# On the Efficiency of the VSIDS Decision Heuristic

Sharad Malik and Victor Ying*
Princeton University
*MIT starting Fall 2016

**Theoretical Foundations of SAT Solving Workshop**
**Fields Institute, University of Toronto**
**August 16, 2016**

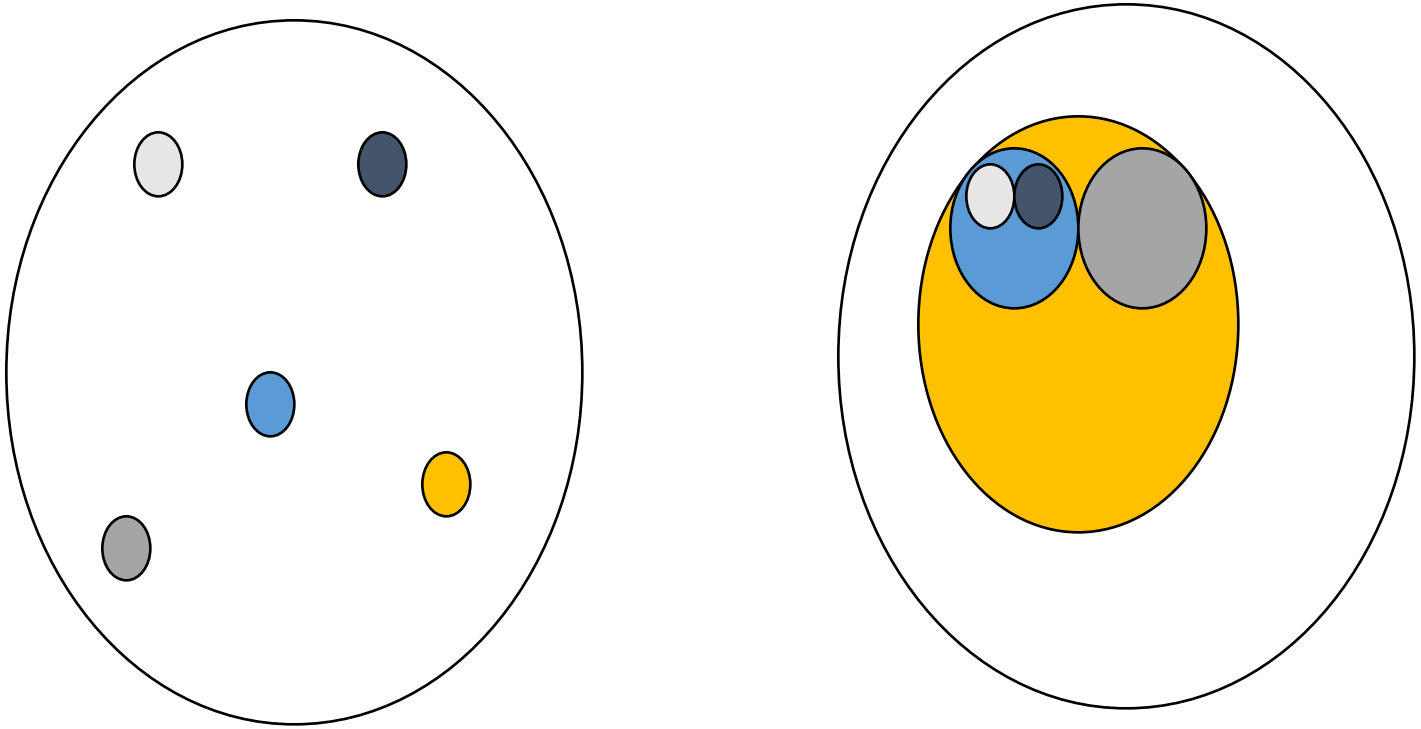
# VSIDS: Variable State Independent Decaying Sum

- Activity Based Decision Heuristic
  - Rank variables by literal count in the initial clause database
  - Only increment counts as new (learned) clauses are added
  - Periodically, divide all counts by a constant
- Quasi-static
  - Static because it doesn't depend on variable state
  - Not static because it gradually changes as new clauses are added
    - Decay causes bias toward **recent** conflicts.
    - Has a beneficial interaction with 2-literal watching
- Very effective
  - Variations and improvements widely used
  - Ongoing efforts to understand its effectiveness

M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineerng an efficient SAT solver," DAC 2001

J. H. Liang, V. Ganesh, E. Zulkoski, A. Zaman, and K. Czarnecki, "Understanding VSIDS branching heuristics in conflict-driven clause-learning SAT solvers," HVC 2015

# Activity Based Heuristics and Locality Based Search

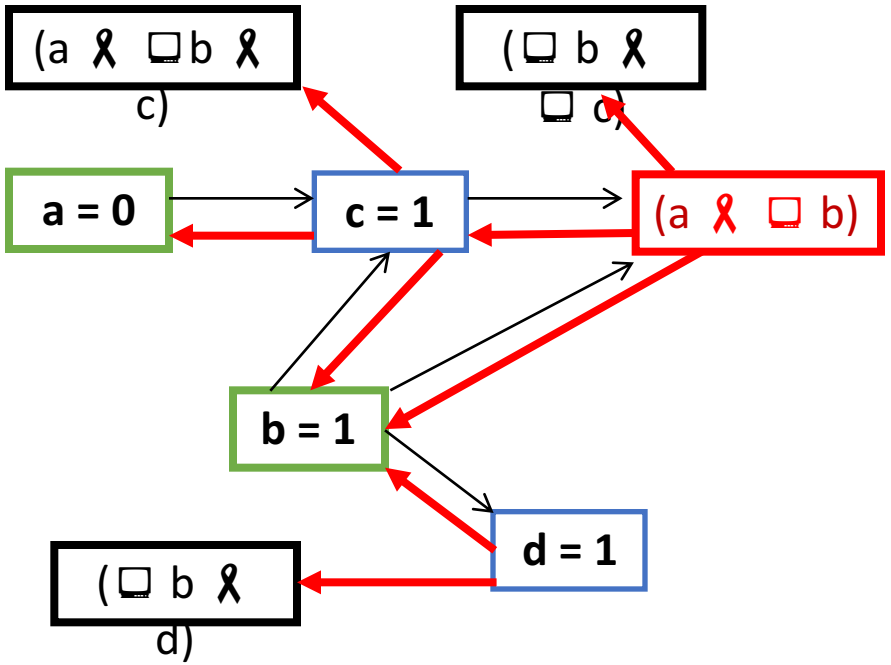


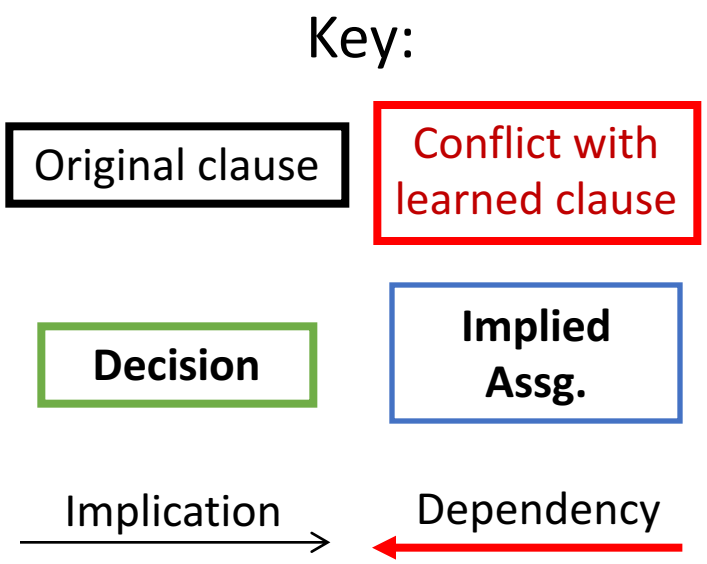
- By focusing on a sub-space, the covered spaces tend to coalesce
  - More opportunities for resolution, and generating smaller clauses, since most of the variables are common.
  - Variable activity based heuristics lead to locality based search

