

# Understanding and using SAT solvers

A practitioner perspective

Daniel Le Berre<sup>1</sup>

CRIL-CNRS UMR 8188

Summer School 2009: Verification Technology, Systems &  
Applications

Nancy, October 12-16, 2009

---

1. Contains material provided by Prof. Joao Marques Silva



1/211



Anatomy of a modern CDCL SAT solver (continued)

Certifiable Unsat answers

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT

# First UIP conflict analysis based on Resolution !

Perform resolution steps in reverse order of the assignments.

Suppose

$decisionLevel(f) = x$  and

$decisionLevel(c) = y$  with  $x > y$ .

Propagations deriving from  $a$  :  $g, b, d, e$

Reasons of the propagations :

$$= (a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg b \vee c \vee d) \wedge (\neg b \vee e)$$

Conflicting clause (resolvent);

$$(\neg d @_x \vee \neg e @_x \vee f @_x)$$

# First UIP conflict analysis based on Resolution !

Perform resolution steps in reverse order of the assignments.

Suppose

$decisionLevel(f) = x$  and

$decisionLevel(c) = y$  with  $x > y$ .

Propagations deriving from  $a$  :  $g, b, d, e$

Reasons of the propagations :

$$= (a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b) \wedge (\neg b \vee c \vee d)$$

Conflicting clause (resolvent);

$$(\neg b @_x \vee \neg d @_x \vee f @_x)$$

# First UIP conflict analysis based on Resolution !

Perform resolution steps in reverse order of the assignments.

Suppose

$decisionLevel(f) = x$  and

$decisionLevel(c) = y$  with  $x > y$ .

Propagations deriving from  $a : g, b, d, e$

Reasons of the propagations :

$$= (a \vee c \vee f) \wedge (\neg a \vee g) \wedge (\neg g \vee b)$$

Conflicting clause (resolvent);

$$(\neg b @_x \vee c @_y \vee f @_x)$$

# First UIP conflict analysis based on Resolution !

Perform resolution steps in reverse order of the assignments.

Suppose

$decisionLevel(f) = x$  and

$decisionLevel(c) = y$  with  $x > y$ .

Propagations deriving from  $a$  :  $g, b, d, e$

Reasons of the propagations :

$$= (a \vee c \vee f) \wedge (\neg a \vee g)$$

Conflicting clause (resolvent);

$$(\neg g @ x \vee c @ y \vee f @ x)$$

# First UIP conflict analysis based on Resolution !

Perform resolution steps in reverse order of the assignments.

Suppose

$decisionLevel(f) = x$  and

$decisionLevel(c) = y$  with  $x > y$ .

Propagations deriving from  $a$  : g,b,d,e

Reasons of the propagations :

$$= (a \vee c \vee f)$$

Conflicting clause (resolvent);

$$(\neg a@x \vee c@y \vee f@x)$$

# First UIP conflict analysis based on Resolution !

Perform resolution steps in reverse order of the assignments.

Suppose

$decisionLevel(f) = x$  and

$decisionLevel(c) = y$  with  $x > y$ .

Propagations deriving from  $a$  :  $g, b, d, e$

Reasons of the propagations :

Conflicting clause (resolvent) ;

$$(c @ y \vee f @ x)$$

First UIP ! only one literal at decision level  $x$  left.

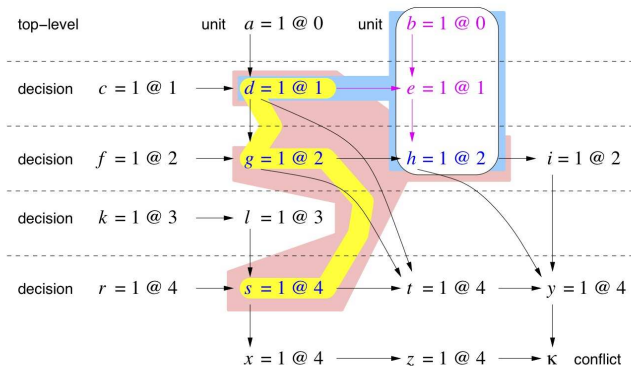
# Conflict Clause minimization

Minisat 1.13, N. Sörensson, A. Biere. Minimizing Learned Clauses. In Proc. 12th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'09), Lecture Notes in Computer Science (LNCS) vol. 5584, pages 237-243, Springer 2009.

- ▶ Clauses generated using the 1st UIP scheme can be simplified
- ▶ Using simple direct self subsumption (direct dependencies among the clause's literals outside current decision level) :

$$\text{self subsumption: } \frac{x_1 @ 1 \vee x_2 @ 1 \vee x_3 @ 2}{x_1 @ 1 \vee x_3 @ 2} \quad \frac{x_1 @ 1 \vee \neg x_2 @ 1}{x_1 @ 1 \vee x_3 @ 2}$$

- ▶ Using a chain of resolution steps



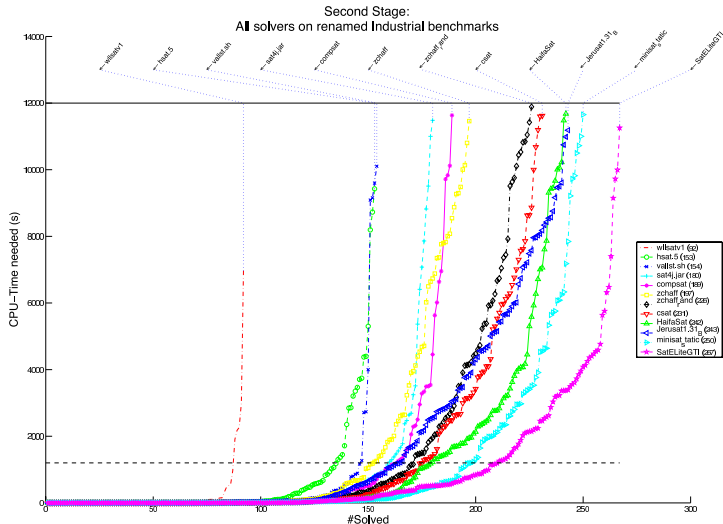
$$\begin{array}{c}
 \frac{(\bar{e} \vee \bar{g} \vee h) \quad (\bar{d} \vee \bar{g} \vee \bar{s} \vee \bar{h})}{(\bar{d} \vee \bar{b} \vee e) \quad (\bar{e} \vee \bar{d} \vee \bar{g} \vee \bar{s})} \\
 \frac{(b) \quad (\bar{b} \vee \bar{d} \vee \bar{g} \vee \bar{s})}{(\bar{d} \vee \bar{g} \vee \bar{s})}
 \end{array}$$

# Preprocessing

Niklas Eén, Armin Biere : Effective Preprocessing in SAT Through Variable and Clause Elimination. SAT 2005 : 61-75

- ▶ Variable elimination
  - ▶ as in DP60 if the number of clauses does not increase
  - ▶ by substitution if a definition such as  $x \leftrightarrow y_1 \vee \dots \vee y_n$  or  $x \leftrightarrow y_1 \wedge \dots \wedge y_n$  is detected.
- ▶ Clause subsumption
  - ▶ self subsumption
  - ▶ classical subsumption
- ▶ SatELite : de-facto standard pre-processor since 2005
- ▶ Included in Minisat 2 (better integration with the SAT solver)
- ▶ Still used by SAT solver designers that do not want to implement their own

# Clause Minimization and Preprocessing @SAT COMP. 2005



# Phase Saving

Knot Pipatsrisawat, Adnan Darwiche : A Lightweight Component Caching Scheme for Satisfiability Solvers. SAT 2007 : 294-299

- ▶ To concentrate on a single component, keep track of the phase of assigned literals when restarting.
- ▶ Always branch first on the recorded phase when taking a decision.
- ▶ A **small change** in the code of the solver, a **big improvement** in practice (at least for pure SAT :) ) !
- ▶ Note : RSAT forgets the phase after a while ...

# Rapid Restarts

Jinbo Huang : The Effect of Restarts on the Efficiency of Clause Learning. IJCAI 2007 : 2318-2323

- ▶ Restarts bounds usually grow slowly until being large enough to ensure completeness
- ▶ Different restart strategies make huge differences depending of the benchmarks
- ▶ Rapid Restarts strategies usually a good companion for Phase Saving



# Armin Inner/Outer rapid restarts

Armin Biere : PicoSAT Essentials. JSAT 4(2-4) : 75-97 (2008)

```
int inner = 100, outer = 100;
int restarts = 0, conflicts = 0;

for (;;) {
    ... // run SAT core loop for inner conflicts

    restarts++; conflicts += inner;
    if (inner >= outer) {
        outer *= 1.1; inner = 100;
    else
        inner *= 1.1;
}
```

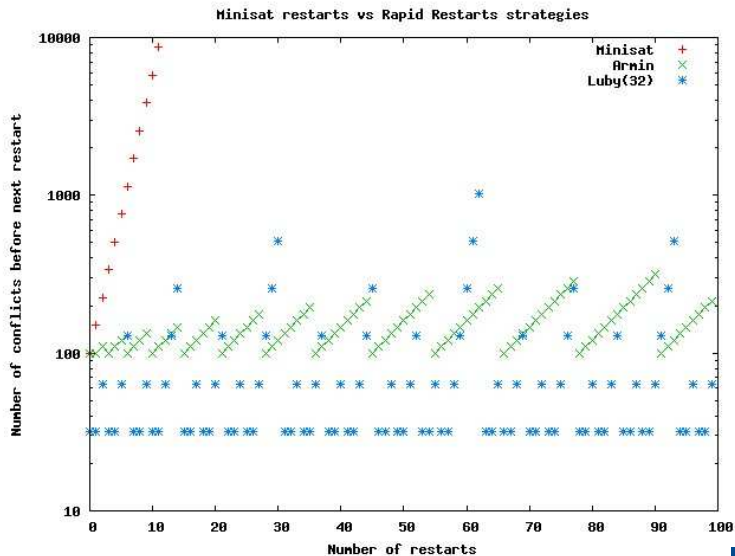
# Luby series rapid restarts

Michael Luby, Alistair Sinclair, David Zuckerman : Optimal Speedup of Las Vegas Algorithms. ISTCS 1993 : 128-133

$$t_i = \begin{cases} 2^{k-1}, & \text{if } i = 2^k - 1; \\ t_{i-2^{k-1}+1}, & \text{if } 2^{k-1} \leq i < 2^k - 1. \end{cases}$$

- ▶ Used in SATZ\_rand and Relsat\_rand within Blackbox [Kautz,Selman 99]
- ▶ Used in Tinisat and RSAT in 2007 with factor 512.

# Comparison of a few different restarts strategies



# Effect of Rapid Restarts in SAT4J

Using the SAT Race 2006 benchmarks set (100 benchmarks), with a timeout of 900 seconds per benchmark :

| Configuration                    | Total | SAT | UNSAT | Time |
|----------------------------------|-------|-----|-------|------|
| MiniSAT                          | 58    | 29  | 29    | 835  |
| Luby (factor 32)                 | 59    | 24  | 35    | 790  |
| Luby (factor 512, no PS, no CCM) | 48    | 19  | 29    | 947  |
| Luby (factor 512, no CCM)        | 55    | 26  | 29    | 866  |
| Luby scheme (factor 512)         | 61    | 29  | 32    | 788  |
| Armin                            | 61    | 27  | 34    | 790  |

Time is given in minutes, on a PIV 3GHz, 1.5GB of RAM, Java 6 VM under Mandriva Linux 2007.1.

# Adaptative restarts during the SAT 2009 competition

**picosat** A. Biere. Adaptive Restart Control for Conflict Driven SAT Solvers. In Proc. 11th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'08), Lecture Notes in Computer Science (LNCS) vol. 4996, Springer 2008.

**Minisat09z** Carsten Sinz, Markus Iser : Problem-Sensitive Restart Heuristics for the DPLL Procedure. SAT 2009 : 356-362

**Lysat** Youssef Hamadi, Said Jabbour, and Lakhdar Sais. ManySAT : a Parallel SAT Solver. Volume 6 (2009), pages 245-262.

**glucose** Predicting Learnt Clauses Quality in Modern SAT Solver G. Audemard, L. Simon, in Twenty-first International Joint Conference on Artificial Intelligence (IJCAI'09), july 2009.



Anatomy of a modern CDCL SAT solver (continued)

**Certifiable Unsat answers**

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT

- ▶ A common proof format for UNSAT proofs exists since 2005.
- ▶ Proposed by Allen van Gelder
- ▶ Resolution and Trace format in 2005
- ▶ Reverse Unit Propagation (RUP) in 2007
- ▶ All the details at  
<http://users.soe.ucsc.edu/~avg/ProofChecker/>

# The proof formats

- ▶ There are three possibilities for the proof format :
  - Resolution format** Each resolution steps are provided and resolvents are explicitly provided.
  - Trace format** Only the resolution steps are provided. Not the resolvents. More compact format.
  - RUP format** Only derived clauses are provided. The checker negate them to derive the empty clause.
- ▶ A checker can check a proof using the resolution format.
- ▶ The trace format can be converted into the resolution format.

# Examples of the proof formats

## Example (Original input file)

```
p cnf 2 3
1 -2 0
1 2 0
-1 0
```

## Example (resolution format)

```
4 2 1 2 2 1 1 2
5 1 3 4 0 0
```

## Example (trace format)

```
4 2 1 2
5 1 3 4
```

# Zchaff certificates (resolution) on easy UNSAT IBM benchmarks (SAT COMP. 2005)

| Bench            | Cert     | Orig | Cost (%) | Size (KB) |
|------------------|----------|------|----------|-----------|
| 01_SAT_dat.k10   | 0,9      | 0,23 | 284,19   | 12656     |
| 07_SAT_dat.k30   | 4,21     | 3,41 | 23,41    | 11353     |
| 07_SAT_dat.k35   | 5,11     | 4,29 | 19,02    | 11836     |
| 18_SAT_dat.k10   | 35,93    | 0,32 | 10957,58 | (711381)  |
| 18_SAT_dat.k15   | (103,42) | 3,64 | 2743,20  | -         |
| 1_11_SAT_dat.k10 | 3,59     | 0,64 | 462,38   | 54786     |
| 1_11_SAT_dat.k15 | 30,08    | 3,65 | 724,87   | (525236)  |
| 20_SAT_dat.k10   | 3,02     | 0,29 | 931,40   | 54839     |
| 23_SAT_dat.k10   | 0,78     | 0,23 | 236,80   | 10251     |
| 23_SAT_dat.k15   | (98,71)  | 1,23 | 7913,41  | -         |
| 26_SAT_dat.k10   | 12,34    | 7,95 | 55,27    | 148       |
| 2_14_SAT_dat.k10 | 15,09    | 0,48 | 3017,36  | (286783)  |
| 2_14_SAT_dat.k15 | (102,2)  | 1,49 | 6778,59  | -         |

# Size of the "easy" IBM benchmarks

| Benchmark              | # var | # clauses | Size     |
|------------------------|-------|-----------|----------|
| 01_SAT_dat.k10.cnf     | 9275  | 38802     | 12656    |
| 07_SAT_dat.k30.cnf     | 11081 | 31034     | 11353    |
| 07_SAT_dat.k35.cnf     | 12116 | 33469     | 11836    |
| 1_11_SAT_dat.k10.cnf   | 28280 | 111519    | 54786    |
| (1_11_SAT_dat.k15.cnf) | 44993 | 178110    | (525236) |
| (18_SAT_dat.k10.cnf)   | 17141 | 69989     | (711381) |
| —18_SAT_dat.k15.cnf—   | 25915 | 106325    |          |
| 20_SAT_dat.k10.cnf     | 17567 | 72087     | 54839    |
| (2_14_SAT_dat.k10.cnf) | 12859 | 49351     | (286783) |
| —2_14_SAT_dat.k15.cnf— | 20302 | 78395     |          |
| 23_SAT_dat.k10.cnf     | 18612 | 76086     | 10251    |
| —23_SAT_dat.k15.cnf—   | 29106 | 119635    |          |
| 26_SAT_dat.k10.cnf     | 55591 | 277611    | 148      |

- ▶ zChaff : Resolution format  
<http://www.princeton.edu/~chaff/zchaff.html>
- ▶ picosat : Resolution and RUP format  
<http://fmv.jku.at/picosat/>
- ▶ boolforce : Resolution format  
<http://fmv.jku.at/booleforce>

# Agenda

Anatomy of a modern CDCL SAT solver (continued)

Certifiable Unsat answers

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT



# Recent CDCL solvers taking advantage of multi-core technology

- MiraXT** Tobias Schubert, Matthew Lewis, and Bernd Becker.  
PaMiraXT : Parallel SAT Solving with Threads and Message Passing. Volume 6 (2009), pages 203-222.
- pMinisat** Geoffrey Chu, Aaron Harwood, and Peter J. Stuckey.  
Cache Conscious Data Structures for Boolean Satisfiability Solvers. Volume 6 (2009), pages 99-120.  
Second place during SAT Race 2008.
- ManySAT** Youssef Hamadi, Said Jabbour, and Lakhdar Sais.  
ManySAT : a Parallel SAT Solver. Volume 6 (2009), pages 245-262. Winner of the SAT Race 2008.  
Several solvers launched in parallel with clause sharing.

