *applied* field! What is the role of theory?

- Clarify conceptual framework, e.g., NP-completeness
- Suggest inherent limitations, e.g., lower bounds
- Explain experimental findings, e.g., CDCL and Resolution
- Suggest experimental possibilties, ???

Challenges on Both Sides

- "What has become clear, however, and also became painfully evident at the SAT workshop, is that worst-case analysis actually sheds very little light on the behavior of algorithms on real-life instances."

- "An annual SAT competition serves as a powerful motivator, as winning a track of the competition is a path to quick professional recognition. At the same time, this heuristic approach lacks science, not only theoretical science but also empirical science."

Beyond Worst-Case Complexity

Roughgarden, 2018:

- *Typical Instances*: articulating properties of "real-worlds" inputs, and proving rigorous and meaningful algorithmic guarantees for inputs with these properties.
- *Stable Instances*
- *Planted and Semi-Random Models*
- *Smoothed Analysis*

"Typical" SAT Instances

Observation: Many SAT-instance families have bounded treewidth and even bounded pathwidth, e.g., SAT-based planning instances and BMC instances.

Counter-Observation: Tree-decomposition-based SAT-solving algorithms are exponential in the treewidth, and real-life instances typically have large treewidth.

Structural Measures and SAT Solver Performance

E. Zulkoski, PhD Dissertation: *Understanding and Enhancing CDCL-based SAT Solvers*, 2018

- *Pro*: A comprehensive performance study of several structural measures on a large set of SAT instances from SAT Competitions.
- *Con*: Not clear how to draw conclusions from a large set of instances.

"Testing Heuristics: We Have It All Wrong"

J.N. Hooker, Journal of Heuristics, 1995:

"The competitive nature of most algorithmic experimentation is a source of problems that are all too familiar to the research community. It is hard to make fair comparisons between algorithms and to assemble realistic test problems. Competitive testing tells us which algorithm is faster but not why. Because it requires polished code, it consumes time and energy that could be better spent doing more experiments. This article argues that a more scientific approach of controlled experimentation, similar to that used in other empirical sciences, avoids or alleviates these problems. We have confused research and development; competitive testing is suited only for the latter."

A Scientific Approach: Scalabale Benchmarks

- Generate a benchmark family of closely-related formulas of increasing size and study how performance scales wrt size, e.g., PHP_n or $unroll(C, n)$ some sequential circuit C.

- Choose benchmark family to have some desired property, e.g., $unroll(C, n)$ has bounded pathwdith.

Example: BDD-SAT vs. ZChaff, 2004

BDD-SAT Pan+V., 2004

- A SAT instance φ is a formula of the form

$$(\exists v_1)(\exists v_2)\ldots(\exists v_n)(c_1 \wedge c_2 \wedge \ldots \wedge c_m)$$

- Compute BDD for φ: 0 or 1
- *Early Quantification*: Perform quantification *eagerly* and conjunction *lazily*.
- *Variable Ordering*: Use fast graph-theoretic heuristics to order variables.
- *Clause Ordering*: Use fast graph-theoretic heuristics to order clauses.