# VSIDS Effective ✓
# VSIDS Efficient?

- What fraction of its work is useful?
  - Help understand potential improvements.
- Study in the context of modern CDCL solvers
  - Unit propagation
  - Learned conflict clause is asserting
- Experimental Setup
  - MiniSAT 2.2.0
    - No preprocessing for now…
  - Log critical events – decisions, implications, conflicts, learned clauses, restarts…
  - Generate and analyze Event Dependency Graph
    - Offline
  - SAT Industrial Benchmarks
    - 300 second timeout to manage memory and log size

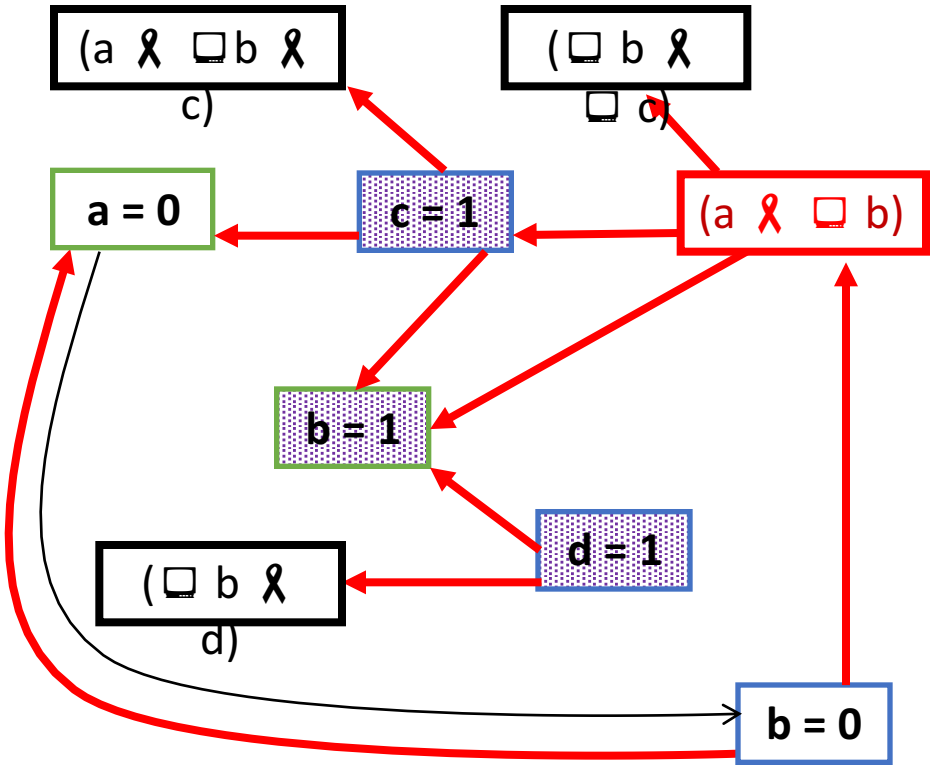
# Implication and Event Dependency Graphs



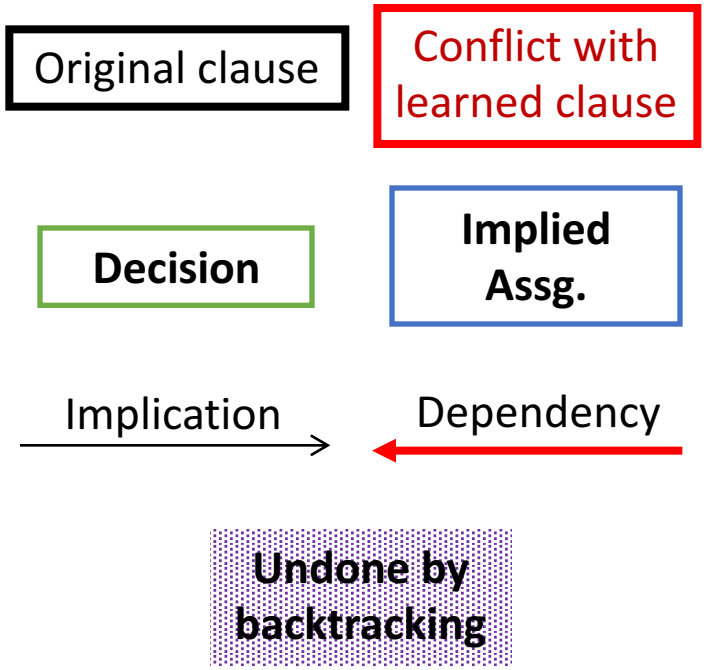




- Implication edges generate dependencies.
- Each implied assignment and conflict clause also depend on antecedent clauses

# Implication and Event Dependency Graphs



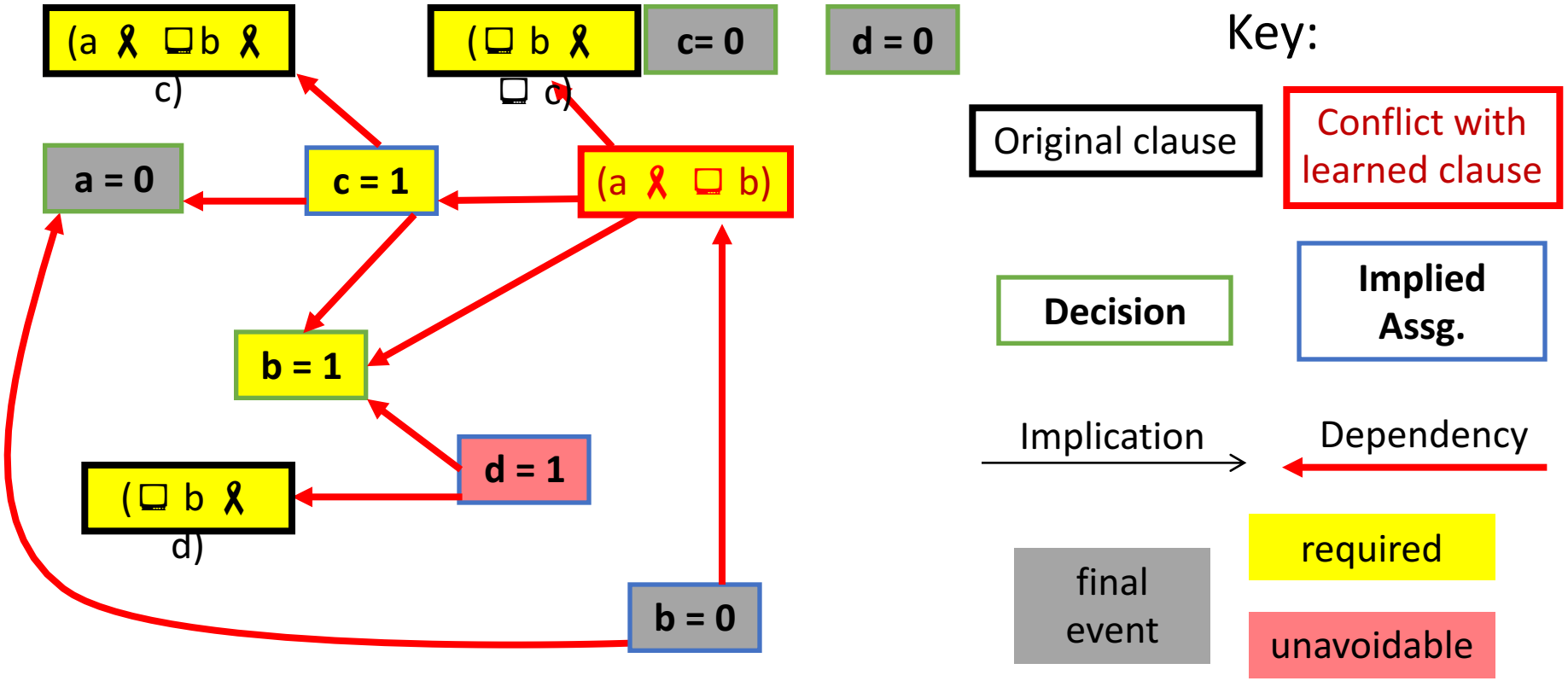

Key:




