

# Empirical Study of the Anatomy of Modern Sat Solvers Restarts.

*H.Katebi, K.Sakallah, J.Marques-Silva*

(presented by Michalis Famelis)

February 3, 2012

# Introductions

Introduction

Features

Setup

Results

Insights

Conclusion

- Paper published in SAT'11.
- H.Katebi, student of K.Sakallah.
- K.Sakallah and J.Marques-Silva: authors of GRASP, that introduced Conflict-Driven Clause Learning.

# Introduction

Introduction

Features

Setup

Results

Insights

Conclusion

- SAT: a hard problem, but with very important applications.
- SAT Solvers:
  - Major advancements in the last 15 years!
  - Contemporary solvers are based on DPLL.
  - Generally can scale up to several million vars/clauses.
- Most modern SAT solvers share a few important features\*.
- Despite dramatic progress:
  - Unpredictable failures in practical problems.
  - Usually not clear which features make problems tractable.
  - Researchers unclear about features' relative importance.

# Contributions

- Study the relative contribution of features of modern SAT solvers.
- Experimentally verify anecdotal opinions held by the research community.
- Stimulate the interest of theoretical computer scientists to study the *“remarkable success and unexpected failures of modern SAT solvers”*.
- Experimentation with MiniSAT.

- ① Introduction
- ② Survey of CDCL Solver features
- ③ Experimental Setup
- ④ Experimental Results
- ⑤ Insights Gained
- ⑥ Conclusion

# Basic Framework

Introduction

**Features**

Setup

Results

Insights

Conclusion

DPLL algorithm:

- Branching.
- Unit propagation.
- Backtracking.

Modern SAT solvers extend this basic framework with algorithmic enhancements and heuristics.

# Features in MiniSAT (1/2)

Introduction

Features

Setup

Results

Insights

Conclusion

## Conflict-driven clause learning

- Introduced by the GRASP solver.
- Analyze conflict to find the *effective* learned clause.
  - A small set of assignments sufficient to expose the conflict.
- Modern solvers also do Conflict Clause Minimization (local/recursive)

## Random search restarts

- Complete search algorithms: heavy-tailed cost distribution.
- Unpredictable effect on run times.
- Solve by randomization. In SAT solvers: randomly backtrack to root (MiniSAT uses the Luby sequence).

# Features in MiniSAT (2/2)

Introduction

Features

Setup

Results

Insights

Conclusion

## Two-Literal Watching

- Introduced in the Chaff solver.
- Update status of a clause only when one of its watched literals is assigned 0.
  - Lazy data structure greatly improving Boolean constraint propagation.

## Conflict-based adaptive branching

- Branching heuristics for minimizing decision steps and overhead.
- DLIS from GRASP, VSIDS from Chaff, Literal Phase Saving from RSat
- MiniSAT: Literal Phase Saving and adapted VSIDS
  - “Activities”: counters for vars instead of literals.

# Secondary Features of MiniSAT

Introduction

Features

Setup

Results

Insights

Conclusion

MiniSAT can be configured for:

- Percentage of random decisions for VSIDS.
- Random initialization of VSIDS Activities.
- Minimization of effective learned clause (none/local/recursive).
- Level of literal phase saving.
- Randomization of restart sequence (Luby/exponential function).

(For the main features MiniSAT had to be instrumented, rather than configured.)

- ① Introduction
- ② Survey of CDCL Solver features
- ③ Experimental Setup**
- ④ Experimental Results
- ⑤ Insights Gained
- ⑥ Conclusion

# Experimental Strategy

Introduction

Features

Setup

Results

Insights

Conclusion

Two experiments:

- ① Study of impact of each of the 4 major features.
- ② Study of 10 configurations of the 5 secondary features.

Experimental approach:

- Subjects: set of 10000 real-world benchmarks.
- Each subject: 10 random reorderings of the CNF.
- Base case: fully enabled MiniSAT.
- Disable one feature at a time and compare with base case.
- For each configuration count the number of instances solved in under 1000 seconds.
- Total: 150000 experimental runs.