# SAT Race 2008 results, parallel track

|                                        | ManySat  | pMiniSat | MiraXT   |
|----------------------------------------|----------|----------|----------|
| #solved:                               | 90       | 85       | 73       |
| #solved SAT/UNSAT:                     | 45/45    | 44/41    | 43/30    |
| #solved in all runs:                   | 79       | 84       | 66       |
| #solved in <sup>2</sup> 900s CPU time: | 69       | 72       | 53       |
| rank:                                  | <b>1</b> | <b>2</b> | <b>3</b> |

# SAT Race 2008 results, main track

|                    | MiniSat 2.1 | pMiniSat* | Barcelologic | Rsat  | MXC   | preSAT | CMUSAT | kw    | picosat | Eureka |
|--------------------|-------------|-----------|--------------|-------|-------|--------|--------|-------|---------|--------|
| #solved:           | 81          | 79        | 77           | 76    | 74    | 72     | 72     | 70    | 70      | 69     |
| #solved SAT/UNSAT: | 43/38       | 39/40     | 35/42        | 36/40 | 40/34 | 38/34  | 40/32  | 32/38 | 40/30   | 29/40  |
| rank by solved:    | 1           | 2         | 3            | 4     | 5     | 6      | 6      | 8     | 8       | 10     |
| speed points:      | 7.26        | 5.61      | 5.10         | 5.04  | 4.79  | 4.73   | 4.72   | 5.19  | 5.16    | 5.44   |
| rank by speed:     | 1           | 2         | 6            | 7     | 8     | 9      | 10     | 4     | 5       | 3      |
| total score:       | 88.26       | 84.61     | 82.10        | 81.04 | 78.79 | 76.73  | 76.72  | 75.19 | 75.16   | 74.44  |
| rank:              | 1           | 2         | 3            | 4     | 5     | 6      | 7      | 8     | 9       | 10     |

|                    | ManySat* | LocalMinisat | MiraXT* | SATzilla | eSAT  | Spear | Tinisat | clasp | SAT4J |
|--------------------|----------|--------------|---------|----------|-------|-------|---------|-------|-------|
| #solved:           | 68       | 64           | 63      | 60       | 60    | 60    | 56      | 54    | 49    |
| #solved SAT/UNSAT: | 30/38    | 32/32        | 37/26   | 30/30    | 29/31 | 30/30 | 28/28   | 23/31 | 26/23 |
| rank by solved:    | 11       | 12           | 13      | 14       | 14    | 14    | 17      | 18    | 19    |
| speed points:      | 3.99     | 3.68         | 4.26    | 4.18     | 3.81  | 3.32  | 3.08    | 3.19  | 2.43  |
| rank by speed:     | 13       | 15           | 11      | 12       | 14    | 16    | 18      | 17    | 19    |
| total score:       | 71.99    | 67.68        | 67.26   | 64.18    | 63.81 | 63.32 | 59.08   | 57.19 | 51.43 |
| rank:              | 11       | 12           | 13      | 14       | 15    | 16    | 17      | 18    | 19    |

# CPU time vs wall clock time

- ▶ SAT race 2008 parallel track is using wall clock time to stop the solvers.
- ▶ The solvers were ran 3 times : one benchmark solved if solved in at least one run.
- ▶ Other decisions taken for the SAT competition ...

We'll have to deal with multicores computers, let's start thinking about it.

- ▶ Naive parallelization should not work on many cores : memory access is a hard bottleneck for SAT solvers
- ▶ We would like to observe if multithreaded solvers scale well on a machine with 16 cores.

We'll have to deal with multicores computers, let's start thinking about it.

- ▶ Naive parallelization should not work on many cores : memory access is a hard bottleneck for SAT solvers
- ▶ We would like to observe if multithreaded solvers scale well on a machine with 16 cores.
- ▶ Problem : not enough competitors !

# Parallel track : the competitors

| Solver name                        | Authors                                                             |
|------------------------------------|---------------------------------------------------------------------|
| No limit on threads                |                                                                     |
| gNovelty+-T<br>satake<br>ttsth-5-0 | Duc-Nghia Pham and Charles Gretton<br>Kota Tsuyuzaki<br>Ivor Spence |
| Limited to 4 threads               |                                                                     |
| ManySAT 1.1 aimd 0/1/2             | Youssef Hamadi, Saïd Jabbour, Lakhdar Saïs                          |

# Parallel track : the settings

- ▶ Parallel solvers ran on 3 different computers :
  - 2 processors with the main track, first stage, at CRIL.
  - 4 cores on a cluster of 4 core computers at LRI.
  - 16 cores on one specific 16 core computer at LRI.
- ▶ The solvers are given 10000s CPU time to be shared by the different threads : to be compared with the second stage of the main track.
- ▶ We ran only solvers able to use the 16 cores on the 16 core computer.

# Parallel track : the results

| Solver                       | Total | SAT | UNSAT | CPU Time  |
|------------------------------|-------|-----|-------|-----------|
| Application                  |       |     |       |           |
| 2 Threads (CRIL)             |       |     |       |           |
| ManySAT 1.1 aimd 1           | 193   | 71  | 122   | 173344.71 |
| 4 Threads (LRI)              |       |     |       |           |
| ManySAT 1.1 aimd 1           | 187   | 68  | 119   | 112384.15 |
| ManySAT 1.1 aimd 0           | 185   | 69  | 116   | 103255.01 |
| ManySAT 1.1 aimd 2           | 181   | 65  | 116   | 104021.63 |
| satake                       | 118   | 52  | 66    | 50543.61  |
| ttsth-5-0                    | 7     | 3   | 4     | 2274.38   |
| 16 Threads (LRI)             |       |     |       |           |
| satake                       | 106   | 40  | 66    | 130477.38 |
| ttsth-5-0                    | 7     | 3   | 4     | 9007.53   |
| Random                       |       |     |       |           |
| gNovelty+-T (2 threads CRIL) | 314   | 314 | -     | 143439.69 |
| gNovelty+-T (4 threads LRI)  | 296   | 296 | -     | 95118.33  |
| gNovelty+-T (16 threads LRI) | 237   | 237 | -     | 68173.49  |

# Agenda

Anatomy of a modern CDCL SAT solver (continued)

Certifiable Unsat answers

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT



# Implementation details matters !

- ▶ A big part of the knowledge for efficient SAT solver designed spread thanks to **source code** !
- ▶ Since the beginning the SAT community has been keen to run competitive events ;
  - ▶ 1992 : Paderborn
  - ▶ 1993 : **The second DIMACS challenge** [standard input format] Johnson, D. S., & Trick, M. A. (Eds.). (1996). Cliques, Coloring and Satisfiability : Second DIMACS Implementation Challenge, Vol. 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science. AMS.
  - ▶ 1996 : Beijing
  - ▶ 1999 : SATLIB
  - ▶ 2000 : SAT-Ex
  - ▶ Since 2002 : **yearly competition (or Race)**

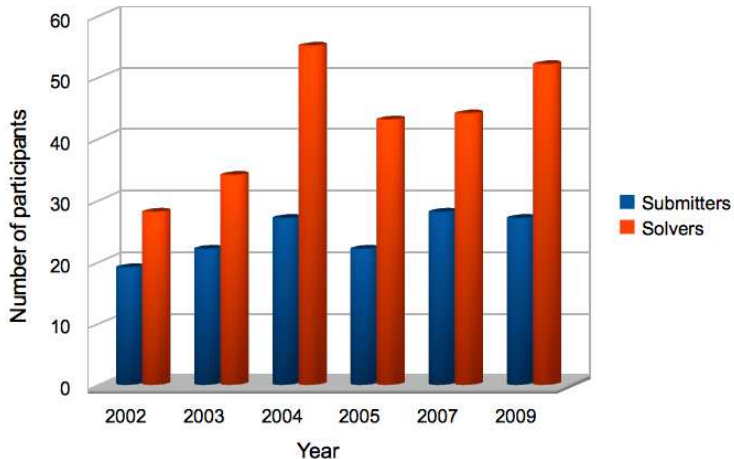
# DP 60 and DLL62 : the very first SAT competition !

*In the present paper, a uniform proof procedure for quantification theory is given which is feasible for use with some rather complicated formulas and which does not ordinarily lead to exponentiation. The superiority of the present procedure over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes [Davis and Putnam, 1960].*

*The well-formed formula (...) which was beyond the scope of Gilmore's program was proved in under two minutes with the present program [Davis et al., 1962]*

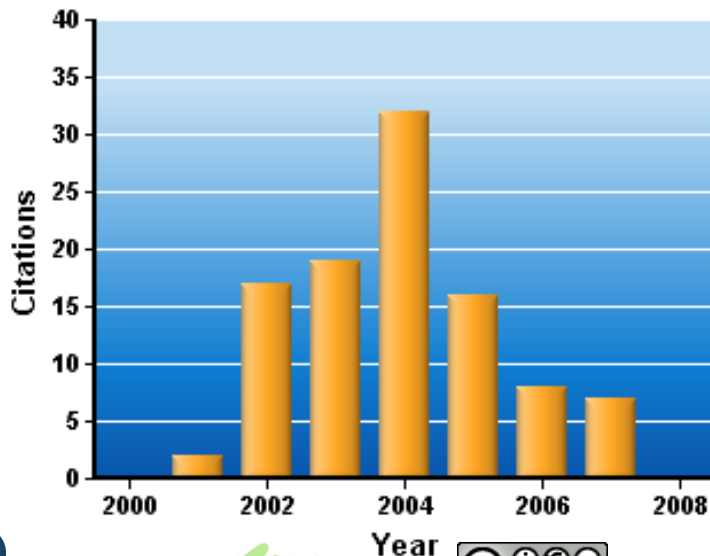
# Evolution of the participation to the SAT competition

## Evolution of the audience of the SAT competition



# Evolution of the citations of Chaff's paper

<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.7475>



# Impressive results of the SAT 2009 competition

SAT 2009 Competition: main competition (phase 1): solvers results per benchmarks

http://www.cril.univ-artois.fr/SAT09/results/bench.php?dev=22&idbench=70953

SAT4J - Recherche Google Graphviz Apple - Supp...00 Slide ... Nokia 6600 S... | Paul Bain Extending D... Conditions AndrewEscobar.com

Bug 291985 Submitted - [compile... SAT 2009 Competition: main com...

## General information on the benchmark

|                                                                 |                                                           |
|-----------------------------------------------------------------|-----------------------------------------------------------|
| Name                                                            | APPLICATIONS/c32sat/<br>post-cbmc-zfcp-2.8-u2-noholes.cnf |
| MD5SUM                                                          | c4aa2ddc60eee766fbd95a32eed5b43                           |
| Bench Category                                                  | APPLICATION (applications instances)                      |
| Best result obtained on this benchmark                          | SAT                                                       |
| Best CPU time to get the best result obtained on this benchmark | 51.5012                                                   |
| Satisfiable                                                     |                                                           |
| (Un)Satisfiability was proved                                   |                                                           |
| Number of variables                                             | 10950109                                                  |
| Number of clauses                                               | 32697150                                                  |
| Sum of the clauses size                                         | 76320846                                                  |
| Maximum clause length                                           | 65                                                        |
| Minimum clause length                                           | 1                                                         |
| Number of clauses of size 1                                     | 2415                                                      |
| Number of clauses of size 2                                     | 21783823                                                  |
| Number of clauses of size 3                                     | 10907882                                                  |
| Number of clauses of size 4                                     | 1592                                                      |
| Number of clauses of size 5                                     | 131                                                       |
| Number of clauses of size over 5                                | 1307                                                      |

## Results of the different solvers on this benchmark

| Solver Name                              | TraceID | Answer | CPU time | Wall clock time |
|------------------------------------------|---------|--------|----------|-----------------|
| SApperIoT_base (complete)                | 1563400 | SAT    | 51.5012  | 204.435         |
| picosat 913 (complete)                   | 1563397 | SAT    | 118.794  | 120.066         |
| adaptg2wsat2009 2009-03-23 (incomplete)  | 1563429 | ?      | 0        | 0.00536097      |
| Hybrid2 2009-03-22 (incomplete)          | 1563418 | ?      | 0        | 0.00509205      |
| TNM 2009-03-22 (incomplete)              | 1563420 | ?      | 0        | 0.00590894      |
| IPAWS 2009-03-22 (incomplete)            | 1563407 | ?      | 0.000999 | 0.00650804      |
| adaptg2wsat2009+ 2009-03-23 (incomplete) | 1563430 | ?      | 0.000999 | 0.00578992      |
| NCVW 2009-03-22 (incomplete)             | 1563419 | ?      | 0.000999 | 0.00562698      |
| MXC 2009-03-10 (complete)                | 1563395 | ?      | 2.40463  | 2.42289         |

Ouvrir « http://www.cril.univ-artois.fr/SAT09/results/solver.php?dev=22&idbenchmark=70953 »

# Impressive results of the SAT 2009 competition

SAT 2009 Competition: main competition (phase2): solvers results per benchmarks

http://www.cril.univ-artois.fr/SAT09/results/bench.php?dev=29&idbench=71111

SAT4J - Recherche Google Graphviz Apple - Supp...00 Slide ... Nokia 6600 S... | Paul Bain Extending D... Conditions AndrewEscobar.com

Bug 291985 Submitted - [compile... SAT 2009 Competition: main com...