- Implication graph represents a single state in solver execution.
- Event dependency graph represents entire solver execution. Later events depend on conflict clauses, which depend on previous events.
- Subsumes implication graphs, resolution proofs

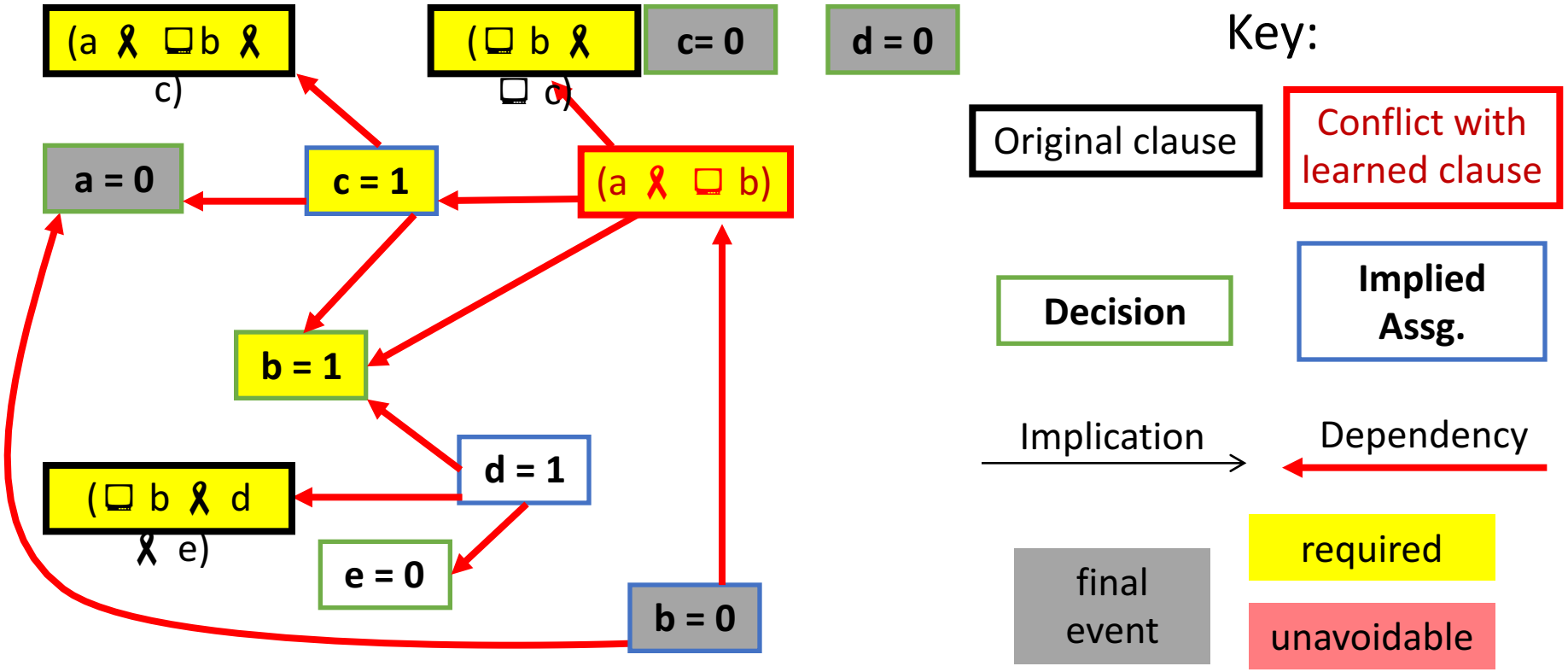
Standardize format for event logs?
Study solution evolution

# Required, Unavoidable and Wasted Work



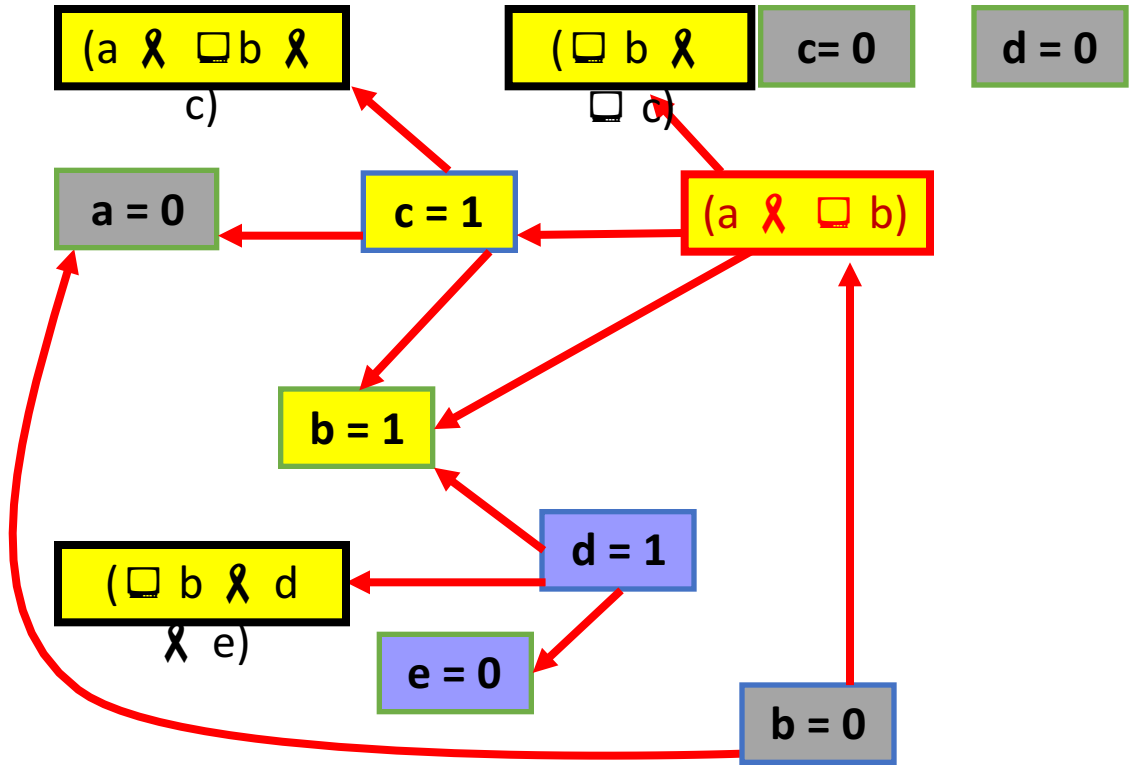

- Solver finishes with a final conflict or satisfying assignment.
- Final event and its dependencies are required.
- Implications with required dependencies are unavoidable.

# Required, Unavoidable and Wasted Work




- Solver finishes with a final conflict or satisfying assignment.
- Final event and its dependencies are required.
- Implications with required dependencies are unavoidable.
- Unrequired implied assignments are avoidable/wasted if they depend on an unrequired decision.

# Required, Unavoidable and Wasted Work




Key:

Original clause

Conflict with learned clause

Decision

Implied Assg.