# Experimental Subjects

Introduction

Features

Setup

Results

Insights

Conclusion

- 10000 benchmarks from 12 application areas, since early 1990s.
- Criteria for selection:
  - Domains where SAT solving has been successful.
  - Benchmarks from SAT Competitions and Races.
  - A “reasonable” number of “easy” instances.
  - Weighting numbers in each area by relative success of application of SAT.
- Excluded random benchmarks: well studied elsewhere and real-word not as random.

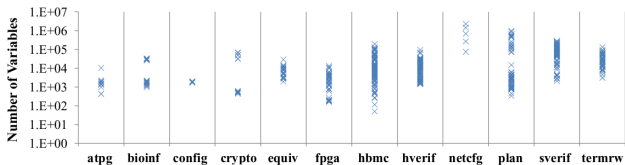
# Subjects By Area

Introduction  
Features  
Setup  
Results  
Insights  
Conclusion

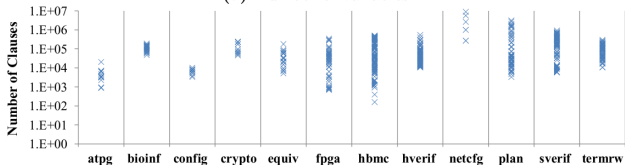
**Table 1.** Benchmark families

Family	Instances	SAT	UNS	UNK	Description
<b>atpg</b>	100	28	72	0	Circuit testing
<b>bioinf</b>	30	8	12	10	Bioinformatics
<b>config</b>	50	15	35	0	Product configuration
<b>crypto</b>	30	26	3	1	Cryptanalysis
<b>equiv</b>	30	5	25	0	Equivalence checking
<b>fpga</b>	50	25	22	3	FPGA routing
<b>hbmc</b>	250	88	146	16	Hardware bounded model checking
<b>hverif</b>	200	125	75	0	Hardware verification
<b>netcfg</b>	10	7	2	1	Network configuration
<b>plan</b>	80	51	24	5	Planning
<b>sverif</b>	120	57	52	11	Software verification
<b>termrw</b>	50	26	22	2	Term rewriting
<b>Total:</b>	1000	461	490	49	

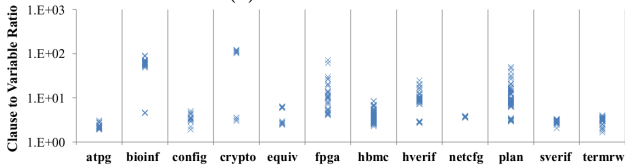
# Statistics of Subjects



(a) Number of variables



(b) Number of clauses



(c) Clause-to-variable ratio

Introduction

Features

Setup

Results

Insights

Conclusion

- ① Introduction
- ② Survey of CDCL Solver features
- ③ Experimental Setup
- ④ Experimental Results**
- ⑤ Insights Gained
- ⑥ Conclusion