## General information on the benchmark

|                                                                 |                                            |
|-----------------------------------------------------------------|--------------------------------------------|
| Name                                                            | CRAFTED/sgen/unsat/sgen1-unsat-121-100.cnf |
| MD5SUM                                                          | 24865fea2f97ba63d115c240cb3cb69f           |
| Bench Category                                                  | CRAFTED (crafted instances)                |
| Best result obtained on this benchmark                          |                                            |
| Best CPU time to get the best result obtained on this benchmark |                                            |
| Satisfiable                                                     |                                            |
| (Un)Satisfiability was proved                                   |                                            |
| Number of variables                                             | 121                                        |
| Number of clauses                                               | 252                                        |
| Sum of the clauses size                                         | 756                                        |
| Maximum clause length                                           | 3                                          |
| Minimum clause length                                           | 3                                          |
| Number of clauses of size 1                                     | 0                                          |
| Number of clauses of size 2                                     | 0                                          |
| Number of clauses of size 3                                     | 252                                        |
| Number of clauses of size 4                                     | 0                                          |
| Number of clauses of size 5                                     | 0                                          |
| Number of clauses of size over 5                                | 0                                          |

## Results of the different solvers on this benchmark

| Solver Name                                         | TraceID | Answer        | CPU time | Wall clock time |
|-----------------------------------------------------|---------|---------------|----------|-----------------|
| SAT07 reference solver: SATzilla CRAFTED (complete) | 1785603 | ? (exit code) | 4998.65  | 5001.1          |
| SATzilla2009_C 2009-03-22 (complete)                | 1825787 | ? (exit code) | 4998.65  | 5000.32         |
| VARSAT-industrial 2009-03-22 (complete)             | 1785604 | ? (TO)        | 5000.04  | 5001.91         |
| glucose 1.0 (complete)                              | 1784160 | ? (TO)        | 5000.04  | 5002.51         |
| IUT_BMB_SAT 1.0 (complete)                          | 1785601 | ? (TO)        | 5000.05  | 5002.21         |
| MXC 2009-03-10 (complete)                           | 1784161 | ? (TO)        | 5000.06  | 5001.81         |
| SApperIoT base (complete)                           | 1785605 | ? (TO)        | 5000.06  | 5001.51         |
| MiniSat 2.1 (Sat-race'08 Edition) (complete)        | 1784159 | ? (TO)        | 5000.1   | 5002.21         |
| clasp 1.2.0-SAT09-32 (complete)                     | 1785600 | ? (TO)        | 5000.1   | 5013.71         |
| precosat 236 (complete)                             | 1784158 | ? (TO)        | 5000.1   | 5002.11         |

# The team

- Organizers
- ▶ Daniel Le Berre
  - ▶ Olivier Roussel
  - ▶ Laurent Simon (apart main track)
- Judges
- ▶ Andreas Goerdt
  - ▶ Ines Lynce
  - ▶ Aaron Stump

Computer infrastructure provided by CRIL (96 bi-processor cluster) and LRI (48 quad-core cluster + one 16 core machine (68GB) for the parallel track).

# The tracks

**Main track** sequential solvers

**competition** Source code of the solver should be available after the competition

**demonstration** Binary code should be available after the competition (for research purpose)

**Parallel** Solvers tailored to run on multicore computers (up to 16 cores)

**Minisat Hack** Submission of (small) patches against latest public release of Minisat2

**Preprocessing track** competition of preprocessors in front of Minisat2.

# Integration of the competition in the SAT'09 conference

Tuesday

- ▶ Efficiently Calculating Tree Measures Using SAT : [bio\\_2 benchmarks](#)
- ▶ Finding Efficient Circuits Using SAT solvers : [mod\\_circuits benchmarks](#)

Wednesday

- ▶ On the fly clause improvement : [Circus, main track](#)
- ▶ Problem sensitive restarts heuristics for the DPLL procedure : [Minisat09z, minisat hack](#)
- ▶ Improved Conflict-Clause Minimization Leads to Improved Propositional Proof Traces : [Minisat2Hack, minisat hack](#)
- ▶ A novel approach to combine SLS and a DPLL solver for the satisfiability problem : [hybridGM, main track](#)
- ▶ Building a Hybrid SAT solver via Conflict Driven, Look-Ahead and Xor reasoning techniques : [MoRsat, main track](#)
- ▶ Improving Variable Selection Process in Stochastic Local Search for Propositional Satisfiability : [slstc, main track](#)
- ▶ VARSAT : Integrating Novel Probabilistic Inference Techniques with DPLL Search : [VARSAT, main track](#)

Thursday

- ▶ Width-Based Restart Policies for Clause Learning : [Rsat, main track](#)

# Common rules to all tracks

- ▶ No more than 3 solvers **per submitter**
- ▶ Compared using a simple static ranking scheme
- ▶ Results available for SAT, UNSAT and SAT+UNSAT benchmarks.
- ▶ Results available to the submitters for checking : **It is the responsibility of the competitor to check that his system performed as expected !**

# New scoring scheme

- ▶ Purse based scoring since 2005 (designed by Allen van Gelder).
  - pros
    - ▶ Take into account various aspects of the solver (power, robustness, speed).
    - ▶ Focus on **singular** solvers
  - cons
    - ▶ Difficult to check (and understand)
    - ▶ Too much weight on singularity?
    - ▶ **Depends on the set of competitors**
- ▶ “Spec 2009” static scoring scheme desirable
  - ▶ To compare easily other solvers (e.g. reference solvers) without disturbing the ranking of the competitors.
  - ▶ To allow anybody to compare his solver to the SAT 2009 competitors on similar settings.

- NBTOTAL** Total number of benchmarks to solve
- NBSOLVED** Total number of benchmarks solved within a given timeout
- NBUNSOLVEDSERIES** Total number of set of benchmarks for which the solver was unable to solve any element.
- TIMEOUT** Time allowed to solve a given benchmark
- ti** Time needed to solve a given benchmark, within the time limit
- PENALTY** Constant to use as a penalty for benchmarks not solved within the timeout
- SERIESPENALTY** Constant to use as a penalty for a set of benchmarks in which all members cannot be solved by the solver.

# Spec 2009 proposals

- ▶ Lexicographical *NBSOLVED*,  $\sum t_i$
- ▶ Cumulative time based, with timeout penalty

$$\sum t_i + (NBTOTAL - NBSOLVED) * TIMEOUT * PENALTY$$

- ▶ Cumulative time based, with timeout penalty, log based

$$\sum \log_{10}(1+t_i) + (NBTOTAL - NBSOLVED) * \log_{10}((1 + TIMEOUT) * PENALTY)$$

- ▶ Cumulative time based, with timeout and robustness penalties (Proposed by Marijn Heule)

$$\sum t_i + (NBTOTAL - NBSOLVED) * TIMEOUT * PENALTY + NBUNSOLVEDSERIES * SERIESPENALTY$$

- ▶ SAT 2005 and 2007 purse based scoring

# Spec 2009 proposals and results of the votes

- ▶ Lexicographical *NBSOLVED*,  $\sum t_i$  9 votes
- ▶ Cumulative time based, with timeout penalty 3 votes

$$\sum t_i + (NBTOTAL - NBSOLVED) * TIMEOUT * PENALTY$$

- ▶ Cumulative time based, with timeout penalty, log based

$$\sum \log_{10}(1+t_i) + (NBTOTAL - NBSOLVED) * \log_{10}((1 + TIMEOUT) * PENALTY)$$

- ▶ Cumulative time based, with timeout and robustness penalties (Proposed by Marijn Heule) 4 votes

$$\sum t_i + (NBTOTAL - NBSOLVED) * TIMEOUT * PENALTY + NBUNSOLVEDSERIES * SERIESPENALTY$$

- ▶ SAT 2005 and 2007 purse based scoring

# Industrial vs Application

- ▶ Many instances in the industrial category do not come from industry
- ▶ Application better reflects the wide use of SAT technology

# Judges decisions regarding the selection of submitted vs existing benchmarks

- ▶ No more than 10% of the benchmarks should come from the same source.
- ▶ The final selection of benchmarks should contain 45% existing benchmarks and 55% submitted benchmarks.
- ▶ The final selection should contain 10% easy, 40% medium and 50% hard benchmarks.
- ▶ Duplicate benchmarks found after the selection was done will simply be removed from the selection. No other benchmarks will be added to the selection.

# Application benchmarks submitted to the competition

- Aprove** (*Carsten Fuhs*) Term Rewriting systems benchmarks.
- BioInfo I** (*Fabien Corblin*) Queries to find the maximal size of a biological behavior without cycles in discrete genetic networks.
- BioInfo II** (*Maria Louisa Bonet*) Evolutionary trees (presented on Tuesday).
- Bit Verif** (*Robert Brummayer*) Bit precise software verification generated by the SMT solver Boolector.
- C32SAT** Submitted by Hendrik Post and Carsten Sinz. Software verification generated by the C32SAT satisfiability checker for C programs.
- Crypto** (*Milan Sesum*) Encode attacks for both the DES and MD5 crypto systems.
- Diagnosis** (*Anbulagan and Alban Grastien*) 4 different encodings of discrete event systems.

# Application benchmarks : classification

| Origin       | EASY |       | MEDIUM |       | HARD |       |        | Total |
|--------------|------|-------|--------|-------|------|-------|--------|-------|
|              | SAT  | UNSAT | SAT    | UNSAT | SAT  | UNSAT | UNKOWN |       |
| SAT RACES    | 6    | 18    | 43     | 50    | 3    | 21    | -      | 141   |
| SAT COMP 07  | 6    | 15    | 47     | 49    | 7    | 12    | 45     | 181   |
| SUBMITTED 09 | 60   | 38    | 38     | 60    | 8    | 12    | 102    | 318   |
| Total        | 72   | 71    | 128    | 159   | 18   | 45    | 147    | 640   |

| Origin     | EASY |       | MEDIUM |       | HARD |       |        | Total |
|------------|------|-------|--------|-------|------|-------|--------|-------|
|            | SAT  | UNSAT | SAT    | UNSAT | SAT  | UNSAT | UNKOWN |       |
| Aprove     | 21   | -     | 4      | -     | -    | -     | -      | 25    |
| BioInfo I  | 3    | -     | 6      | 11    | -    | -     | -      | 20    |
| BioInfo II | 9    | -     | 4      | 3     | -    | -     | 24     | 40    |
| Bit Verif  | -    | 14    | -      | 22    | -    | 6     | 23     | 65    |
| C32SAT     | -    | 1     | 1      | 3     | -    | 3     | 2      | 10    |
| Crypto     | 5    | -     | 7      | 6     | 4    | -     | 40     | 62    |
| Diagnosis  | 22   | 23    | 16     | 15    | 4    | 3     | 13     | 96    |
| Total      | 60   | 38    | 38     | 60    | 8    | 12    | 102    | 318   |

# Crafted benchmarks submitted to the competition

**Edge Matching** Submitted by Marijn Heule. Four encodings of edge matching problems

**Mod Circuits** submitted by Grigory Yaroslavtsev. Presented on Tuesday.

**Parity Games** submitted by Oliver Friedmann. The generator encodes parity games of a fixed size  $n$  that forced the strategy improvement algorithm to require at least  $i$  iterations.

**Ramsey Cube** Submitted by Philipp Zumstein.

**RB SAT** Submitted by Nouredine Ould Mohamedou. Random CSP problems encoded into SAT.

**Sgen** submitted by Ivor Spence. Small but hard satisfiability benchmarks, either SAT or UNSAT.

**SGL** submitted by Calin Auton. Random SGL model -SRSGL. Sub Graph isomorphism problems.

# Difficulty of crafted benchmarks

| Origin        | EASY |       | MEDIUM |       | HARD |       |        | Total |
|---------------|------|-------|--------|-------|------|-------|--------|-------|
|               | SAT  | UNSAT | SAT    | UNSAT | SAT  | UNSAT | UNKOWN |       |
| Edge Matching | -    | -     | 20     | -     | 6    | -     | 6      | 32    |
| ModCircuits   | -    | 1     | 4      | 1     | -    | -     | 13     | 19    |
| Parity Games  | 6    | 8     | 7      | 2     | -    | -     | 1      | 24    |
| Ramsey Cube   | 1    | -     | 5      | 3     | -    | -     | 1      | 10    |
| RBSAT         | -    | -     | 34     | 1     | -    | -     | 325    | 360   |
| SGEN          | 5    | 1     | 4      | 2     | -    | -     | 9      | 21    |
| SGL           | 106  | -     | 1      | -     | -    | -     | -      | 107   |
| Total         | 118  | 10    | 75     | 9     | 6    | -     | 355    | 573   |

| Origin | EASY |       |     | MEDIUM |       |     | HARD |       |     |     | Total |
|--------|------|-------|-----|--------|-------|-----|------|-------|-----|-----|-------|
|        | SAT  | UNSAT | ALL | SAT    | UNSAT | ALL | SAT  | UNSAT | UNK | ALL |       |
| old    | -    | 4     | 4   | 19     | 42    | 61  | 4    | 12    | 58  | 74  | 139   |
| new    | 19   | 7     | 26  | 50     | 9     | 59  | 6    | -     | 70  | 76  | 161   |
| Total  | 19   | 11    | 30  | 69     | 65    | 120 | 11   | 10    | 129 | 150 | 300   |

Back to the first competition aim :

- ▶ a lot of new methods exist, but hard to tell which one is the best
- ▶ SatELite is widely used, but getting old
- ▶ We want to encourage new methods
- ▶ Allow to easily enhance all solvers by just adding preprocessors in front of them

# Preprocessor track : competitors

| Solver name             | Authors                                                    |
|-------------------------|------------------------------------------------------------|
| Competition division    |                                                            |
| IUT_BMB_SIM 1.0         | Abdorrahim Bahrami, Seyed Rasoul Mousavi, Kiarash Bazargan |
| ReVivAI 0.23            | Cédric Piette                                              |
| ReVivAI 0.23 + SatElite | Cédric Piette                                              |
| SatElite + ReVivAI 0.23 | Cédric Piette                                              |
| Demonstration division  |                                                            |
| kw_pre                  | Johan Alfredsson                                           |
| Reference solvers       |                                                            |
| minisat2-core           | Niklas Een and Niklas Sorensson                            |
| minisat2-simp           | Niklas Een and Niklas Sorensson                            |

# Preprocessing track : experimental settings

**Benchmarks** the one from the main track in both application and crafted categories.

**SAT engine** Minisat2 070721 core solver (without preprocessing).

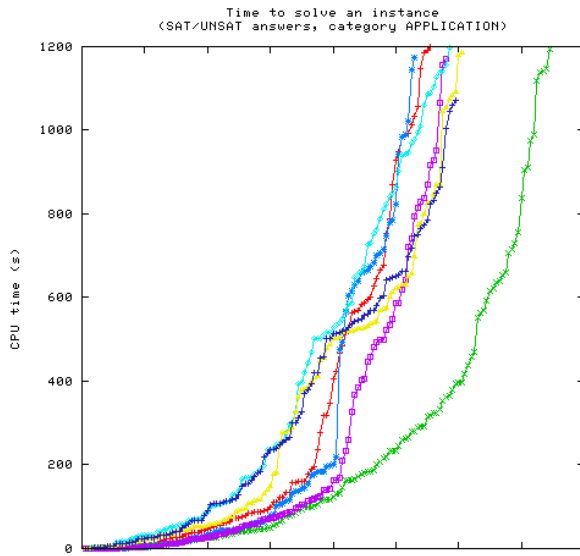
**Comparison criteria** the preprocessor and the engine seen as a black box.