Implication

Dependency

final event

required

unavoidable

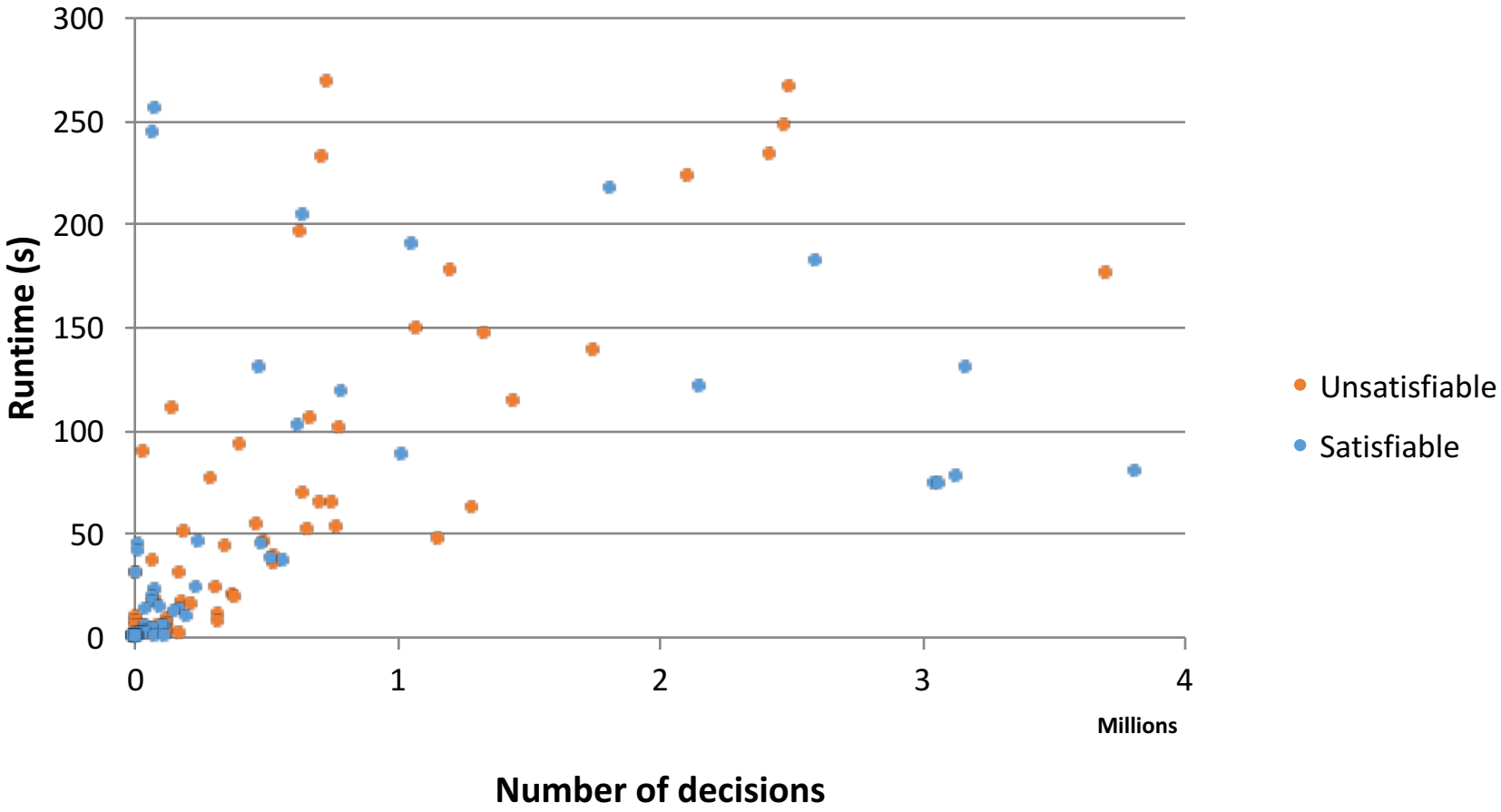
wasted

unrequired

Ignores contributions to decision heuristic (VSIDS score)

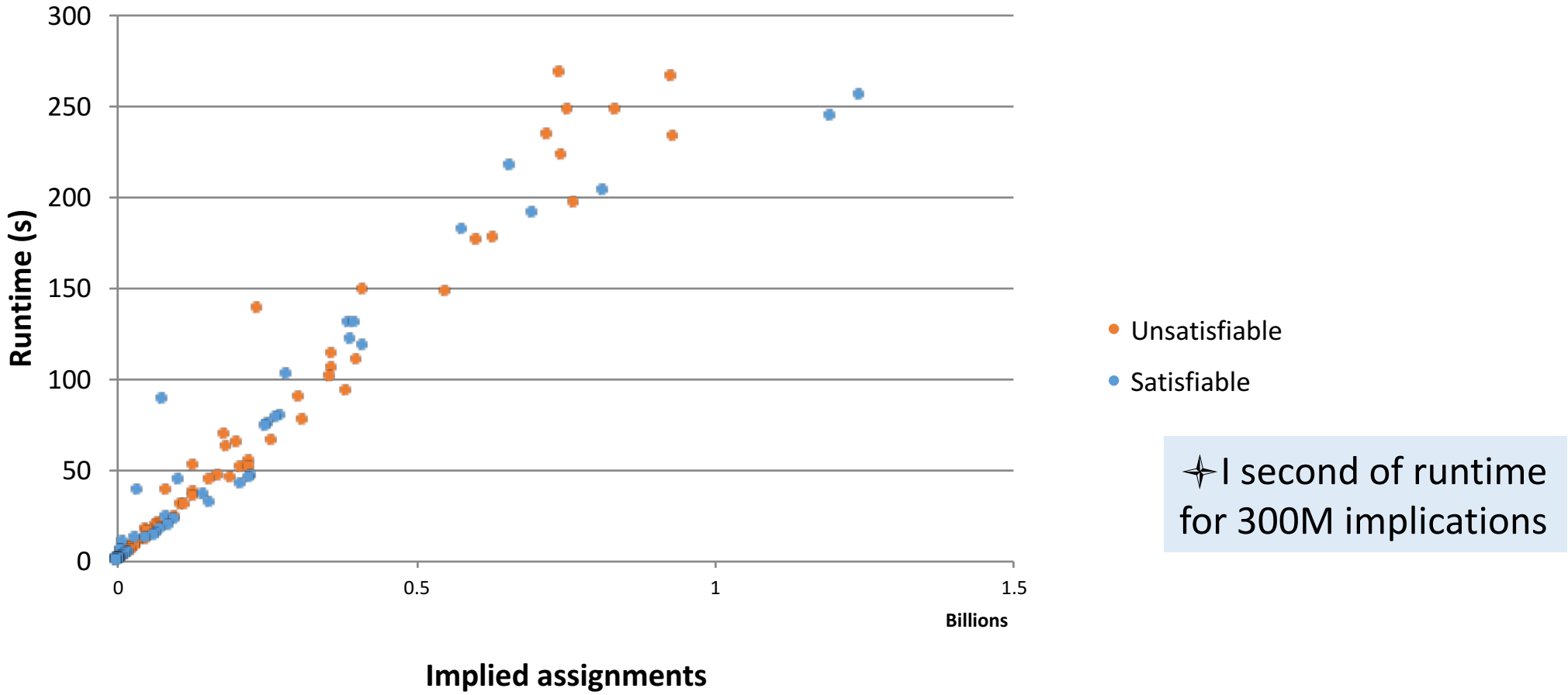
- Solver finishes with a final conflict or satisfying assignment.
- Final event and its dependencies are required.
- Implications with required dependencies are unavoidable.
- Unrequired implied assignments are avoidable/wasted if they depend on an unrequired decision.

# Where is time spent?




Number of decisions is a poor predictor of runtime.

