SAT Competition 2016: Recent Developments

Tomás Balyo Marijn J.H. Heule Matti Järvisalo

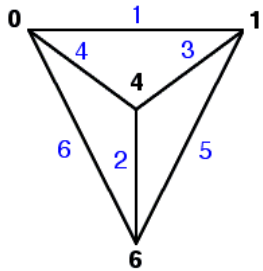
UNIVERSITY OF HELSINKI

AAAI-17

February 6, 2017

Satisfiability (SAT) solving has many applications

formal verification

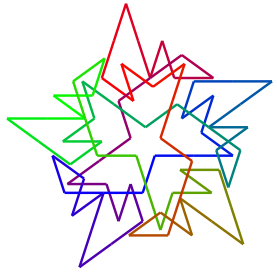


graph theory

bioinformatics

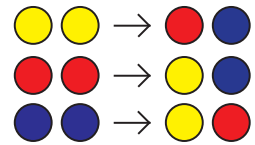
train safety

planning

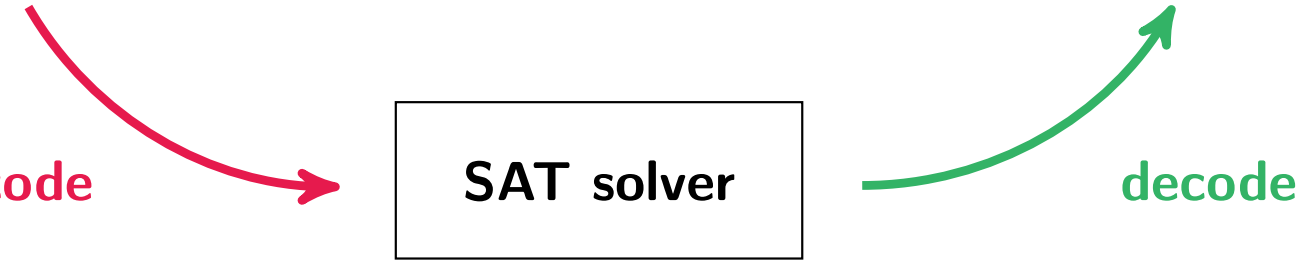


number theory

cryptography



rewrite termination



SAT solvers deal with formulas with millions of clauses!

What is the SAT Competition?

International Competition of Boolean Satisfiability solvers

Focus: "The purpose of the competition is to identify new challenging benchmarks and to promote new solvers for the propositional satisfiability problem (SAT) as well as to compare them with state-of-the-art solvers."

Long tradition:

- First SAT Competition in 2002
- Nice SAT Competitions
- Four SAT Races
- One SAT Challenge (2012)

What was new in SAT Competition 2016?

We had two new tracks:

- Agile Track: compare solvers with small overhead on thousands of easy benchmarks using a small time limit.
- NoLimit Track: remove all requirement for solvers in the other tracks, such as proving a model for SAT benchmarks; proving a proof for UNST benchmarks; submit only code sources; and allowing portfolios. Only new benchmarks were used in this track.

Proof validation of unsatisfiability results now mainstream:

- Mandatory proof logging in the Main Track
- Binary proof format used to reduce space usage.

Tracks I

Main (Sequential) Track (29 solvers)

- 300 "application" and 200 "crafted" benchmarks
- 5,000 (s) solving limit & 20,000 (s) proof checking limit
- Solvers run on a single core
- Proof logging of unsatisfiability results required

Parallel Track (13 solvers)

- The same benchmark suite as the Main Track
- 5,000 (s) limit for solving
- 24 (48) CPU cores (hyper-threading), 64GB RAM

Random Satisfiable Track (9 solvers)

- 240 random satisfiable benchmarks
- 5,000 (s) limit for solving

Tracks II

Incremental Library Track (8 solvers)

- benchmarks originate from applications, such as hardware model checking, using different parameter settings.
- average rank for each application determines winner

Introducing Agile Track (30 solvers)

- 5,000 benchmarks, all coming from SMT solving
- 60 (s) limit for solving

Introducing No-Limit Track (21 solvers)

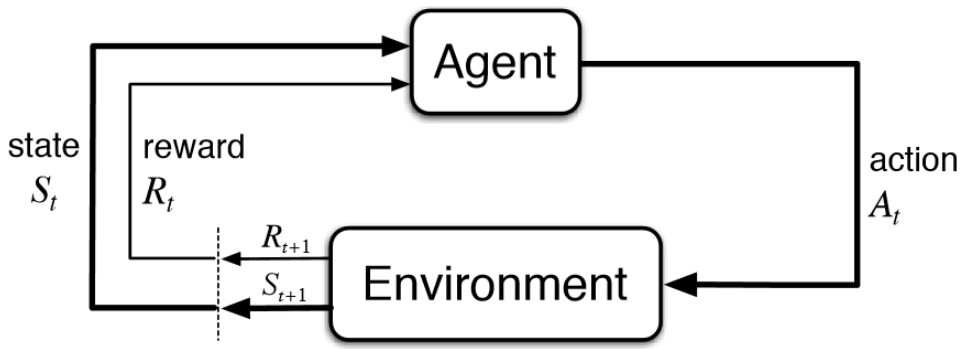
- 350 brand new benchmarks (subset of Main Track suite)
- 5,000 (s) limit for solving
- Most of the solvers provided source codes and models

Winning Techniques and Solvers

Reinforcement Learning in Main Track (MapleCOMPSP)

Branching heuristics in SAT solvers have been similar for over a decade using Variable State Independent Decaying Sum [Moskewicz, Madigan, Zhao, Zhang, and Malik '01].