**Timeout** 1200s.

# Preprocessing track : the results in application category

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 163   | 67  | 96    | 33886.97 |
| 1    | kw_pre                    | 149   | 58  | 91    | 34591.65 |
| 2    | ReVivAI 0.23 + SatElite   | 121   | 51  | 70    | 39093.24 |
| 3    | SatElite + ReVivAI 0.23   | 119   | 48  | 71    | 38374.13 |
| 4    | ReVivAI 0.23              | 117   | 53  | 64    | 44067.36 |
| 5    | minisat2-simp             | 116   | 46  | 70    | 25111.90 |
| 6    | IUT_BMB_SIM 1.0           | 111   | 46  | 65    | 30273.14 |
| 7    | minisat2-core             | 106   | 47  | 59    | 23477.71 |

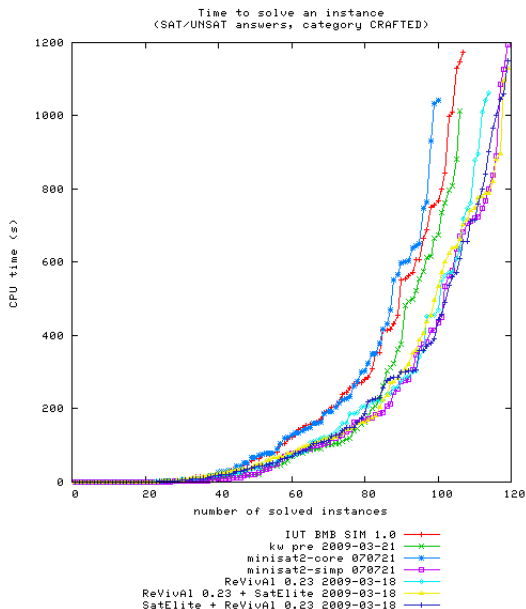
# Preprocessing track running time : application (SAT+UNSAT)



# Preprocessing track : the results in crafted category

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 137   | 92  | 45    | 20732.67 |
| 1    | minisat2-simp             | 119   | 76  | 43    | 23212.54 |
| 2    | SatElite + ReVivAI 0.23   | 119   | 75  | 44    | 24059.71 |
| 3    | ReVivAI 0.23 + SatElite   | 119   | 75  | 44    | 24622.54 |
| 4    | ReVivAI 0.23              | 114   | 72  | 42    | 20435.40 |
| 5    | IUT_BMB_SIM 1.0           | 107   | 74  | 33    | 23163.33 |
| 6    | kw_pre                    | 106   | 72  | 34    | 16298.74 |
| 7    | minisat2-core             | 100   | 69  | 31    | 17639.05 |

# Preprocessing track running time : crafted (SAT+UNSAT)



- ▶ Observe the effect of clearly identified “small changes” in a widely used solver
- ▶ Help understand what is really important in Minisat, what can be improved, ...
- ▶ Ensure that all solvers are comparable (small syntactic changes)
- ▶ Encourage easy entries to the competition (e.g. Master or first year PhD student)

# Minisat Hack competitors

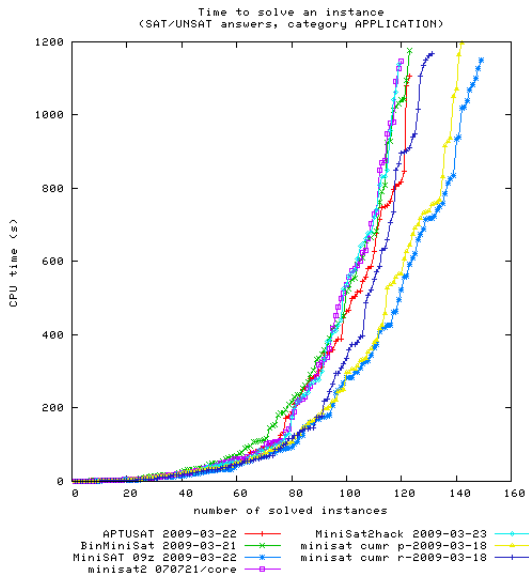
| Solver name       | Authors                                                                  |
|-------------------|--------------------------------------------------------------------------|
| Submissions       |                                                                          |
| APTUSAT           | Alexander Mishunin and Grigory Yaroslavtsev                              |
| BinMiniSat        | Kiyonori Taniguchi, Miyuki Koshimura, Hiroshi Fujita, and Ryuzo Hasegawa |
| MiniSAT 09z       | Markus Iser                                                              |
| MiniSat2hack      | Allen Van Gelder                                                         |
| minisat_cumr p/r  | Kazuya Masuda and Tomio Kamada                                           |
| Reference solvers |                                                                          |
| minisat2 core     | Niklas Een and Niklas Sorensson                                          |

Solvers presented during the SAT 2009 conference

# Minisat hack results

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 169   | 71  | 98    | 40959.35 |
| 1    | MiniSAT 09z               | 149   | 59  | 90    | 37228.91 |
| 2    | minisat_cumr p            | 142   | 58  | 84    | 32636.31 |
| 3    | minisat_cumr r            | 131   | 60  | 71    | 29316.97 |
| 4    | APTUSAT                   | 123   | 54  | 69    | 25418.27 |
| 5    | BinMiniSat                | 123   | 48  | 75    | 29326.67 |
| 6    | minisat2 core             | 120   | 53  | 67    | 25600.16 |
| 7    | MiniSat2hack              | 119   | 52  | 67    | 24024.97 |

# Minisat hack running time : application (SAT+UNSAT)



# The main track : competitors

| Solver name                       | Authors                                                                                                    |
|-----------------------------------|------------------------------------------------------------------------------------------------------------|
| adaptg2wsat2009/++                | chuMin Li, Wanxia Wei                                                                                      |
| CircUs                            | Hyojung Han                                                                                                |
| clasp 1.2.0-SAT09-32              | Benjamin Kaufmann                                                                                          |
| CSat 2009-03-22                   | Guanfeng Lv, Qian Wang, Kaile Su                                                                           |
| glucose 1.0                       | Gilles Audemard and Laurent Simon                                                                          |
| gnovelty+/2/2-H                   | Duc-Nghia Pham and Charles Gretton                                                                         |
| Hybrid2                           | Wanxia Wei, Chu Min Li, and Harry Zhang                                                                    |
| hybridGM 1/3/7                    | Adrian Balint                                                                                              |
| HydraSAT base/flat/multi          | Christoph Baldow, Friedrich Gräter, Steffen Hölldobler, Norbert Manthey, Max Seelemann, Peter Steinke, ... |
| iPAWS                             | John Thornton and Duc Nghia Pham                                                                           |
| IUT_BMB-SAT 1.0                   | Abdorrahim Bahrami, Seyed Rasoul Mousavi, Kiarash Bazargan                                                 |
| LySAT c/i                         | Youssef Hamadi, Saïd Jabbour, Lakhdar Saïs                                                                 |
| march_hi/nn                       | Marijn Heule                                                                                               |
| MoRsats                           | Jingchao Chen                                                                                              |
| MXC                               | David Bregman                                                                                              |
| NCVWr                             | Wanxia Wei, Chu Min Li, and Harry Zhang                                                                    |
| picosat 913                       | Armin Biere                                                                                                |
| precosat 236                      | Armin Biere                                                                                                |
| Rsat                              | Knot Pipatsrisawat and Adnan Darwiche                                                                      |
| SApperIoT base/hrp                | Stephan Kottler                                                                                            |
| SAT4J CORE 2.1 RC1                | Daniel Le Berre                                                                                            |
| SATzilla2009_C/I/R                | Lin Xu, Frank Hutter, Holger H. Hoos and Kevin Leyton-Brown                                                |
| slstc 1.0                         | Anton Belov, Zbigniew Stachniak                                                                            |
| TNM                               | Wanxia Wei and Chu Min Li                                                                                  |
| tts-5-0                           | Ivor Spence                                                                                                |
| VARSAT-crafted/random/industrial  | Eric Hsu                                                                                                   |
| kw 2009-03-20                     | Johan Alfredsson                                                                                           |
| MiniSat 2.1 (Sat-race'08 Edition) | Niklas Sorensson, Niklas Een                                                                               |

# The main track : reference solvers from 2007

| Solver name      | Authors                                                     |
|------------------|-------------------------------------------------------------|
| Random           |                                                             |
| adaptg2wsat+     | Wanxia Wei, Chu-Min Li and Harry Zhang                      |
| gnovelty+        | Duc Nghia Pham and Charles Gretton                          |
| March KS         | Marijn Heule and Hans van Maaren                            |
| SATzilla RANDOM  | Lin Xu, Frank Hutter, Holger H. Hoos and Kevin Leyton-Brown |
| Application      |                                                             |
| picosat 535      | Armin Biere                                                 |
| Rsat 07          | Knot Pipatsrisawat and Adnan Darwiche                       |
| Crafted          |                                                             |
| SATzilla CRAFTED | Lin Xu, Frank Hutter, Holger H. Hoos and Kevin Leyton-Brown |
| minisat SAT 2007 | Niklas Sorensson and Niklas Een                             |

# Main track : phase 1, application

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 196   | 79  | 117   | 33863.84 |
| 1    | precosat 236              | 164   | 65  | 99    | 37379.67 |
| 2    | MiniSat 2.1               | 155   | 65  | 90    | 27011.56 |
| 3    | LySAT i                   | 153   | 57  | 96    | 35271.11 |
| 4    | glucose 1.0               | 152   | 54  | 98    | 34784.84 |
| 5    | MiniSAT 09z               | 152   | 59  | 93    | 37872.87 |
| 6    | kw                        | 150   | 58  | 92    | 35080.23 |
| 7    | ManySAT 1.1 aimd 1        | 149   | 54  | 95    | 34834.19 |
| 8    | ManySAT 1.1 aimd 0        | 149   | 54  | 95    | 38639.59 |
| 9    | MXC                       | 147   | 62  | 85    | 27968.90 |
| 10   | ManySAT 1.1 aimd 2        | 145   | 51  | 94    | 34242.50 |
| 11   | CircUs                    | 144   | 59  | 85    | 36680.28 |
| 12   | Rsat                      | 143   | 53  | 90    | 31000.89 |
| 13   | SATzilla2009.L            | 142   | 60  | 82    | 33608.36 |
| 14   | minisat_cumr p            | 141   | 58  | 83    | 29304.08 |
| 15   | picosat 913               | 139   | 63  | 76    | 34013.47 |
| 16   | clasp 1.2.0-SAT09-32      | 138   | 53  | 85    | 33317.37 |
| 17   | Rsat 2007                 | 133   | 56  | 77    | 28975.23 |
| 18   | SApperloT base            | 129   | 55  | 74    | 31762.78 |
| 19   | picosat 535               | 126   | 59  | 67    | 33871.13 |

# Main track : phase 1, application continued

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 196   | 79  | 117   | 33863.84 |
| 20   | LySAT c                   | 123   | 51  | 72    | 26865.49 |
| 21   | IUT_BMB_SAT 1.0           | 116   | 46  | 70    | 20974.40 |
| 22   | HydraSAT Base             | 116   | 53  | 63    | 26856.33 |
| 23   | HydraSAT-Flat Flat        | 115   | 51  | 64    | 26016.34 |
| 24   | VARSAT-industrial         | 110   | 49  | 61    | 22753.77 |
| 25   | SApperIoT hrp             | 107   | 42  | 65    | 20954.19 |
| 26   | HydraSAT-Multi            | 106   | 49  | 57    | 16308.48 |
| 27   | SATzilla2009_C            | 106   | 45  | 61    | 25974.72 |
| 28   | VARSAT-crafted            | 99    | 44  | 55    | 23553.01 |
| 29   | SAT4J CORE 2.1 RC1        | 95    | 46  | 49    | 25380.84 |
| 30   | satake                    | 92    | 40  | 52    | 18309.62 |
| 31   | CSat 2009-03-22           | 91    | 40  | 51    | 20461.14 |
| 32   | SATzilla2009_R            | 59    | 36  | 23    | 6260.03  |
| 33   | VARSAT-random             | 59    | 25  | 34    | 16836.65 |
| 34   | march_hi                  | 21    | 9   | 12    | 5170.80  |
| 35   | march_nn                  | 21    | 10  | 11    | 6189.51  |
| 36   | Hybrid2                   | 12    | 11  | 1     | 3851.84  |
| 37   | adaptg2wsat2009           | 11    | 8   | 3     | 1746.45  |
| 38   | adaptg2wsat2009++         | 11    | 8   | 3     | 1806.37  |

# Main track : phase 1, application continued two

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 196   | 79  | 117   | 33863.84 |
| 39   | slstc 1.0                 | 10    | 9   | 1     | 2093.29  |
| 40   | tts                       | 10    | 6   | 4     | 2539.03  |
| 41   | NCVWr                     | 10    | 9   | 1     | 2973.72  |
| 42   | iPAWS                     | 8     | 8   | 3     | 1400.34  |
| 43   | ttsth-5-0                 | 8     | 4   | 4     | 2937.42  |
| 44   | hybridGM7                 | 7     | 7   | -     | 468.76   |
| 45   | gnoelty+                  | 7     | 7   | -     | 1586.83  |
| 46   | gNoelty+-T                | 7     | 7   | -     | 1826.46  |
| 47   | TNM                       | 6     | 5   | 1     | 1157.83  |
| 48   | hybridGM 1                | 5     | 5   | -     | 731.62   |
| 49   | hybridGM3                 | 5     | 5   | -     | 1103.11  |
| 50   | gnoelty+2                 | 4     | 4   | -     | 91.85    |

# First stage : crafted

| Rank | Solver                            | Total | SAT | UNSAT | CPU Time |
|------|-----------------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS)         | 194   | 124 | 70    | 19204.67 |
| 1    | clasp 1.2.0-SAT09-32              | 131   | 78  | 53    | 22257.76 |
| 2    | SATzilla2009_I                    | 128   | 86  | 42    | 21700.11 |
| 3    | SATzilla2009_C                    | 125   | 73  | 52    | 16701.85 |
| 4    | MXC 2009-03-10                    | 124   | 80  | 44    | 22256.57 |
| 5    | precosat 236                      | 122   | 81  | 41    | 22844.50 |
| 6    | IUT_BMB_SAT 1.0                   | 120   | 76  | 44    | 22395.97 |
| 7    | minisat SAT 2007                  | 119   | 76  | 43    | 22930.58 |
| 8    | SATzilla CRAFTED                  | 114   | 82  | 32    | 18066.80 |
| 9    | MiniSat 2.1 (Sat-race'08 Edition) | 114   | 74  | 40    | 18107.02 |
| 10   | glucose 1.0                       | 114   | 75  | 39    | 20823.96 |
| 11   | VARSAT-industrial                 | 113   | 73  | 40    | 22306.77 |
| 12   | SApperloT base                    | 113   | 73  | 40    | 22826.65 |
| 13   | picosat 913                       | 112   | 80  | 32    | 17111.73 |
| 14   | LySAT c                           | 112   | 70  | 42    | 21080.61 |
| 15   | CircUs                            | 107   | 70  | 37    | 16148.01 |
| 16   | kw                                | 106   | 72  | 34    | 16460.37 |
| 17   | Rsat                              | 105   | 71  | 34    | 14010.73 |
| 18   | SATzilla2009_R                    | 104   | 78  | 26    | 14460.38 |
| 19   | ManySAT 1.1 aimd 1                | 103   | 72  | 31    | 14991.64 |