# Where is time spent?

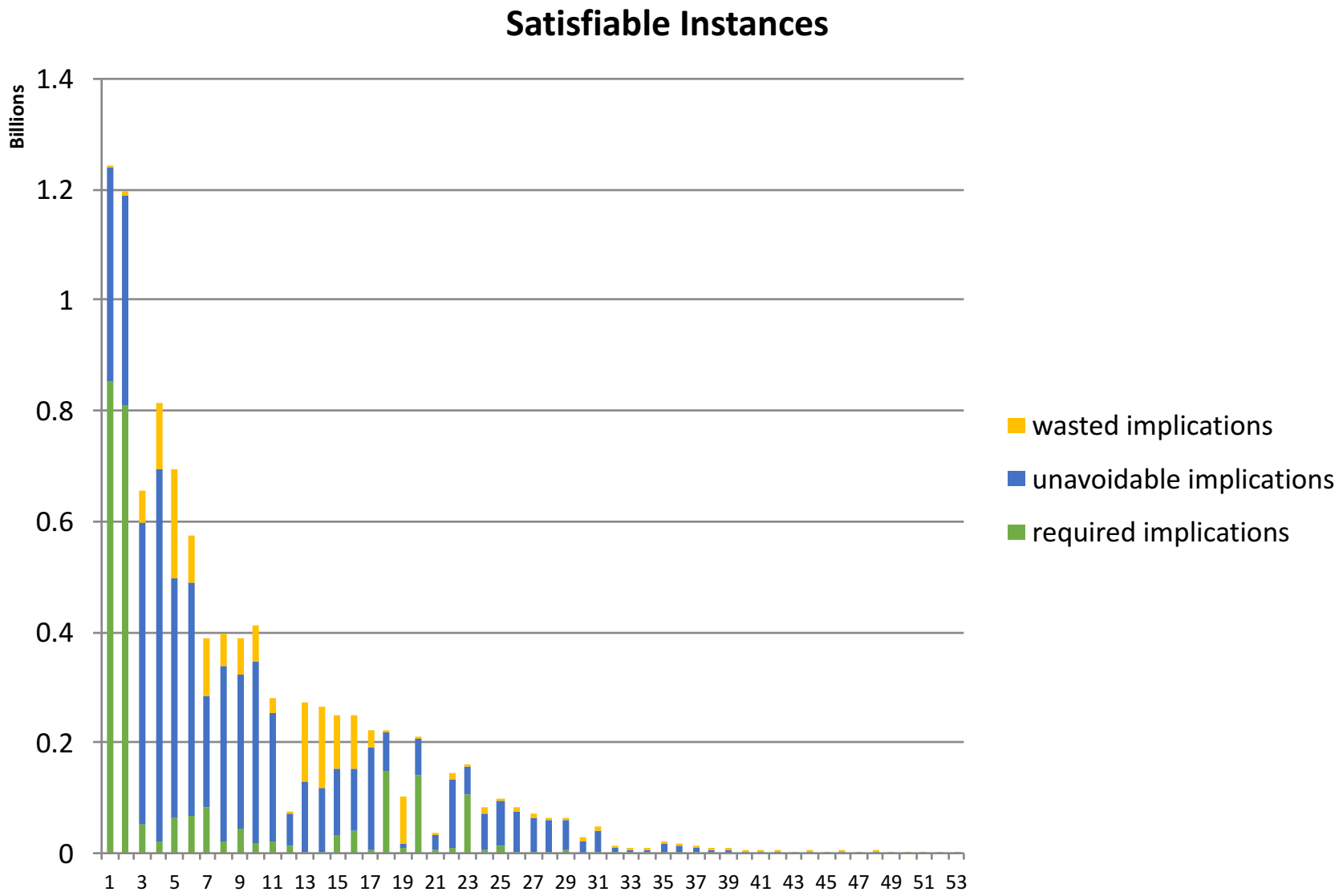



✦1 second of runtime for 300M implications

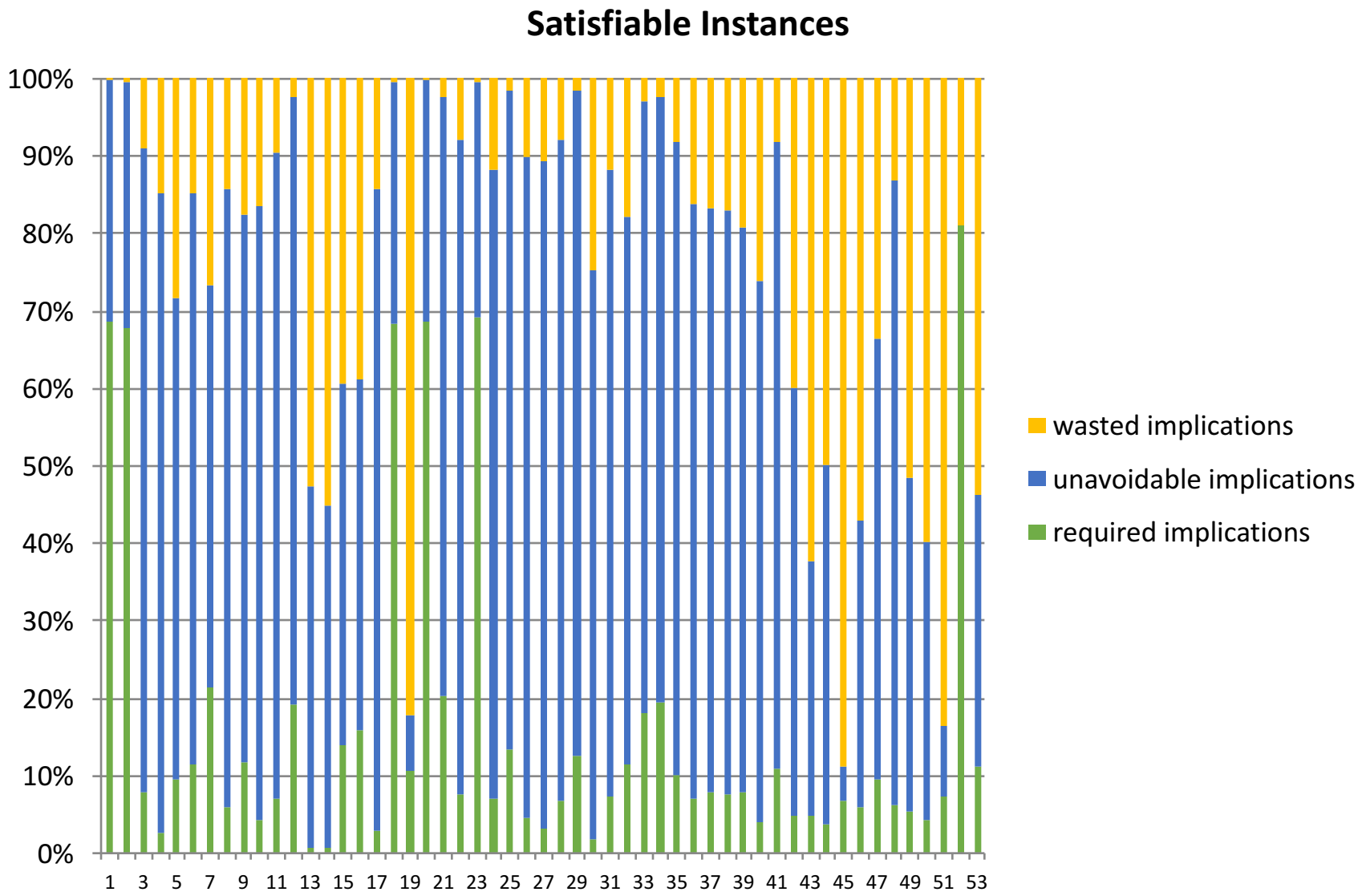
Most time is spent finding implications. Reducing unrequired implications is essential.