BDDs vs. ZChaff: Round I

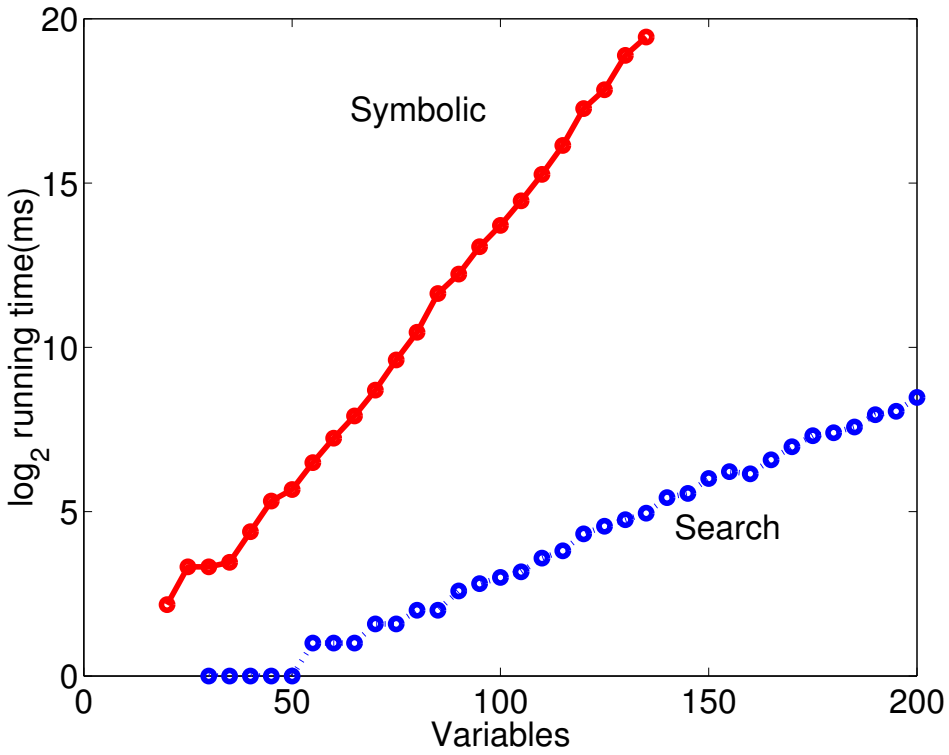

Figure 1: Random 3-CNF, density=6

BDDs vs. ZChaff: Round II

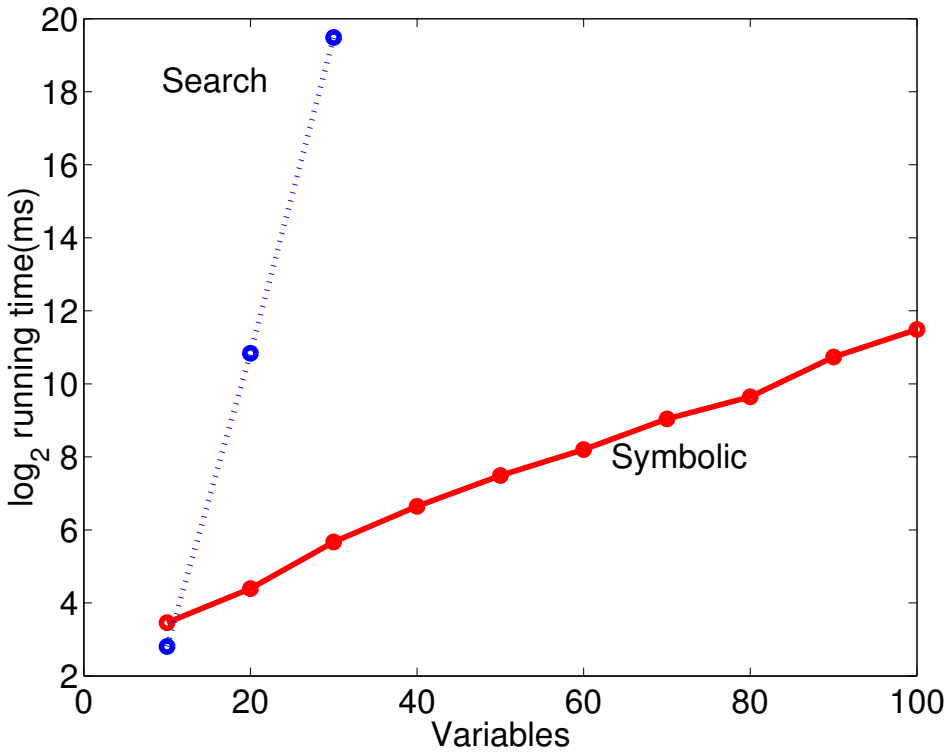

Figure 2: Random Biconditionals

BDDs vs. ZChaff: Round III

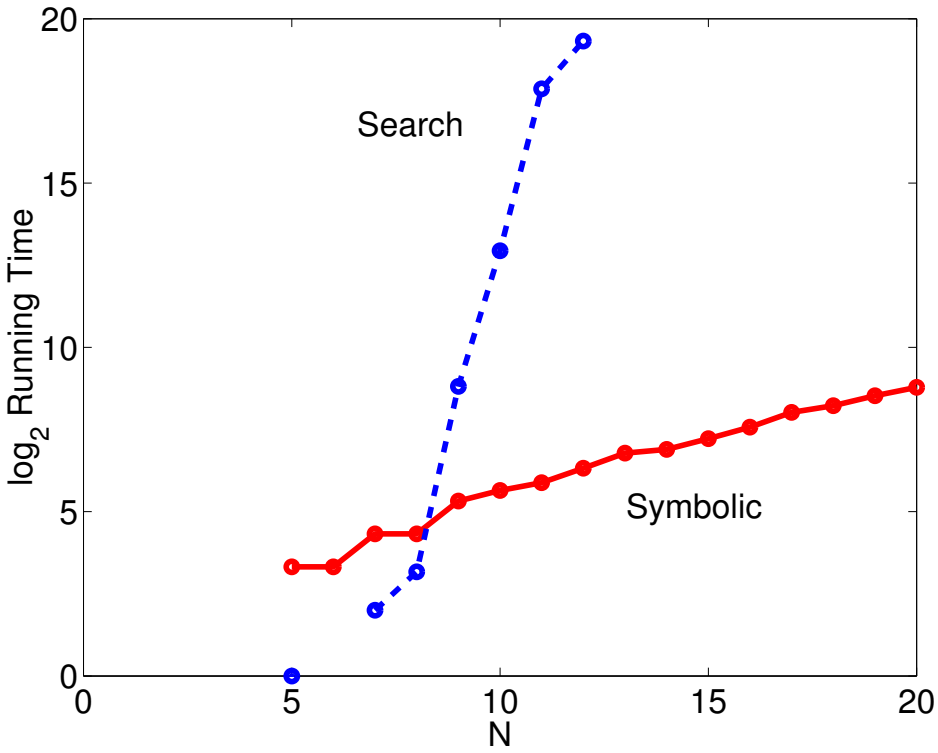

Figure 3: Mutilated Checkerboard

Scalable Benchmarks: Recent Work

Seeking Practical CDCL Insights from Theoretical SAT Benchmakrs, Elffers et al., IJCAI'18

Methodology: Scalable theoretical SAT benchmarks, e.g., Tseitin formulas, pebbling formulas, etc.

Sample Results:

- While the theoretical power of restarts is open, the practical value is clear.
- Clause learning is powerful both theoretically and practically.
- VSIDS decay factor is important practically, but we do not understand why.

Challenge to Experimentalists

Challenge: Perform a comprehensive SAT-heuristic performance analysis on scalable, industrial benchmarks.

Questions:

- What is the Big O complexity of SAT solvers on scalabale benchmarks, e.g., $(1+\varepsilon)^n$?
- Do successful SAT-solving heuristics offer *additive*, *multiplicative*, or *exponential* advantage?
- Do the inights from scalability analysis agree with the insights from the Competition?

Research Challenge

Observations:

- CDCL is resolution based
- BDD-proofs dominate resolution exponentially [Atserias+Kolaitis+V., 2004].
- CDCL combines search and inference.
- BDD-SAT uses only inference, but introduces new variables (BDD nodes), so can be viewed as "Extended Frege".

Research Challenge: Can we develop an effective search+inference SAT-solver based on BDD-proofs?

Important: Evaluate performance on scalable benchmarks, rather than via competition!

Three SAT Communities

- *Experimentalists* – competition driven
- *Proof Theorists* – lower-bounds driven
- *Algorithmicists* – upper-bounds driven

Worst-Case 3-SAT-Solving Algorithms

- Monien and Speckenmeyer, 1985: $O(1.6181^n)$
- Kullman, 1999: $O(1.5045^n)$
- Iwama+Tamaki, 2004: $O(1.324^n)$ (rand.)
- Hertli, 2018: $O(1.30704^n)$ (rand.)

Questions:

- Any connection to proof complexity?
- Has anyone implemented these algorithms?
- Are these algorithms any good?
- Do they at least offer new ideas for heuristics?
- Why no practical randomized algorithm for SAT?

In Conclusion

Figure 4: Change is Hard