# First stage : crafted continued

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 194   | 124 | 70    | 19204.67 |
| 20   | HydraSAT-Multi            | 103   | 70  | 33    | 20825.53 |
| 21   | HydraSAT-Flat             | 102   | 70  | 32    | 17796.15 |
| 22   | SApperIoT hrp             | 102   | 69  | 33    | 20647.84 |
| 23   | minisat_cumr p            | 102   | 75  | 27    | 23176.38 |
| 24   | VARSAT-crafted            | 102   | 61  | 41    | 23304.40 |
| 25   | LySAT i                   | 100   | 69  | 31    | 14874.18 |
| 26   | ManySAT 1.1 aimd 2        | 99    | 70  | 29    | 14211.48 |
| 27   | ManySAT 1.1 aimd 0        | 99    | 71  | 28    | 15251.61 |
| 28   | HydraSAT base             | 99    | 66  | 33    | 16718.94 |
| 29   | MiniSAT 09z               | 99    | 72  | 27    | 17027.31 |
| 30   | VARSAT-random             | 84    | 47  | 37    | 14023.19 |
| 31   | satake                    | 75    | 55  | 20    | 16261.12 |
| 32   | iPAWS                     | 71    | 71  | -     | 7352.89  |
| 33   | SAT4J CORE 2.1 RC1        | 71    | 50  | 21    | 15136.95 |
| 34   | adaptg2wsat2009           | 70    | 68  | 2     | 9425.51  |
| 35   | adaptg2wsat2009++         | 66    | 64  | 2     | 5796.69  |
| 36   | Hybrid2                   | 66    | 66  | -     | 10425.56 |
| 37   | CSat                      | 65    | 50  | 15    | 10319.33 |

# First stage : crafted continued two

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 194   | 124 | 70    | 19204.67 |
| 38   | march_hi                  | 63    | 45  | 18    | 10622.02 |
| 39   | TNM                       | 62    | 62  | -     | 8181.19  |
| 40   | March KS                  | 61    | 42  | 19    | 9021.93  |
| 41   | march_nn                  | 58    | 43  | 15    | 6232.17  |
| 42   | gnovelty+                 | 54    | 54  | -     | 5853.95  |
| 43   | gNovelty+-T               | 53    | 53  | -     | 5073.82  |
| 44   | hybridGM                  | 51    | 51  | -     | 5298.30  |
| 45   | hybridGM3                 | 51    | 51  | -     | 6737.29  |
| 46   | NCVWr                     | 48    | 48  | -     | 12116.63 |
| 47   | tts 5-0                   | 46    | 25  | 21    | 2507.80  |
| 48   | ttsth-5-0                 | 46    | 24  | 22    | 4020.68  |
| 49   | gnovelty+2                | 46    | 44  | 2     | 4840.28  |
| 50   | hybridGM7                 | 38    | 38  | -     | 4385.10  |
| 51   | slstc 1.0                 | 33    | 33  | -     | 4228.67  |

# Random results

| Rank | Solver                    | Total | SAT | UNSAT | CPU Time |
|------|---------------------------|-------|-----|-------|----------|
|      | Virtual Best Solver (VBS) | 459   | 359 | 100   | 62339.75 |
| 1    | SATzilla2009_R            | 365   | 299 | 66    | 51997.72 |
| 2    | TNM                       | 317   | 317 | -     | 35346.17 |
| 3    | gnoelty+2                 | 305   | 305 | -     | 26616.48 |
| 4    | hybridGM3                 | 299   | 299 | -     | 23272.79 |
| 5    | hybridGM7                 | 298   | 298 | -     | 25567.23 |
| 6    | adaptg2wsat2009++         | 297   | 297 | -     | 26432.65 |
| 7    | hybridGM 1                | 294   | 294 | -     | 23732.78 |
| 8    | adaptg2wsat2009           | 294   | 294 | -     | 26658.47 |
| 9    | Hybrid2                   | 290   | 290 | -     | 30134.40 |
| 10   | gnoelty+                  | 281   | 281 | -     | 25523.72 |
| 11   | NCVWr                     | 278   | 278 | -     | 31132.10 |
| 12   | gnoelty+                  | 272   | 272 | -     | 21956.28 |
| 13   | SATzilla RANDOM           | 268   | 177 | 91    | 42919.16 |
| 14   | gNoelty+-T                | 266   | 266 | -     | 22823.37 |
| 15   | adaptg2wsat+              | 265   | 265 | -     | 22333.18 |
| 16   | iPAWS                     | 258   | 258 | -     | 19296.93 |
| 17   | march_hi                  | 247   | 147 | 100   | 65568.89 |
| 18   | march_nn                  | 243   | 145 | 98    | 66494.85 |
| 19   | March KS                  | 239   | 149 | 90    | 57869.03 |
| 20   | SATzilla2009_L            | 145   | 90  | 55    | 37645.86 |

# Random results : weak solvers

| Rank | Solver                            | Total | SAT | UNSAT | CPU Time |
|------|-----------------------------------|-------|-----|-------|----------|
| 21   | slstc 1.0                         | 118   | 118 | -     | 13250.77 |
| 22   | clasp 1.2.0-SAT09-32              | 84    | 66  | 18    | 32979.32 |
| 23   | VARSAT-random                     | 83    | 72  | 11    | 30273.41 |
| 24   | picosat 913                       | 79    | 57  | 22    | 29440.52 |
| 25   | SATzilla2009_C                    | 73    | 61  | 12    | 22395.73 |
| 26   | VARSAT-industrial                 | 71    | 61  | 10    | 27295.84 |
| 27   | VARSAT-crafted                    | 70    | 60  | 10    | 27367.38 |
| 28   | SApperloT base                    | 70    | 53  | 17    | 28249.79 |
| 29   | IUT_BMB_SAT 1.0                   | 63    | 50  | 13    | 25630.38 |
| 30   | MXC                               | 61    | 50  | 11    | 28069.37 |
| 31   | LySAT c                           | 60    | 48  | 12    | 24329.68 |
| 32   | MiniSat 2.1 (Sat-race'08 Edition) | 41    | 37  | 4     | 16957.09 |
| 33   | minisat_cumr p                    | 29    | 29  | -     | 14078.15 |
| 34   | precosat 236                      | 27    | 25  | 2     | 9522.84  |
| 35   | satake                            | 24    | 24  | -     | 11034.05 |
| 36   | SApperloT hrp                     | 17    | 13  | 4     | 7724.34  |
| 37   | glucose 1.0                       | 17    | 17  | -     | 7772.56  |

# Random problems : very bad solvers

| Rank | Solver             | Total | SAT | UNSAT | CPU Time |
|------|--------------------|-------|-----|-------|----------|
| 38   | HydraSAT-Flat      | 16    | 15  | 1     | 5738.82  |
| 39   | HydraSAT-Multi     | 16    | 16  | -     | 7836.19  |
| 40   | HydraSAT Base      | 13    | 13  | -     | 4930.65  |
| 41   | CircUs             | 8     | 8   | -     | 2553.24  |
| 42   | ManySAT 1.1 aimd 0 | 7     | 7   | -     | 1783.47  |
| 43   | ManySAT 1.1 aimd 2 | 6     | 6   | -     | 957.09   |
| 44   | LySAT i            | 6     | 6   | -     | 2124.49  |
| 45   | CSat               | 6     | 6   | -     | 2263.92  |
| 46   | Rsat               | 5     | 5   | -     | 1801.20  |
| 47   | ManySAT 1.1 aimd 1 | 5     | 5   | -     | 4144.36  |
| 48   | kw                 | 4     | 4   | -     | 635.52   |
| 49   | SAT4J CORE 2.1 RC1 | 4     | 4   | -     | 1440.19  |
| 50   | MiniSAT 09z        | 3     | 3   | -     | 1096.04  |
| 51   | tts 5.0            | 0     | 0   | 0     | 0.00     |
| 52   | ttsth-5-0          | 0     | 0   | 0     | 0.00     |

# A few remarks ...

- ▶ CDCL solvers are the best performers both in the application and crafted category !
- ▶ Incomplete solvers only good for Random SAT category
- ▶ SatZilla approach really impressive !

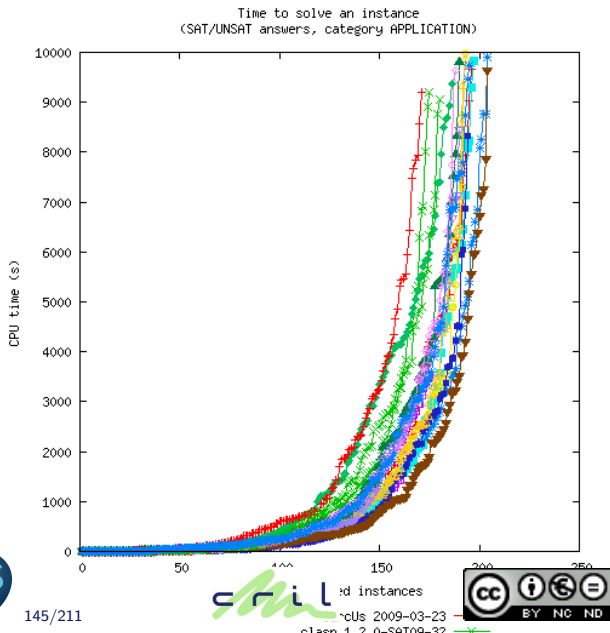
# What about the second stage ?

- ▶ Keep only the best solvers for each category
- ▶ Increase the timeout : 10000s for application, 5000s for crafted
- ▶ See which solvers are the winners this year ...

# Final results, Application, SAT+UNSAT

| Rank | Solver                            | Total | SAT | UNSAT | CPU Time  |
|------|-----------------------------------|-------|-----|-------|-----------|
|      | Virtual Best Solver (VBS)         | 229   | 91  | 138   | 153127.06 |
| 1    | precosat 236                      | 204   | 79  | 125   | 180345.80 |
| 2    | glucose 1.0                       | 204   | 77  | 127   | 218826.10 |
| 3    | LySAT i                           | 197   | 73  | 124   | 198491.53 |
| 4    | CircUs                            | 196   | 77  | 119   | 229285.44 |
| 5    | SATzilla2009_I 2009-03-22         | 195   | 81  | 114   | 234743.41 |
| 6    | MiniSat 2.1 (Sat-race'08 Edition) | 194   | 78  | 116   | 144548.45 |
| 7    | ManySAT 1.1 aimd 1                | 193   | 71  | 122   | 173344.71 |
| 8    | MiniSAT 09z                       | 193   | 78  | 115   | 184696.75 |
| 9    | MXC                               | 190   | 79  | 111   | 180409.82 |
| 10   | minisat_cumr p                    | 190   | 75  | 115   | 206371.06 |
| 11   | Rsat                              | 188   | 74  | 114   | 187726.95 |
| 12   | SApperloT base                    | 186   | 78  | 108   | 282488.39 |
| 13   | Rsat 2007-02-08                   | 180   | 69  | 111   | 195748.38 |
| 14   | kw                                | 175   | 67  | 108   | 90213.34  |
| 15   | clasp 1.2.0-SAT09-32              | 175   | 60  | 115   | 163460.74 |
| 16   | picosat 535                       | 171   | 76  | 95    | 209004.97 |

# Cactus plot : application SAT+UNSAT



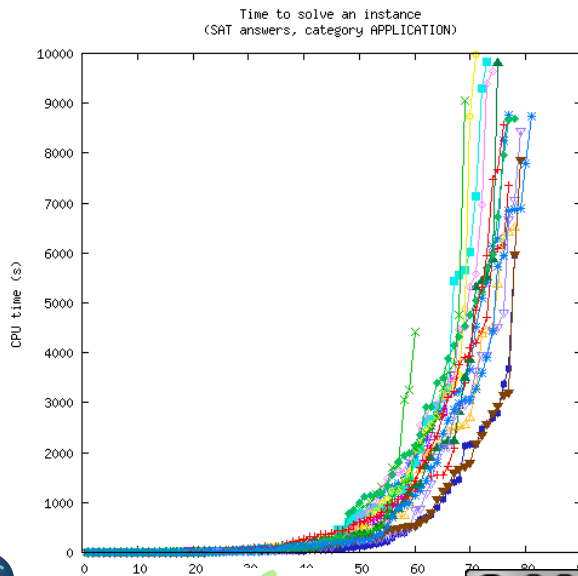
# Final results, Application, SAT only

| Rank | Solver                            | SAT | CPU Time  |
|------|-----------------------------------|-----|-----------|
|      | Virtual Best Solver (VBS)         | 91  | 52336.24  |
| 1    | SATzilla2009_I                    | 81  | 96609.87  |
| 2    | precosat 236                      | 79  | 52903.18  |
| 3    | MXC                               | 79  | 75203.55  |
| 4    | MiniSat 2.1 (Sat-race'08 Edition) | 78  | 42218.37  |
| 5    | MiniSAT 09z                       | 78  | 75075.48  |
| 6    | SApperIoT base                    | 78  | 111286.45 |
| 7    | CircUs                            | 77  | 74720.59  |
| 8    | glucose 1.0                       | 77  | 90532.72  |
| 9    | picosat 535                       | 76  | 84382.33  |
| 10   | minisat_cumr p                    | 75  | 67373.20  |
| 11   | Rsat 2009-03-22                   | 74  | 85363.26  |
| 12   | LySAT i                           | 73  | 81793.98  |
| 13   | ManySAT 1.1 aimd 1                | 71  | 62994.30  |
| 14   | Rsat 2007-02-08                   | 69  | 47294.67  |
| 15   | kw                                | 67  | 31254.87  |
| 16   | clasp 1.2.0-SAT09-32              | 60  | 25529.94  |

# Final results, Application, SAT only

| Rank | Solver                            | SAT | CPU Time  |
|------|-----------------------------------|-----|-----------|
|      | Virtual Best Solver (VBS)         | 91  | 52336.24  |
| 1    | SATzilla2009_I                    | 81  | 96609.87  |
| 2    | precosat 236                      | 79  | 52903.18  |
| 3    | MXC                               | 79  | 75203.55  |
| 4    | MiniSat 2.1 (Sat-race'08 Edition) | 78  | 42218.37  |
| 5    | MiniSAT 09z                       | 78  | 75075.48  |
| 6    | SApperloT base                    | 78  | 111286.45 |
| 7    | CircUs                            | 77  | 74720.59  |
| 8    | glucose 1.0                       | 77  | 90532.72  |
| 9    | picosat 535                       | 76  | 84382.33  |
| 10   | minisat_cumr p                    | 75  | 67373.20  |
| 11   | Rsat 2009-03-22                   | 74  | 85363.26  |
| 12   | LySAT i                           | 73  | 81793.98  |
| 13   | ManySAT 1.1 aimd 1                | 71  | 62994.30  |
| 14   | Rsat 2007-02-08                   | 69  | 47294.67  |
| 15   | kw                                | 67  | 31254.87  |
| 16   | clasp 1.2.0-SAT09-32              | 60  | 25529.94  |

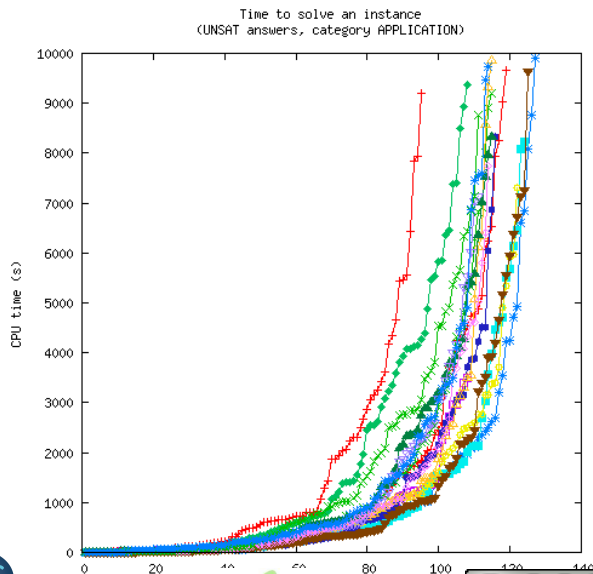
# Cactus plot : application SAT (timeout matters !)