# Satisfiable Instance Results

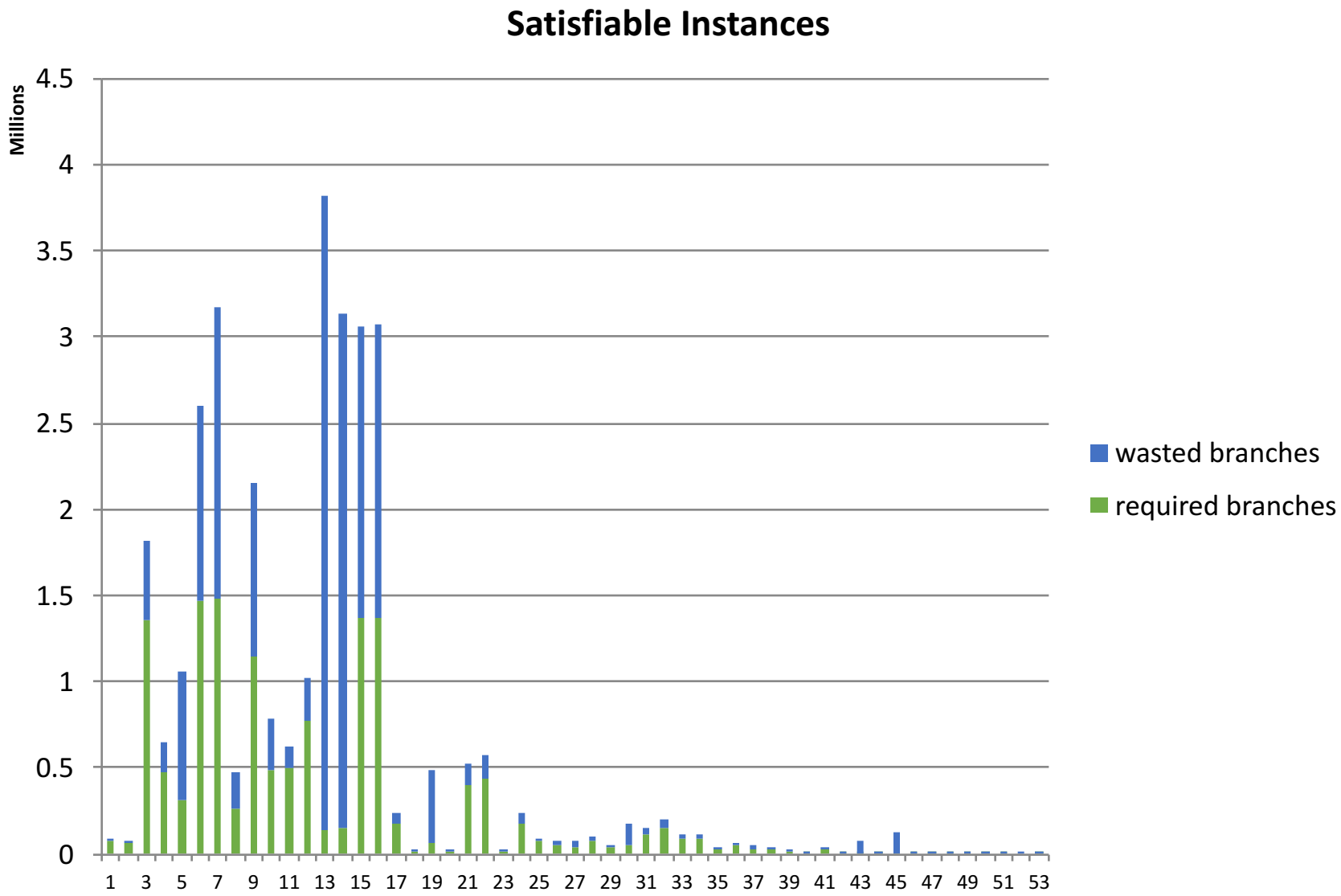
SAT Industrial Benchmarks
- 53 instances under 300 second timeout



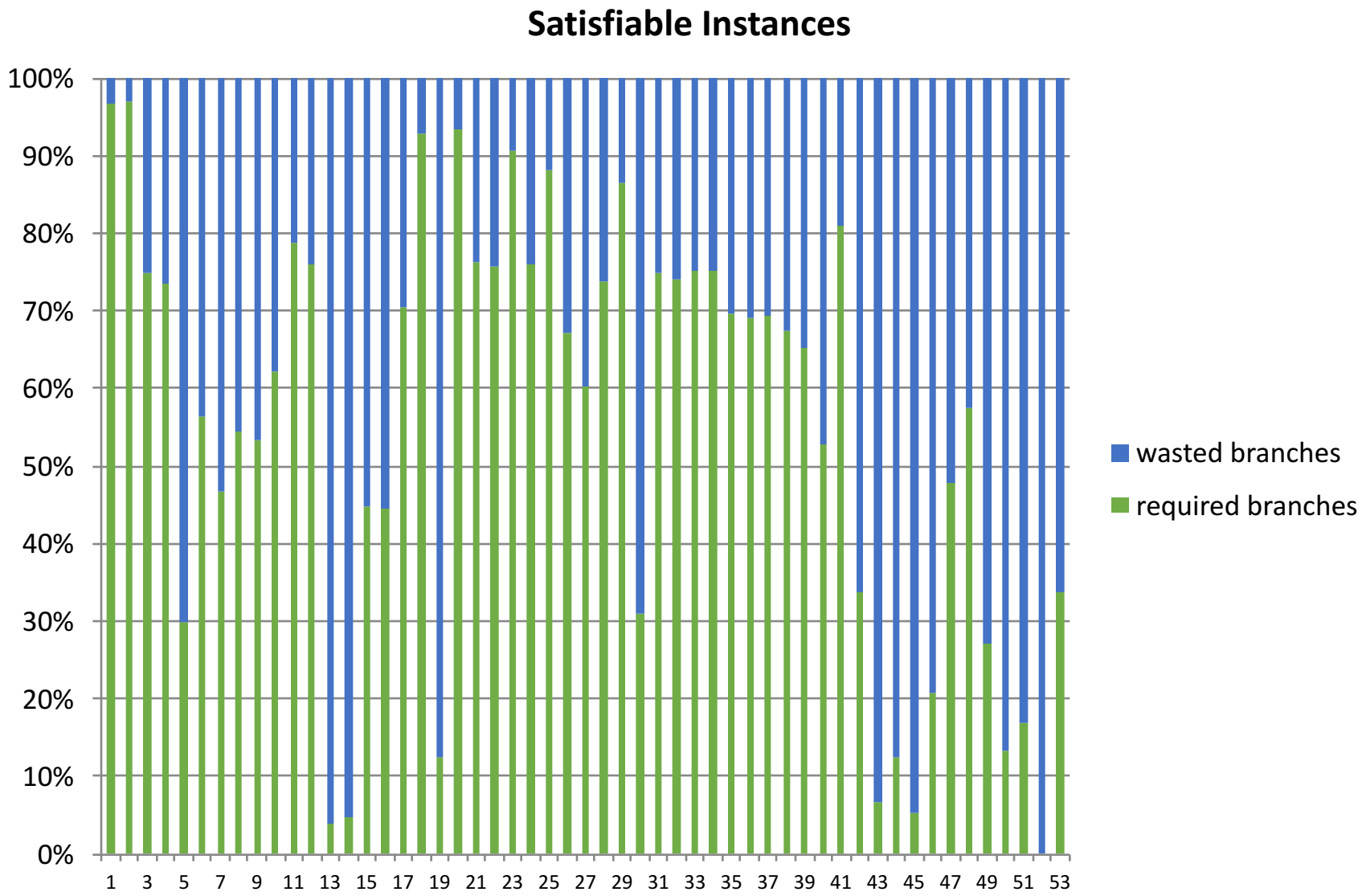

Only small fraction (15%) wasted, so limited improvements available in decision heuristic alone.




Most (58%) implications are “unavoidable” but not required, new ideas needed to exploit this.