The SAT Competition 2016 winner used an alternative branching heuristics based on reinforcement learning [Liang, Ganesh, Poupart, Czarnecki '16].



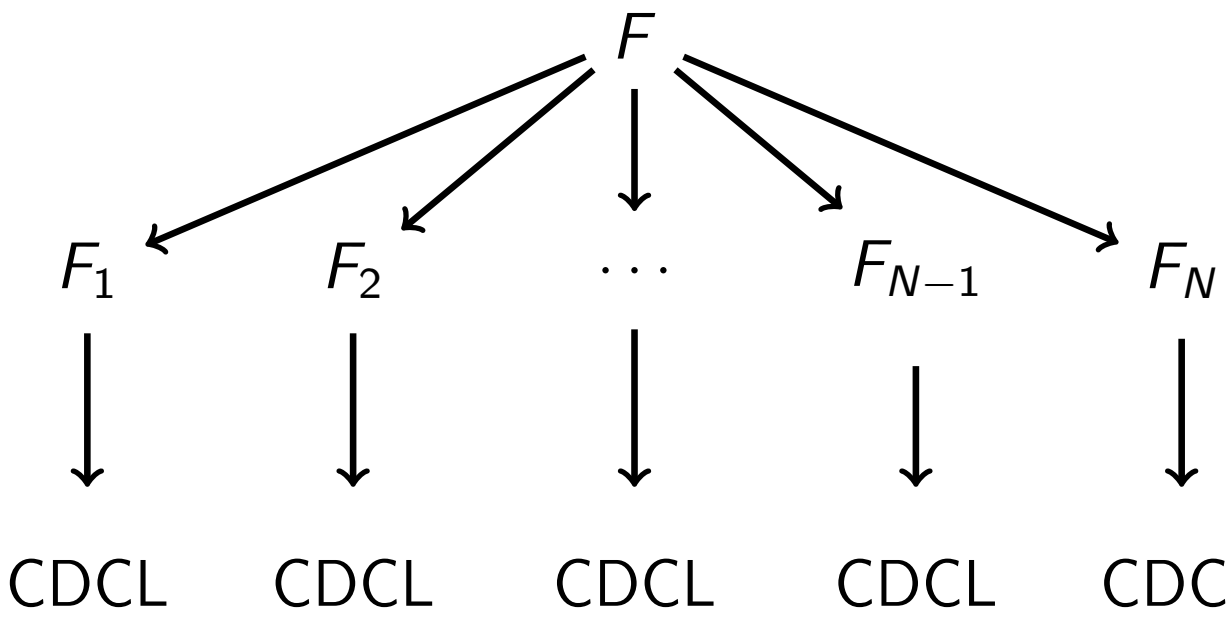
The reward for each variable is the fraction of conflict clauses in which it was involved while being assigned.

Cube-and-Conquer in Parallel Track (Treengeling) I

The Cube-and-Conquer paradigm has two phases:

Cube First a look-ahead solver is employed to split the problem — the splitting tree is cut off appropriately.

Conquer At the leaves of the tree, CDCL solvers are employed.



The Cube-and-Conquer solver Treengeling by Armin Biere overwhelmingly won the Parallel track of the competition.

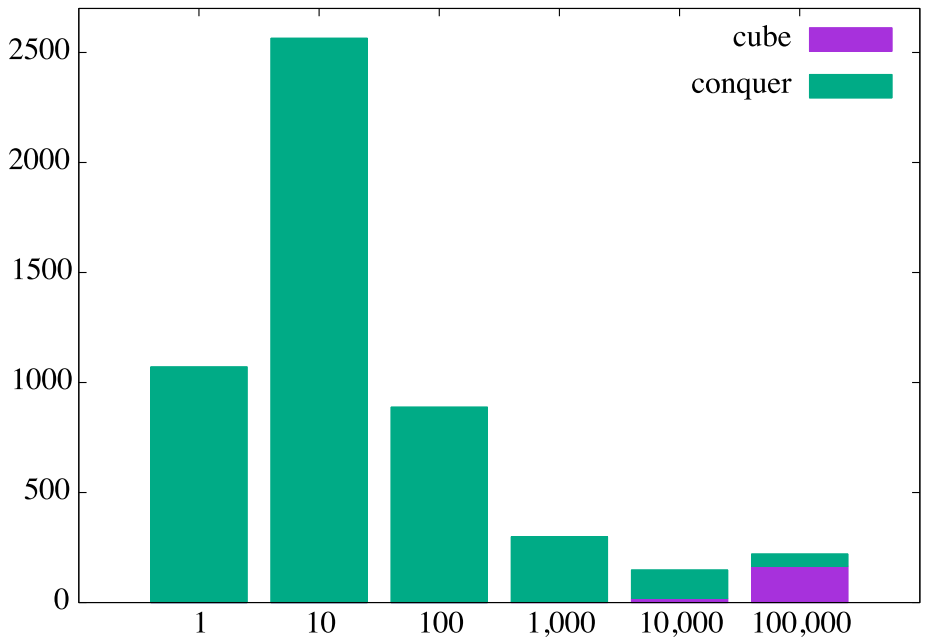
Cube-and-Conquer in Parallel Track (Treengeling) II