# Final results, Application, UNSAT only

| Rank | Solver                            | UNSAT | CPU Time  |
|------|-----------------------------------|-------|-----------|
|      | Virtual Best Solver (VBS)         | 138   | 100790.82 |
| 1    | glucose 1.0                       | 127   | 128293.39 |
| 2    | precosat 236                      | 125   | 127442.62 |
| 3    | LySAT i                           | 124   | 116697.55 |
| 4    | ManySAT 1.1 aimd 1                | 122   | 110350.41 |
| 5    | CircUs                            | 119   | 154564.85 |
| 6    | MiniSat 2.1 (Sat-race'08 Edition) | 116   | 102330.08 |
| 7    | MiniSAT 09z                       | 115   | 109621.27 |
| 8    | clasp 1.2.0-SAT09-32              | 115   | 137930.80 |
| 9    | minisat_cumr p                    | 115   | 138997.86 |
| 10   | Rsat                              | 114   | 102363.69 |
| 11   | SATzilla2009_I                    | 114   | 138133.54 |
| 12   | MXC                               | 111   | 105206.27 |
| 13   | Rsat 2007-02-08                   | 111   | 148453.71 |
| 14   | kw 2009-03-20                     | 108   | 58958.47  |
| 15   | SApperloT base                    | 108   | 171201.93 |
| 16   | picosat 535                       | 95    | 124622.64 |

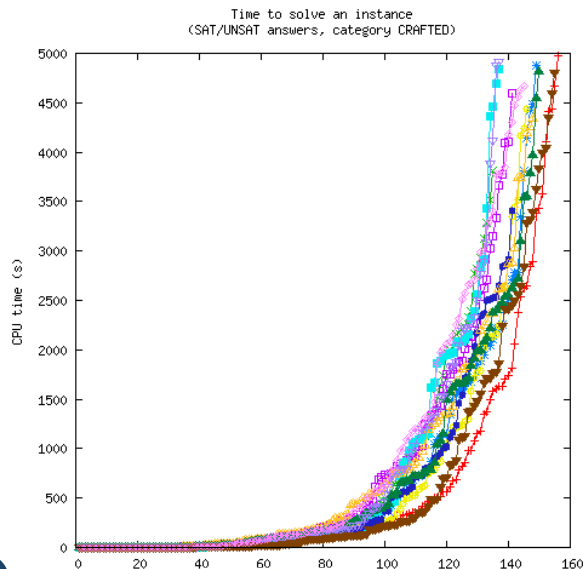
# Cactus plot : application UNSAT (timeout matters !)



# Final results, Crafted, SAT+UNSAT

| Rank | Solver                            | Total | SAT | UNSAT | CPU Time  |
|------|-----------------------------------|-------|-----|-------|-----------|
|      | Virtual Best Solver (VBS)         | 187   | 108 | 79    | 62264.60  |
| 1    | clasp 1.2.0-SAT09-32              | 156   | 92  | 64    | 89194.49  |
| 2    | SATzilla2009_C                    | 155   | 83  | 72    | 94762.27  |
| 3    | minisat SAT 2007                  | 150   | 90  | 60    | 99960.89  |
| 4    | IUT_BMB_SAT 1.0                   | 149   | 89  | 60    | 93502.16  |
| 5    | SApperloT base                    | 149   | 92  | 57    | 108298.52 |
| 6    | MXC                               | 146   | 91  | 55    | 76965.59  |
| 7    | VARSAT-industrial                 | 145   | 85  | 60    | 119365.13 |
| 8    | precosat 236                      | 141   | 90  | 51    | 66318.44  |
| 9    | LySAT c                           | 141   | 83  | 58    | 89925.84  |
| 10   | SATzilla CRAFTED                  | 137   | 84  | 53    | 76856.90  |
| 11   | MiniSat 2.1 (Sat-race'08 Edition) | 137   | 87  | 50    | 78381.80  |
| 12   | glucose 1.0                       | 135   | 86  | 49    | 70385.63  |

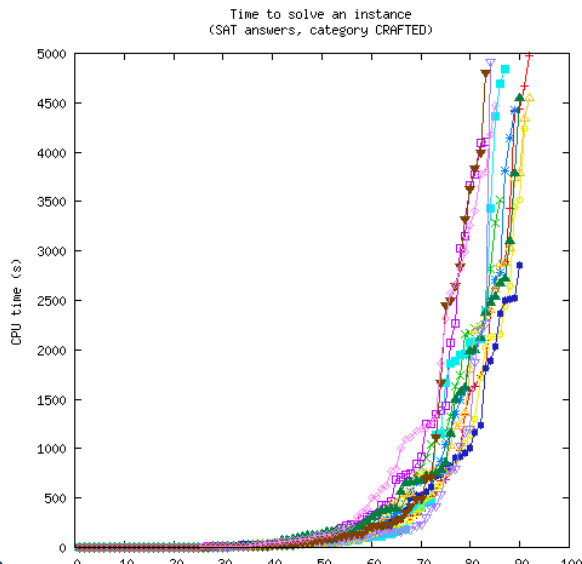
# Cactus plot : crafted SAT+UNSAT



# Final results : crafted SAT only

| Rank | Solver                            | SAT | CPU Time |
|------|-----------------------------------|-----|----------|
|      | Virtual Best Solver (VBS)         | 108 | 21224.84 |
| 1    | clasp 1.2.0-SAT09-32              | 92  | 49775.04 |
| 2    | SApperloT base                    | 92  | 54682.14 |
| 3    | MXC 2009-03-10                    | 91  | 39227.16 |
| 4    | precosat 236                      | 90  | 34447.16 |
| 5    | minisat SAT 2007                  | 90  | 48346.20 |
| 6    | IUT_BMB_SAT 1.0                   | 89  | 45287.01 |
| 7    | MiniSat 2.1 (Sat-race'08 Edition) | 87  | 41994.77 |
| 8    | glucose 1.0                       | 86  | 37779.61 |
| 9    | VARSAT-industrial                 | 85  | 54521.77 |
| 10   | SATzilla CRAFTED                  | 84  | 21726.48 |
| 11   | SATzilla2009_C                    | 83  | 39383.44 |
| 12   | LySAT c                           | 83  | 42073.80 |

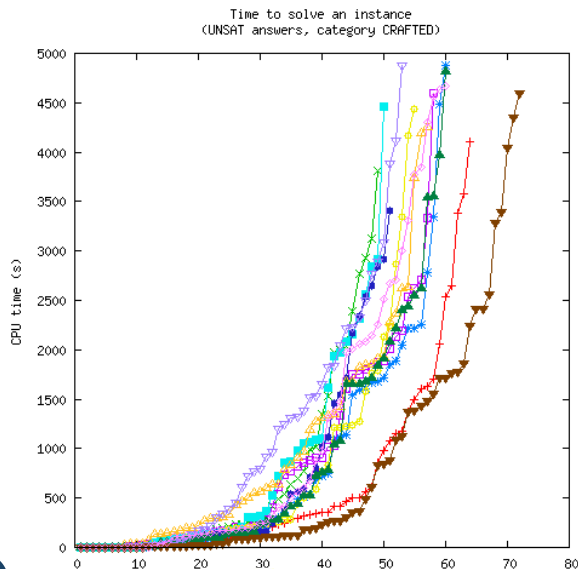
# Cactus plot : crafted SAT only



# Final results : crafted UNSAT only

| Rank | Solver                            | UNSAT | CPU Time |
|------|-----------------------------------|-------|----------|
|      | Virtual Best Solver (VBS)         | 79    | 41039.76 |
| 1    | SATzilla2009_C                    | 72    | 55378.83 |
| 2    | clasp 1.2.0-SAT09-32              | 64    | 39419.45 |
| 3    | IUT_BMB_SAT 1.0                   | 60    | 48215.14 |
| 4    | minisat SAT 2007                  | 60    | 51614.69 |
| 5    | VARSAT-industrial                 | 60    | 64843.36 |
| 6    | LySAT c                           | 58    | 47852.03 |
| 7    | SApperloT base                    | 57    | 53616.38 |
| 8    | MXC                               | 55    | 37738.43 |
| 9    | SATzilla CRAFTED                  | 53    | 55130.42 |
| 10   | precosat 236                      | 51    | 31871.28 |
| 11   | MiniSat 2.1 (Sat-race'08 Edition) | 50    | 36387.03 |
| 12   | glucose 1.0                       | 49    | 32606.02 |

# Cactus plot : crafted UNSAT only



# Agenda

Anatomy of a modern CDCL SAT solver (continued)

Certifiable Unsat answers

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT

# Extending CDCL : Underlying assumption

- *Techniques suitable/efficient for SAT are also suitable for extensions to SAT.*

# Extending CDCL : Underlying assumption

- ▶ *Techniques suitable/efficient for SAT are also suitable for extensions to SAT.*
- ▶ Problem : is it always true ?

# Extending CDCL : Underlying assumption

- ▶ *Techniques suitable/efficient for SAT are also suitable for extensions to SAT.*
- ▶ Problem : is it always true ?
- ▶ Answer : Not sure. We are very lucky when working with plain clauses and existential quantifiers.
- ▶ Beware of the benchmarks used to evaluate the solvers.

# Extending CDCL : Underlying assumption

- ▶ *Techniques suitable/efficient for SAT are also suitable for extensions to SAT.*
- ▶ Problem : is it always true ?
- ▶ Answer : Not sure. We are very lucky when working with plain clauses and existential quantifiers.
- ▶ Beware of the benchmarks used to evaluate the solvers.
- ▶ This talk :
  - ▶ How to extend a CDCL solver to manage linear pseudo boolean constraints.
  - ▶ Implementation in SAT4J
  - ▶ Application to MAXSAT

# Pure SAT is not sufficient to solve many problems

Things are getting harder

- ▶ SAT is a **decision problem** : answer is yes/no.
- ▶ For **very constrained** problems, a **proof of satisfiability** is usually a sufficient answer for users : SAT solvers provide them.
  - ▶ Bounded Model Checking : the proof is a bug
  - ▶ Planning : the proof is a plan
- ▶ For **underconstrained** problems, there are many solutions. Users look for the **best solution** : **optimization problem**.

## Linear Pseudo-Boolean constraint

$$-3x_1 + 4x_2 - 7x_3 + x_4 \leq -5$$

- ▶ variables  $x_i$  take their value in  $\{0, 1\}$
- ▶ coefficients and degree are integer-valued constants

## Pseudo-Boolean decision problem : NP-complete

$$\begin{cases} (a_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (a_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (b) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

- ▶ conjunction of PB-constraints
- ▶ often got an objective function :

$$\min : 4x_2 + x_5$$

- ▶ Optimization problem

# Interest of Linear Pseudo Boolean constraints

## Concise boolean function representation

- ▶ clauses generalization :

$$x_1 \vee x_2 \vee x_3$$

translates into

$$x_1 + x_2 + x_3 \geq 1$$

- ▶ cardinality constraints generalization :

$$\textit{atleast}(2, \{x_1, x_2, x_3\})$$

translates into

$$x_1 + x_2 + x_3 \geq 2$$

- ▶ expressivity : the translation into clauses of a PB-constraint can have an exponential size wrt the PB-constraint (without additional variables).

# Linear Pseudo Boolean constraints normalization

Assumptions used when designing a solver

- ▶ usual form :  $\geq$  comparison operator and positive constants
- ▶ reminder :  $\bar{x} = 1 - x$

$$-3x_1 + 4x_2 - 7x_3 + x_4 \leq -5$$

$$\equiv 3x_1 - 4x_2 + 7x_3 - x_4 \geq 5$$

$$\equiv 3x_1 + (4 - 4x_2) + 7x_3 + (1 - x_4) \geq 5 + 4 + 1$$

$$\equiv 3x_1 + 4\bar{x}_2 + 7x_3 + \bar{x}_4 \geq 10$$

# Rules on LPB constraints : Linear combination

$$\text{linear combination: } \frac{\begin{array}{l} \sum_i a_i \cdot x_i \geq k \\ \sum_i a'_i \cdot x_i \geq k' \end{array}}{\sum_i (\alpha \cdot a_i + \alpha' \cdot a'_i) \cdot x_i \geq \alpha \cdot k + \alpha' \cdot k'}$$

with  $\alpha > 0$  and  $\alpha' > 0$

$$\frac{\begin{array}{l} x_1 + x_2 + 3x_3 + x_4 \geq 3 \quad 2\overline{x_1} + 2\overline{x_2} + x_4 \geq 3 \\ 2x_1 + 2x_2 + 6x_3 + 2x_4 + 2\overline{x_1} + 2\overline{x_2} + x_4 \geq 2 \times 3 + 3 \\ 2x_1 + 2x_2 + 6x_3 + 2x_4 + 2 - 2x_1 + 2 - 2x_2 + x_4 \geq 9 \\ 6x_3 + 3x_4 \geq 5 \end{array}}{}$$

Note that  $2x + 2\overline{x} = 2$ , not 0 !

Note that the coefficients are growing !

# Some other inference rules

division One can always reduce a LPB constraint to a clause !

$$\frac{5x_3 + 3x_4 \geq 5}{\begin{array}{l} \lceil 5/5 \rceil x_3 + \lceil 3/5 \rceil x_4 \geq \lceil 5/5 \rceil \\ x_3 + x_4 \geq 1 \end{array}}$$

weakening

$$\frac{5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8}{3x_2 + 2x_3 + 2x_4 + x_5 \geq 3}$$

saturation

$$\frac{6x_3 + 3x_4 \geq 5}{5x_3 + 3x_4 \geq 5}$$

# Resolution extends Cutting Planes

- ▶ Linear combination + division = cutting plane proof system (complete).
- ▶ First introduced for linear programming by R. Gomory in 1958
- ▶ Cutting planes can be seen as a generalization of the resolution (J.N. Hooker, 1988)
- ▶ Replace Resolution during Conflict Analysis by Cutting Planes

# Original aim of the work

- ▶ Solve Pseudo-Boolean decision and optimization problems !
- ▶ By extending Conflict Driven Clause Learning solvers with a stronger reasoning engine
- ▶ While keeping as much features found in nowadays SAT solvers : **based on a generic and flexible SAT solver platform, SAT4J**
- ▶ Compare this approach with other PB-solvers, especially the ones based on completely different approaches

# Generalizing Clauses to Constraints

- 1994 Barth[1] DPLL extension to pseudo boolean constraints
- 1997 Walser [10] and Prestwich [6, 7] local search LPB solvers
- 2001 Aloul et al [8] CDC<sub>L</sub> handling LPB constraints (clause learning) : Satire, Satzoo, Minisat, PBS
- 2002 Dixon and Ginsberg[3] extended RelSAT to LPB (LPB learning)
- 2003 Chai and Kuehlmann [2] CDC<sub>L</sub> with LPB learning
- 2004 Dixon et al [5] describe a generic CDC<sub>o</sub>L solver based on group theory
- 2004 Thiffault et al [9] CDC<sub>L</sub> solver handling arbitrary boolean gates (clause learning).