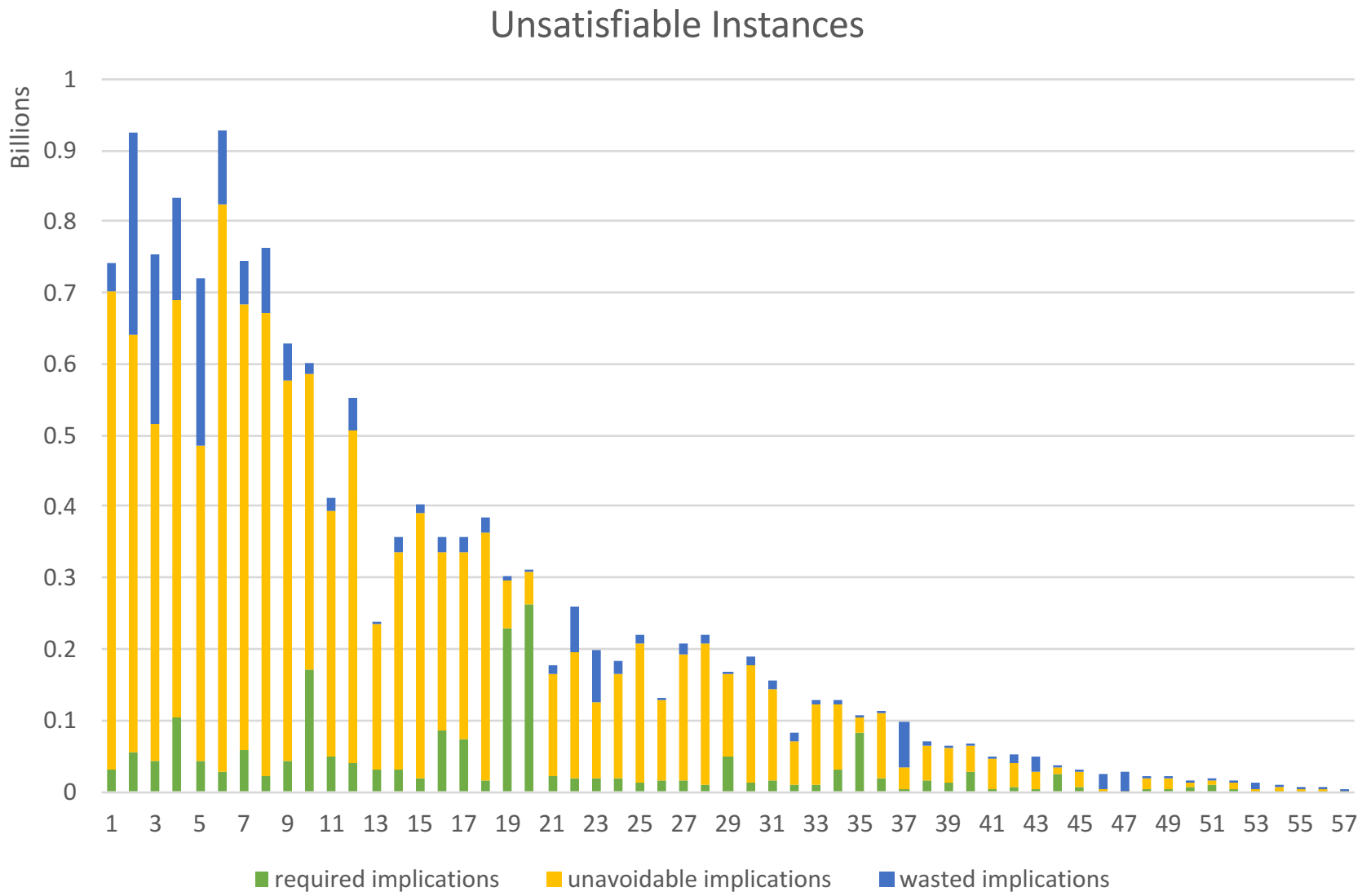

A sizable fraction (56%) of branches wasted, but these branches don’t yield many implications.



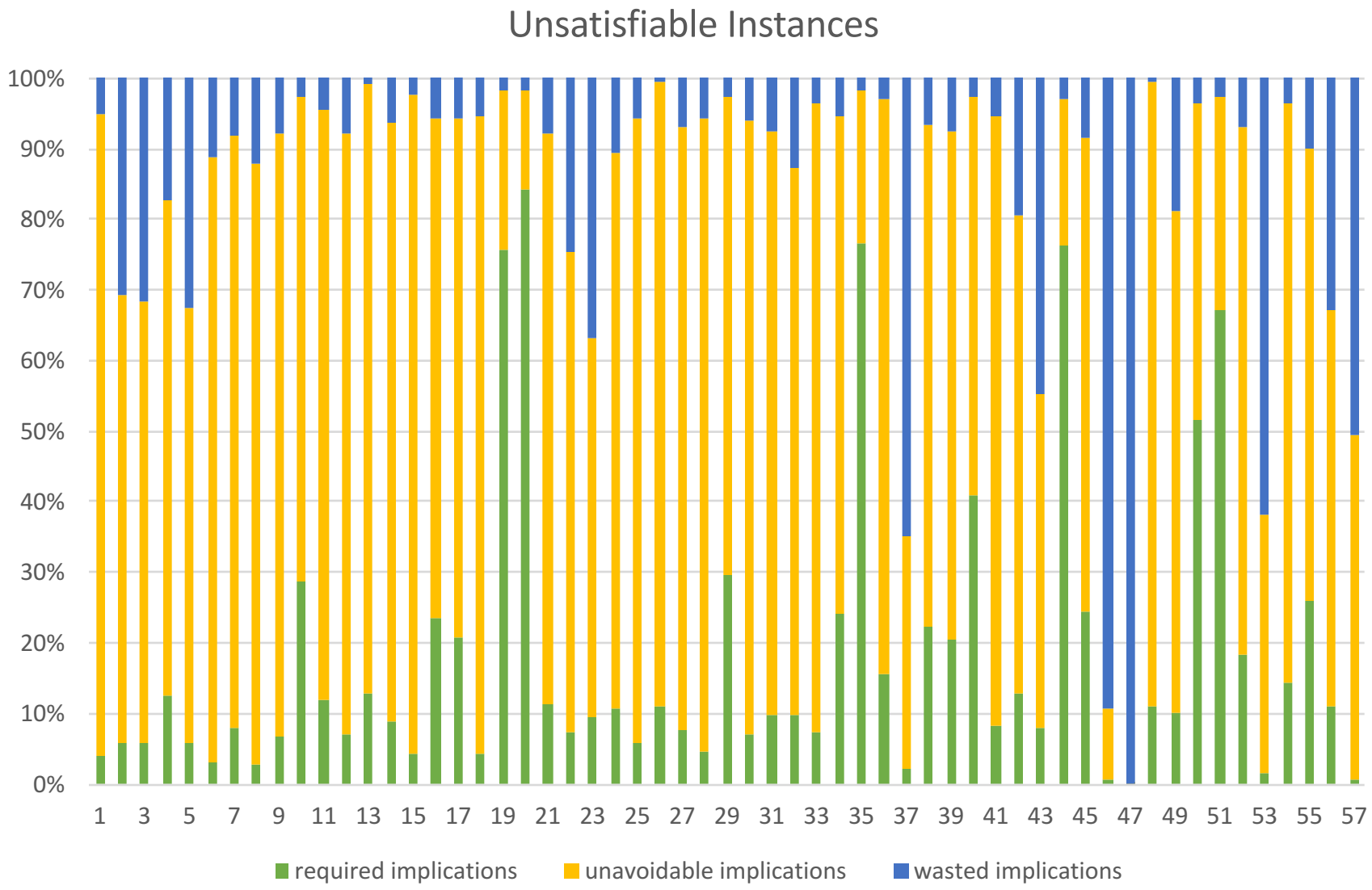

A sizable fraction (56%) of branches wasted, but these branches don’t yield many implications.