Let N denote the number of leaves in the cube-phase:

- the case $N = 1$ means pure CDCL,
- and very large N means pure look-ahead.

Consider the total run-time (y-axis) in dependency on N (x-axis):

- typically, first it increases, then
- it decreases, but only for a large number of subproblems!



Example with Schur Triples and 5 colors: a formula with 708 vars and 22608 clauses.

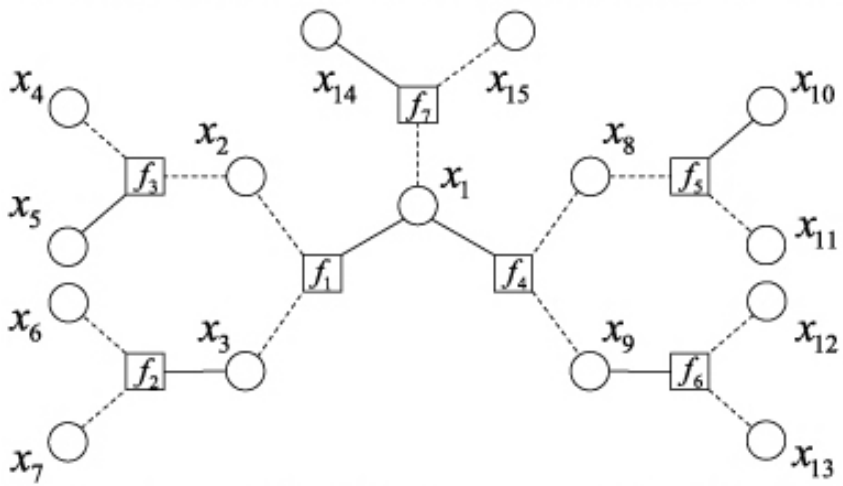
Subproblems are solved with Glucose 3.0 as conquer solver.

The performance tends to be optimal when the cube and conquer times are comparable.

Message Passing in Random Track (Dimetheus)

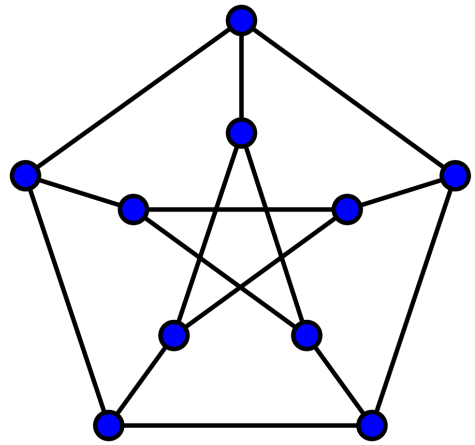
Local search algorithms used to be relatively simple: randomly pick a falsified clause and flip one of the literals to true.

The winner of the Random track, Dimetheus [Gableske '16], is more complicated and uses message passing to compute good starting assignments for conventional SLS algorithms.



Symmetry Breaking in NoLimit (BreakIDCOMiniSatPS)

Several benchmarks contain symmetries — in particular in the "crafted" category. Adding symmetry-breaking predicates could significantly improve performance. However, detecting symmetries could be costly as well.



The BreakIDCOMiniSatPS solver [Devriendt, Bogaerts, Bruynooghe, and Denecker '16] was the only solver during this competition that included symmetry breaking. It allowed them to solve some hard benchmarks.

BreakIDCOMiniSatPS did not participate in the Main Track because the authors had no time to implement proof logging.

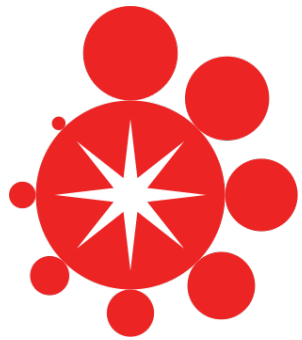
Further Information and Acknowledgements

Further Information:

- Homepage: `satcompetition.org/`
- Proceedings with solver and benchmark descriptions `helda.helsinki.fi/handle/10138/164630`

Acknowledgments

- Thanks for all the benchmarks
- Thanks to Aaron Stump and StarExec
- Thanks to TACC for the Lonestar5 resources



SAT Competition 2016: Recent Developments

Tomás Balyo Marijn J.H. Heule Matti Järvisalo

UNIVERSITY OF HELSINKI

AAAI-17

February 6, 2017