## CDC<sub>o</sub>L : Conflict Driven Constraint Learning

- ▶ Look Ahead : propagate forced truth values and detect violated constraints.
- ▶ Look Back :
  - ▶ Inference rule between a conflict and a reason
  - ▶ *assertive constraints*
  - ▶ Computing the backtrack level
- ▶ Data Structure : lazy one such as Watched Literals?
- ▶ Validation : being able to solve e.g. pigeon hole problems in polynomial time.

# Requirements for a CDC<sub>o</sub>L solver

- ▶ Look Ahead : propagate forced truth values and detect violated constraints.
- ▶ Look Back :
  - ▶ Inference rule between a conflict and a reason
  - ▶ *assertive constraints*
  - ▶ Computing the backtrack level
- ▶ Data Structure : lazy one such as Watched Literals?

# Forced truth values : Implicative and Assertive constraints

- ▶ *unit clause* : a clause that propagates one truth value to be satisfiable.
- ▶ *implicative constraint* : a constraint that propagates *at least* one truth value to be satisfiable.
- ▶ a constraint  $C$  is *implicative* iff  $\exists a_i x_i \in C$  such that  $\sum_{j \neq i} a_j < k$  or  $\sum a_j - k < a_i$ .

# Forced truth values : Implicative and Assertive constraints

- ▶ *unit clause* : a clause that propagates one truth value to be satisfiable.
- ▶ *implicative constraint* : a constraint that propagates *at least* one truth value to be satisfiable.
- ▶ a constraint  $C$  is *implicative* iff  $\exists a_i x_i \in C$  such that  $\sum_{j \neq i} a_j < k$  or  $\sum a_j - k < a_i$ .
- ▶ Example :  $3x_1 + 2x_2 + x_3 \geq 5$  implies  $x_1$  and  $x_2$ 
  - ▶  $2 + 1 < 5$  so  $x_1$  must be satisfied.
  - ▶ Same thing on  $2x_2 + x_3 \geq 2$ .

# Forced truth values : Implicative and Assertive constraints

- ▶ *unit clause* : a clause that propagates one truth value to be satisfiable.
- ▶ *implicative constraint* : a constraint that propagates *at least* one truth value to be satisfiable.
- ▶ a constraint  $C$  is *implicative* iff  $\exists a_i x_i \in C$  such that  $\sum_{j \neq i} a_j < k$  or  $\sum a_j - k < a_i$ .
- ▶ Example :  $3x_1 + 2x_2 + x_3 \geq 5$  implies  $x_1$  and  $x_2$ 
  - ▶  $2 + 1 < 5$  so  $x_1$  must be satisfied.
  - ▶ Same thing on  $2x_2 + x_3 \geq 2$ .
- ▶ One can note that  $\sum a_j - k = 1$  so any literal  $x_i$  with a coef greater than 1 must be propagated.

$\sum a_j - k$  is called *poss* in Dixon *et al* [4] and *slack* in Chai and Kuehlmann [2].

- ▶ a constraint is *falsified* iff its *poss*  $< 0$  ;
- ▶ a constraint is *implicative* iff  $\exists a_i x_i$  such that *poss*  $- a_i < 0$ .

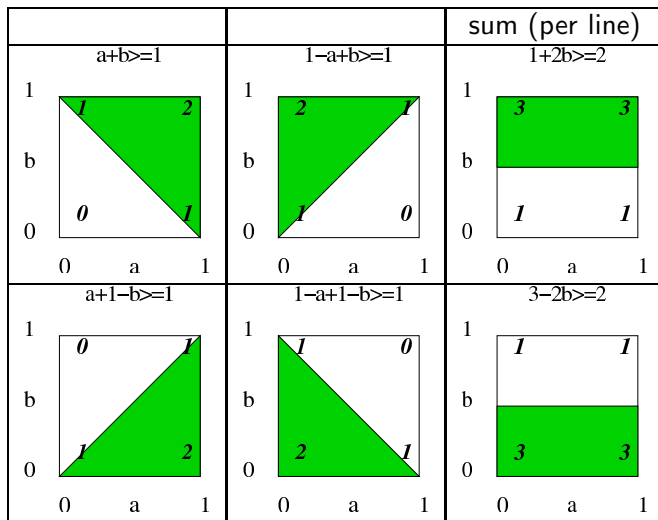
As a result, maintaining the value of *poss* during the search is sufficient to detect when a constraint becomes falsified or implicative.

# Requirements for a CDC<sub>o</sub>L solver

- ▶ Look Ahead : implicative constraints  
OPB, PBS, Satzoo, etc.  
Do not pass the pigeon hole test.
- ▶ Look Back :
  - ▶ Inference rule : cutting planes but ...
  - ▶ *assertive constraints*
  - ▶ Computing the backtrack level
- ▶ Data Structure : lazy one such as Watched Literals?

# Problem with cutting planes (why saturation ?)

Example :  $\{a \vee b, a \vee \neg b, \neg a \vee b, \neg a \vee \neg b\}$ .



# Requirements for a CDC<sub>o</sub>L solver

- ▶ Look Ahead : implicative constraints
- ▶ Look Back :
  - ▶ Inference rule : cutting planes but ...
  - ▶ *assertive constraints* : will become implicative when backtracking
  - ▶ Computing the backtrack level
- ▶ Data Structure : lazy one such as Watched Literals?

# Computing the backtrack level

- ▶ Just a *max* for clauses
- ▶ More complicated for LPBC : an LPB constraint may be assertive at different backtrack level. “It is important to backtrack at the highest level, then eventually undoing numbers of assignments, in order to maintain logical consistency within the solver (Chai and Kuehlmann)”.

Example :  $3x_1 + 2x_2 + x_3 \geq 4$  implies  $x_1$  is satisfied. As soon as  $x_2$  is falsified, the constraint will imply  $x_3$  to be satisfied, and vice-versa.

# Requirements for a CDC<sub>o</sub>L solver

- ▶ Look Ahead : implicative constraints
  - ▶ Look Back :
    - ▶ Inference rule : cutting planes but ...
    - ▶ *assertive constraints* : will become implicative when backtracking
    - ▶ Computing the backtrack level
- PRS, PBChaff and Galena.  
Pass the pigeon hole test.
- ▶ *Data Structure : lazy one such as Watched Literals?*

# Watched Literals for LPB constraints

Proposed in [2] and [5].

- General case :

Let  $M = \max(a_i)$

NbWatch = minimal number of literals  $x_i$  such that

$$\sum a_i \geq k + M.$$

- Cardinality constraints :

$$M = 1$$

$$NbWatch = k + 1$$

- Clauses :

$$M = 1$$

$$k = 1$$

$$NbWatch = 2$$

# Watched literals : consequences

- ▶ In LPB constraints, the number of WL is varying during the search.
- ▶ In cardinality constraints, the greater the degree, the greater the number of WL.
- ▶ Clauses are the best case !
- ▶ Big difference for cardinality or LPB constraint learning

# Practical issue : Detecting assertive constraints

- ▶ A decision literal is a UIP in  $CDC_L$  . This is no longer true for  $CDC_oL$  .
- ▶ A clause is unit at most once in each branch in the resolution tree. Not true for assertive constraints.
- ▶ Saturation and reduction have to be used to ensure that applying cutting planes during conflict analysis results in a conflicting constraint.

# Computing an assertive clause

- ▶ Let  $C$  be a falsified constraint
- ▶  $S = \text{lit}(C)_{>dl}$
- ▶  $D = \text{lit}(C)_{=dl}$
- 1 Pick the reason  $R$  for the latest assignment  $a$  in  $C$
- 2 Compute  $S = S \cup \text{lit}(R)_{>dl}$  and  $D = D \cup \text{lit}(R)_{=dl} \setminus \{a\}$
- ▶ Repeat 1 – 2 until  $|D| = 1$

# Computing an assertive constraint

- ▶ Let  $C$  be a falsified constraint
- ▶ Pick the reason  $R$  for the latest assignment  $a$  in  $C$
- ▶ compute  $\alpha$  and  $\alpha'$  to remove  $a$  from  $C$ .
- ▶ Weaken  $R$  if needed to ensure that the result of applying CP is falsified (**reduction**)
- ▶ Apply **cutting plane** :  $C = CP(C, R)$
- ▶ Apply **saturation**
- ▶ Keep track of the slack of the generated constraint
- ▶ Repeat until the above number is 0
- ▶ **Use arbitrary precision arithmetic to prevent overflow**

# Computing an assertive constraint

- ▶ Let  $C$  be a falsified constraint
- ▶ Pick the reason  $R$  for the latest assignment  $a$  in  $C$
- ▶ compute  $\alpha$  and  $\alpha'$  to remove  $a$  from  $C$ .
- ▶ Weaken  $R$  if needed to ensure that the result of applying CP is falsified (reduction)
- ▶ Apply cutting plane :  $C = CP(C, R)$
- ▶ Apply saturation
- ▶ Keep track of the slack of the generated constraint
- ▶ Repeat until the above number is 0
- ▶ Use arbitrary precision arithmetic to prevent overflow Not needed if reduced to cardinality constraint

# Example

$$\begin{cases} (C_1) & 5x_1 + 3x_2 + 2x_3 + 2x_4 + x_5 \geq 8 \\ (C_2) & 5\overline{x_1} + 3\overline{x_2} + 2\overline{x_3} + 2\overline{x_4} + \overline{x_5} \geq 5 \\ (C_3) & x_1 + x_3 + x_4 \geq 2 \end{cases}$$

$$\neg x_5^0, x_1^0[C_1], \neg x_4^1, x_3^1[C_3], x_2^1[C_1]$$

$$Poss(C_1) = +2, Poss(C_2) = -2$$

$$\text{Red. } x_1 : (C'_1) \quad 3x_2 + 2x_3 + 2x_4 + x_5 \geq 3 \quad \text{poss}=+2$$

$$\text{Red. } x_3 : (C''_1) \quad x_2 + x_4 + x_5 \geq 1 \quad \text{poss}=0$$

$$CP(C_2, C''_1) = 2\overline{x_1}^0 + 2\overline{x_3}^1 + x_4^1 + 2x_5^0 \geq 2$$

Assertive at decision level 0 ( $x_3$  is implied to 1).

Would learn  $\overline{x_1} + x_4 + x_5 \geq 1$  with clause learning.

# SAT4JPseudo : Replacing resolution by cutting planes ...

- ▶ Cutting planes are used during conflict analysis to generate an assertive constraints.
- ▶ Proposed first in Galena (Chai&Kuehlmann 2003) and PBCchaff (Dixon 2002/2004).
  - ▶ Cardinality approach preferred to Full CP
  - ▶ No management of integer overflow
  - ▶ Solvers no longer developed
- ▶ Two versions available : resolution based inference or cutting planes based inference.

# The Pseudo Boolean evaluations

- ▶ Organized by Olivier Roussel and Vasco Manquinho in 2005, 2006, 2007 and 2009
- ▶ **Uniform input format** : OPB files
- ▶ Independent assessment of the PB solvers
- ▶ Results freely available in details
- ▶ **first comprehensive repository of benchmarks**
- ▶ Various technologies used since 2006

## CDCL-like PB-solvers

**PBS** analysis mainly based on resolution

**Pueblo** analysis using both resolution and cutting planes

**Bsolo** branch'n bound PB-solver with lower bounding techniques and heuristics based on 0-1 programming.

## Some other solvers

**MiniSat+** translates PB-constraints into CNF to feed the powerful MiniSat solver

**glpPB** is a front end to the GNU linear programming toolkit.

# The solvers

## Summary of features

|              | Mini-Sat+ | SAT-4J              | Pueblo | PBS   | Bsolo | glpPB            |
|--------------|-----------|---------------------|--------|-------|-------|------------------|
| Input        | Clauses   | LPBC                | LPBC   | LPBC  | LPBC  | LPBC             |
| Inference    | Res.      | Full C.P. (boolean) | Mixed  | Mixed | Mixed | Full C.P. (real) |
| Optimization | L.S.      | L.S.                | L.S.   | L.S.  | B'n'B | Simplex          |

# Partial results of the PB05 evaluation

|       | Mi-<br>ni-<br>Sat+              | SAT-<br>4J                   | Pue-<br>blo            | PBS             | Bsolo            |                   |
|-------|---------------------------------|------------------------------|------------------------|-----------------|------------------|-------------------|
| Dec   | 43<br>(35)                      | 52<br>17                     | <b>61</b><br><b>42</b> | <b>61</b><br>28 | 36<br>8          | UNS<br>SAT        |
| Opt S | 10<br><b>176</b><br>(120)       | 10<br>120<br><b>(226)</b>    | 10<br>160<br>182       | 10<br>133<br>0  | 10<br>159<br>180 | UNS<br>OPT<br>SAT |
| Opt M | 0<br>24<br>(67)                 | <b>2</b><br>19<br><b>107</b> | 0<br><b>34</b><br>74   | 0<br>33<br>0    | 0<br>28<br>82    | UNS<br>OPT<br>SAT |
| Opt B | <b>103</b><br><b>26</b><br>(64) | 85<br>3<br><b>(171)</b>      | -                      | -               | 90<br>9<br>83    | UNS<br>OPT<br>SAT |

# Partial results of the PB06 evaluation

|       | PB06               |               |                    |             |            | ext.          |     |
|-------|--------------------|---------------|--------------------|-------------|------------|---------------|-----|
|       | Mi-<br>ni-<br>Sat+ | SAT4J<br>C.P. | Pue-<br>blo<br>1.4 | PBS<br>4.1L | Bsolo      | SAT4J<br>Res. |     |
| Dec.  | 172                | 79            | <b>204</b>         | 199         | 111        | 165           | UNS |
|       | 148                | 92            | <b>153</b>         | 144         | 118        | 121           | SAT |
| Opt S | 43                 | <b>54</b>     | 37                 | 29          | 40         | 35            | UNS |
|       | 405                | 357           | 385                | 352         | <b>409</b> | 367           | OPT |
|       | 250                | 303           | <b>323</b>         | 0           | 280        | 267           | SAT |
| Opt B | 38                 | 37            | -                  | -           | 30         | <b>40</b>     | UNS |
|       | 33                 | 57            | -                  | -           | 14         | <b>72</b>     | OPT |
|       | 52                 | 77            | -                  | -           | 69         | <b>72</b>     | SAT |