# Unsatisfiable Instance Results

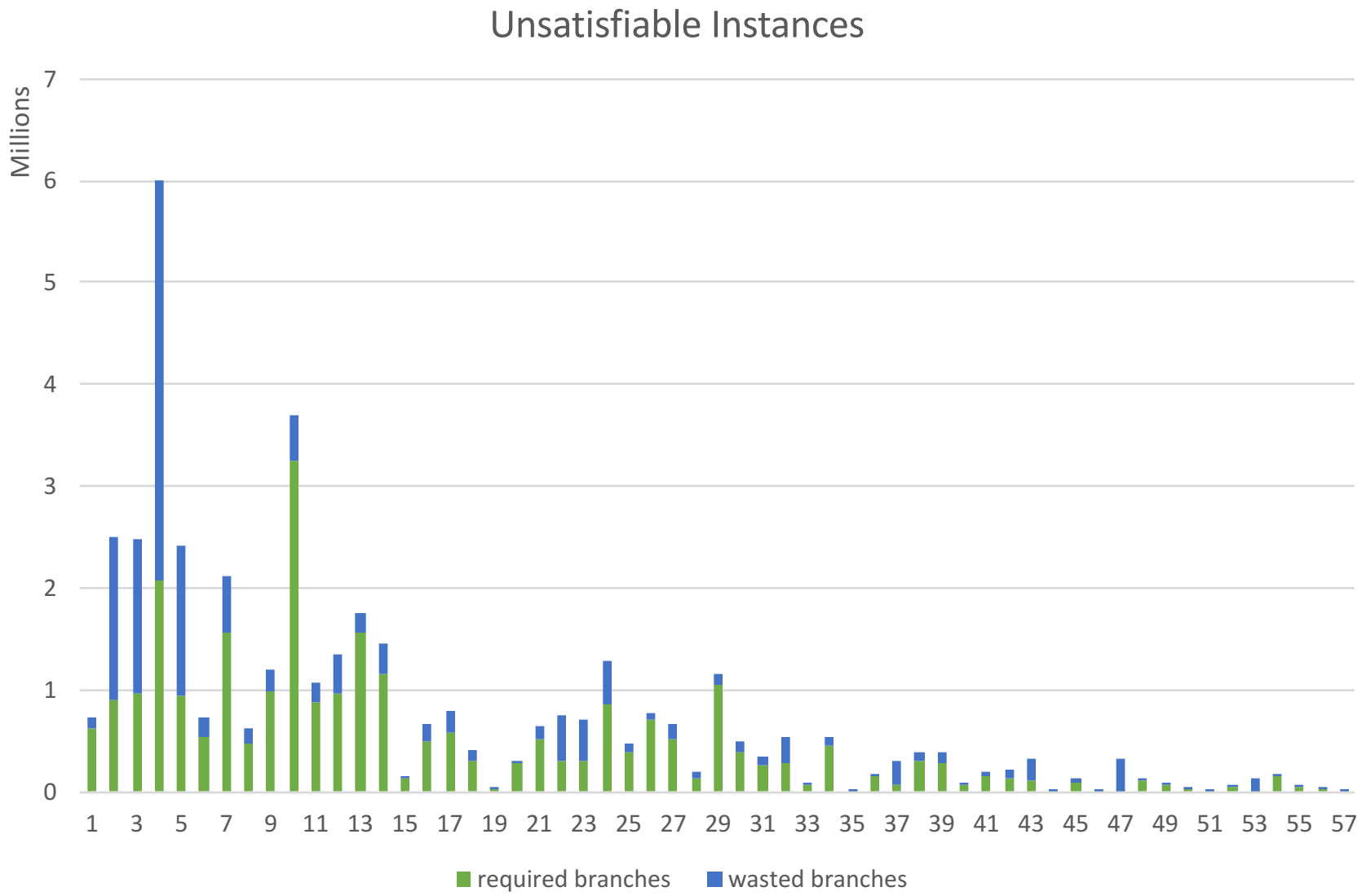
SAT Industrial Benchmarks
- 57 instances under 300 second timeout



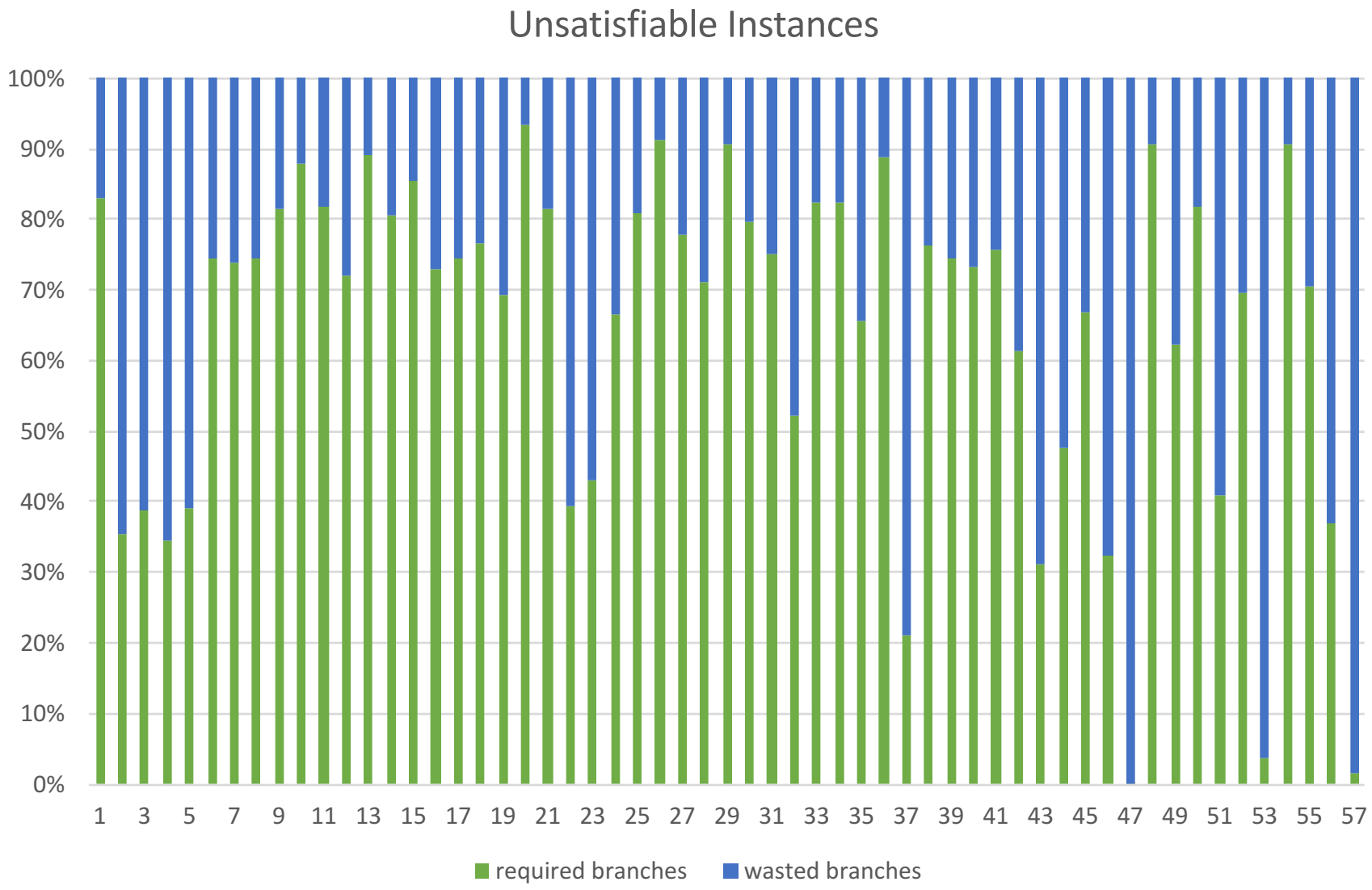

Only small fraction (13%) wasted, so limited improvements available in decision heuristic alone.




Most (74%) implications are “unavoidable” but not required, new ideas needed to exploit this.




A sizable fraction (37%) of branches wasted, but these branches don’t yield many implications.




A sizable fraction (37%) of branches wasted, but these branches don’t yield many implications.

# Conclusions

- VSIDS is mostly efficient.
  - Most implications required or unavoidable.
  - If improved decision heuristic cut out wasted decisions, runtime improvements would be marginal.
- Perhaps improved ordering in implications could help.
  - Most work is “unavoidable” in this analysis that treats the implementation of finding implications as fixed.
- Drive improvements through data analysis.
  - Event log and Event Dependency Graph are useful analysis tools
    - e.g., Time series analysis of Event Dependency Graph to determine “phases”
  - Standardize some form of event log to enable this
    - Subsumes proof logs