# Experiment 1: Results

Introduction

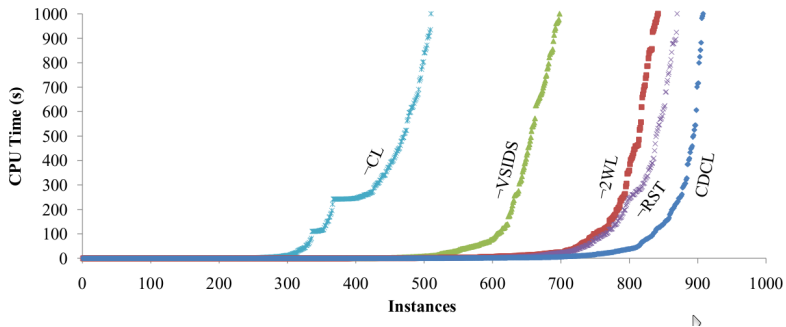
Features

Setup

Results

Insights

Conclusion



# Experiment 1: By Application Area

Introduction

Features

Setup

Results

Insights

Conclusion

**Table 2.** Number of instances solved by disabling major CDCL features

Family	Runs	$\neg$ CL	$\neg$ VSIDS	$\neg$ 2WL	$\neg$ RST	CDCL
atpg	1000	965	1000	1000	1000	1000
bioinf	300	19	34	88	141	150
config	500	472	500	500	500	500
crypto	300	52	22	113	235	237
equiv	300	50	92	187	224	231
fpga	500	325	403	444	441	470
hbmc	2500	762	1872	2241	2307	2333
hverif	2000	1413	1700	1934	1967	1984
netcfg	100	0	20	60	74	87
plan	800	327	449	559	564	650
sverif	1200	336	592	937	754	1006
termrw	500	116	248	346	446	420
Total:	10000	4837	6932	8409	8653	9068

## Experiment 2: Results

**Table 3.** Number of instances solved under different MiniSAT options

Family	CDCL	rnd-freq				rnd-init	ccmin-mode		phase-saving		no-luby
		25	50	75	100		none	basic	none	limited	
atpg	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
bioinf	150	133	107	72	46	150	139	149	150	150	148
config	500	500	500	500	50	500	500	500	500	500	500
crypto	237	67	63	49	35	228	214	223	219	234	<b>243</b>
equiv	231	221	216	181	162	231	220	222	224	<b>235</b>	224
fpga	470	456	453	444	421	470	<b>471</b>	468	454	463	462
hbmc	2333	2328	2322	2225	2057	2328	2328	2333	2318	2326	2315
hverif	1984	<b>1989</b>	<b>1993</b>	<b>1997</b>	1949	1984	<b>1993</b>	<b>1991</b>	1971	<b>1997</b>	1960
netcfg	87	76	75	60	72	80	76	77	74	74	67
plan	650	619	593	526	490	647	637	640	606	636	586
sverif	1006	915	858	762	302	1004	1003	996	976	967	944
termrw	420	416	407	378	291	420	416	417	<b>426</b>	<b>424</b>	<b>444</b>
Total:	9068	8720	8587	8194	7325	9042	8997	9016	8918	9006	8893

(Bold: better performance than default.)

- ① Introduction
- ② Survey of CDCL Solver features
- ③ Experimental Setup
- ④ Experimental Results
- ⑤ Insights Gained**
- ⑥ Conclusion

# Experiment 1

Introduction

Features

Setup

Results

Insights

Conclusion

- Importance of major features:  
CL > VSIDS > 2WL > RST
- Algorithmic optimizations vs heuristics:
  - Enabling CL and 2WL consistently improves results.
  - The performance of VSIDS and RST was more variable.
- Restarts are interesting:
  - Relatively moderate performance gain.
  - Surprisingly restarts also improve things for UNSAT problems.
  - Not as reliable (in some cases  $\neg$ RS worked better).
- Combination of all four features yields best performance.

## Experiment 2

- Generally, default settings perform better.
- In some specific cases, moderate gains.
- Some surprising results:
  - 100% random better than  $\neg$ VSIDS (i.e., DLIS).
  - But DLIS solved many that random could not (i.e., is not subsumed).
  - Need for more investigation of the specific instances and their characteristics.
- Harder to articulate general conclusions as in Exp.1.

- ① Introduction
- ② Survey of CDCL Solver features
- ③ Experimental Setup
- ④ Experimental Results
- ⑤ Insights Gained
- ⑥ Conclusion

# Paper's Conclusions

- Few explanations about why CDCL solvers work/fail when they do.
- Notable exception: proof that Clause Learning is more powerful than regular resolution.
- This paper: preliminary study of impact of features.
- Important future work: characteristics of inputs.
  - Symmetries in CNF formulas.
  - Cut width of graph representation of CNF.
  - Scale-free graph structure of real-world instances.
- So to inform creation of algorithmic improvements and “domain specific” SAT solvers.

# My Remarks

- Concise overview of solvers' features.
- Methodology and results are definitely convincing.
- Research questions and relation to experiments are not clearly explicated.
- But where is the “Threats to Validity” section?
- E.g., why was MiniSAT specifically chosen?
- Very interesting also from the reverse perspective: what domains SAT is good for?