See <http://www.cril.univ-artois.fr/PB06/results/> for details

# Partial results of the PB07 evaluation

|                 | Mini-Sat+         | SAT4J C.P.              | Pueblo 1.4        | PBS4v2                  | Bsolv                   | SAT4J Res.                           |                   |
|-----------------|-------------------|-------------------------|-------------------|-------------------------|-------------------------|--------------------------------------|-------------------|
| Déc.<br>(#371)  | 170<br><b>146</b> | 88<br>56                | <b>202</b><br>139 | 197<br>130              | 137<br>141              | 167<br>110                           | UNS<br>SAT        |
| Opt S<br>(#807) | 35<br>208<br>302  | <b>47</b><br>147<br>397 | 31<br>180<br>389  | 23<br>141<br><b>416</b> | 33<br><b>237</b><br>325 | 35<br>156<br>376                     | UNS<br>OPT<br>SAT |
| Opt B<br>(#388) | 38<br>36<br>67    | 38<br>60<br>96          | -<br>-<br>-       | -<br>-<br>-             | 29<br>25<br>92          | <b>40</b><br><b>78</b><br><b>112</b> | UNS<br>OPT<br>SAT |

See <http://www.cril.univ-artois.fr/PB07/results/> for details

# Partial results of the PB06 evaluation

|                            | #   | MSat+ | SAT4J | Pueblo | PBS | Bsolo | glpPB |
|----------------------------|-----|-------|-------|--------|-----|-------|-------|
| SAT/UNSAT                  |     |       |       |        |     |       |       |
| pigeon                     | 20  | 2     | 20    | 13     | 20  | 2     | 20    |
| queens                     | 100 | 100   | 18    | 99     | 100 | 100   | 100   |
| tsp                        | 100 | 91    | 20    | 100    | 85  | 40    | 42    |
| fpga                       | 57  | 35    | 43    | 57     | 47  | 9     | 26    |
| uclid                      | 50  | 47    | 30    | 42     | 44  | 38    | 10    |
| OPT SMALLINT               |     |       |       |        |     |       |       |
| minprime                   | 156 | 124   | 104   | 118    | 103 | 106   | 52    |
| red.-mps                   | 273 | 46    | 70    | 63     | 27  | 54    | 58    |
| OPT BIGINT                 |     |       |       |        |     |       |       |
| factor.                    | 100 | 14    | 52    | -      | -   | 7     | -     |
| Ardal (subset sum problem) |     |       |       |        |     |       |       |
| Ardal_1                    | 12  | 10    | 2     | 0      | 3   | 2     | 0     |

# Good cases for SAT4JPseudo

- pigeon hole** solvers using resolution cannot solve them. A nice way to check the inference engine of the solvers.
- reduced mps** Those benchmarks are composed of real LPBC, so solvers with CP capabilities have good results on them.
- factorization** SAT4J Heuristics was lucky on half of the benchmarks, because of the way it initializes the phase of the variables to branch on according to the objective function.

# Bad cases for SAT4JPseudo

TSP and Weighted Queens problems contributed by Gayathri Namasivayam for PB06. Much more clauses than cardinality constraints or PB constraints.

One typical example from the Queens problem :

SAT4J C.P. timeout at 1800 seconds after only 7 restarts for  
2338 conflicts at 7 decisions/second

SAT4J Resolution 35 seconds after 16 restarts, 95829 conflicts at  
3320 decisions per seconds

The difference lies only in the conflict analysis procedure !

- ▶ Each constraint is now represented by it's cheaper representation (clause/card/PB).
- ▶ One can filter learned constraints to keep only clauses.
- ▶ the problem lies in learning constraints with increasing coefficients, not just in the cutting plane operation complexity.

# PB-constraints case : what we loose

boolean propagation lazy data structure for maintaining an alert value. All literals are watched (or would need to be dynamic).

Consequence : learning PB constraints does slow down the solver ! Solutions :

- ▶ Reduce learned clauses to Cardinality constraints (Galena, PBChaff)
- ▶ Learn both a clause and a PB constraint, then eventually remove the PB constraint (Pueblo).
- ▶ Learn clauses (Minisat+, PBS).

assertive constraints cannot syntactically be identified.

linear combination between two conflictual constraints doesn't necessary result in a falsified constraint ! Weakening can be needed to obtain a cutting plane.

# Interaction with SAT features

- ▶ SAT4J is updated with latest features found in state-of-the-art SAT solvers : in 2007, RSAT phase savings and Armin Biere's Rapid Restarts.
- ▶ Reason simplification generalized for resolution based PB solver.
- ▶ If those features improve SAT solving, **it does not seem to work for PB constraints.**
- ▶ The number of parameters keeps growing : decision heuristics, restart strategy, conflict analysis, learning scheme, magic constants, etc.
- ▶ **Is it really a good idea ?**

# Binate Covering Problem : The simplest PBO problem

Olivier Coudert : On Solving Covering Problems. DAC 1996 : 197-202

- ▶ Append to the set of clauses an objective function.

$$\min/\max : \sum_{i=1}^n a_i \cdot x_i$$

- ▶  $a_i$  integral coefficient
- ▶ boolean variables  $x_i$  interpreted with truth value  $\in \{0, 1\}$ .
- ▶  $\overline{x_i} = 1 - x_i$ .
- ▶ A note about complexity
  - ▶ SAT is NP-Complete
  - ▶ Binate Covering Problem is NP-Hard
- ▶ Bridges the gap between decision and optimization
- ▶ Still very close to SAT !

# Agenda

Anatomy of a modern CDCL SAT solver (continued)

Certifiable Unsat answers

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT



# From PB to Weighted Partial Max SAT

The minisat+ syndrom : is a SAT solver sufficient for all our needs?

- ▶ Once cardinality constraints and pseudo boolean constraints are managed in a solver, one can easily build a weighted partial Max SAT solver
- ▶ Add a selector variable  $s_i$  per constraint  $C_i : s_i \vee C_i$
- ▶ Objective function : minimize the sum of the selector variables.
- ▶ Partial MAX SAT : **do not add selector variables for hard clauses**
- ▶ Weighted MAXSAT : use a weighted sum instead of a sum.
  - ▶ Minimization problem : reported objective function value is wrong
  - ▶ Special case : **do not add new variables for unit weighted clauses**
- ▶ Using **Linear Search**, to have a SAT, SAT, ..., SAT, UNSAT pattern.

# SAT4J MAX-SAT : does it work ?

- ▶ Worst MAX-SAT solver in 2006
- ▶ Worst MAX-SAT solver in 2007
- ▶ Worst MAX-SAT solver in 2008 ?

# SAT4J MAX-SAT : does it work ?

- ▶ Worst MAX-SAT solver in 2006
- ▶ Worst MAX-SAT solver in 2007
- ▶ Worst MAX-SAT solver in 2008 ?

| Instance set          | #  | Clone      | IncWMaxSatz | LB-PSAT      | MiniMaxSAT | MSU1.2       | MSU4.0     |
|-----------------------|----|------------|-------------|--------------|------------|--------------|------------|
| MAXCLIQUE/RANDOM/     | 96 | 129.92(79) | 3.71 (96)   | 9.77 (96)    | 2.40 (96)  | 0.00 (0)     | 105.97(61) |
| MAXCLIQUE/STRUCTURED/ | 62 | 199.10(14) | 132.20(35)  | 124.40(32)   | 73.02(36)  | 4.90 (7)     | 105.21(13) |
| MAXONE/3SAT/          | 80 | 331.11(46) | 0.46 (80)   | 18.22(74)    | 1.28 (80)  | 52.86(40)    | 118.24(35) |
| MAXONE/STRUCTURED/    | 60 | 163.19(31) | 156.09(50)  | 2.28 (2)     | 30.48(60)  | 122.70(2)    | 3.34 (1)   |
| Instance set          | #  | PMS        | Toolbar3    | SAT4J MAXSAT | W-MaxSatz  | wmaxsat iccs |            |
| MAXCLIQUE/RANDOM/     | 96 | 68.11(96)  | 11.73(96)   | 113.94(52)   | 48.33(67)  | 44.24(80)    |            |
| MAXCLIQUE/STRUCTURED/ | 62 | 169.41(27) | 155.65(31)  | 50.46(13)    | 121.85(22) | 120.64(22)   |            |
| MAXONE/3SAT/          | 80 | 4.20 (80)  | 107.61(80)  | 96.67(31)    | 109.50(51) | 157.39(75)   |            |
| MAXONE/STRUCTURED/    | 60 | 88.40(35)  | 226.73(44)  | 10.04(60)    | 218.08(29) | 85.31(33)    |            |

Results on Partial Max SAT benchmarks, crafted category !

# SAT4J MAX-SAT : does it work ?

- ▶ Worst MAX-SAT solver in 2006
- ▶ Worst MAX-SAT solver in 2007
- ▶ Worst MAX-SAT solver in 2008 ?

| Instance set       | #    | Clone      | IncWMaxSatz | LB-PSAT      | MiniMaxSAT | MSU1.2        | MSU4.0      |
|--------------------|------|------------|-------------|--------------|------------|---------------|-------------|
| bcp-fir/           | 59   | 144.14(8)  | 227.14(34)  | 93.90(21)    | 258.44(14) | 49.16(46)     | 0.00 (0)    |
| bcp-hipp-yRa1/     | 1183 | 30.43(858) | 51.76(777)  | 47.64(851)   | 23.40(977) | 7.18 (1105)   | 0.29 (348)  |
| bcp-msp/           | 148  | 72.40(89)  | 61.61(99)   | 64.16(32)    | 48.95(104) | 4.84 (25)     | 22.96(79)   |
| bcp-mtg/           | 215  | 42.04(103) | 34.76(140)  | 48.78(163)   | 25.75(206) | 17.51(164)    | 0.43 (22)   |
| bcp-syn/           | 74   | 183.06(23) | 100.95(31)  | 63.46(34)    | 70.65(28)  | 51.15(31)     | 105.24(11)  |
| pbo-mqc/nencdr/    | 128  | 0.00 (0)   | 417.93(29)  | 0.00 (0)     | 190.52(60) | 50.36(54)     | 167.53(115) |
| pbo-mqc/nlogencdr/ | 128  | 285.18(84) | 306.55(59)  | 0.00 (0)     | 194.34(95) | 53.01(65)     | 110.86(128) |
| pbo-routing/       | 15   | 11.08(5)   | 25.94(5)    | 0.00 (0)     | 116.09(13) | 2.89 (15)     | 54.91(15)   |
| Instance set       | #    | PMS        | Toolbar3    | SAT4J MAXSAT | W-MaxSatz  | wmaxsatz_icss |             |
| bcp-fir/           | 59   | 91.63(12)  | 110.20(5)   | 13.36(10)    | 44.09(1)   | 13.75(6)      |             |
| bcp-hipp-yRa1/     | 1183 | 67.93(710) | 56.61(418)  | 12.20(1109)  | 84.33(270) | 45.01(444)    |             |
| bcp-msp/           | 148  | 70.56(85)  | 52.07(60)   | 8.78 (93)    | 81.33(59)  | 41.31(87)     |             |
| bcp-mtg/           | 215  | 73.75(172) | 62.58(68)   | 57.03(196)   | 25.31(34)  | 20.49(82)     |             |
| bcp-syn/           | 74   | 58.37(29)  | 69.84(14)   | 67.38(21)    | 109.08(7)  | 74.66(21)     |             |
| pbo-mqc/nencdr/    | 128  | 661.86(4)  | 0.00 (0)    | 180.58(102)  | 645.37(6)  | 0.00 (0)      |             |
| pbo-mqc/nlogencdr/ | 128  | 482.98(32) | 0.00 (0)    | 117.66(126)  | 426.50(14) | 0.00 (0)      |             |
| pbo-routing/       | 15   | 33.32(5)   | 0.00 (0)    | 26.45(9)     | 94.42(2)   | 18.47(1)      |             |

Results on Partial Max SAT benchmarks, industrial category !

# SAT4J MAX-SAT : does it work ?

- ▶ Worst MAX-SAT solver in 2006
- ▶ Worst MAX-SAT solver in 2007
- ▶ Worst MAX-SAT solver in 2008 ?

| Instance set             | #   | Clone        | IncWMaxsatz | LB-PSAT    | MiniMaxSAT    |
|--------------------------|-----|--------------|-------------|------------|---------------|
| AUCTIONS/AUC_PATHS/      | 88  | 33.87(88)    | 8.41 (88)   | 34.19(63)  | 29.79(88)     |
| AUCTIONS/AUC_REGIONS/    | 84  | 31.26(84)    | 1.94 (84)   | 2.00 (77)  | 1.70 (84)     |
| AUCTIONS/AUC_SCHEDULING/ | 84  | 172.04(75)   | 96.70(81)   | 46.11(68)  | 46.21(84)     |
| RANDOM-NET/              | 350 | 72.23 (236)  | 133.08(108) | 158.77(13) | 129.45(112)   |
| PSEUDO/factor/           | 186 | 7.95 (186)   | 0.07 (186)  | 107.66(9)  | 1.16 (186)    |
| PSEUDO/miplib/           | 16  | 115.13(5)    | 14.36(5)    | 227.00(4)  | 69.77(5)      |
| QCP/                     | 25  | 30.98(15)    | 11.26(20)   | 19.41(16)  | 26.20(20)     |
| WCSP/PLANNING/           | 71  | 10.33(36)    | 15.47(55)   | 23.34(1)   | 11.09(71)     |
| WCSP/SPOT5/DIR/          | 21  | 87.59(6)     | 25.83(4)    | 0.00 (0)   | 3.31 (3)      |
| WCSP/SPOT5/LOG/          | 21  | 14.92(6)     | 0.68 (4)    | 0.00 (0)   | 10.04(4)      |
| Instance set             | #   | SAT4J MAXSAT | Toolbar3    | W-MaxSatz  | wmaxsatz_icss |
| AUCTIONS/AUC_PATHS/      | 88  | 496.64(15)   | 49.19(88)   | 136.05(51) | 181.85(68)    |
| AUCTIONS/AUC_REGIONS/    | 84  | 166.47(76)   | 6.58 (84)   | 5.63 (72)  | 5.28 (84)     |
| AUCTIONS/AUC_SCHEDULING/ | 84  | 316.81(49)   | 75.75(82)   | 61.55(70)  | 74.61(82)     |
| RANDOM-NET/              | 350 | 330.83(13)   | 128.09(50)  | 89.73 (22) | 29.92(48)     |
| PSEUDO/factor/           | 186 | 3.30 (186)   | 234.43(10)  | 10.94(115) | 0.20 (20)     |
| PSEUDO/miplib/           | 16  | 13.00(6)     | 3.10 (4)    | 1.64 (2)   | 1.19 (4)      |
| QCP/                     | 25  | 6.14 (25)    | 193.50(12)  | 14.64(5)   | 226.71(15)    |
| WCSP/PLANNING/           | 71  | 6.55 (71)    | 23.91(52)   | 7.63 (4)   | 1.23 (48)     |
| WCSP/SPOT5/DIR/          | 21  | 76.35(3)     | 129.24(5)   | 0.00 (0)   | 14.79(2)      |
| WCSP/SPOT5/LOG/          | 21  | 63.76(3)     | 111.67(4)   | 0.00 (0)   | 19.60(2)      |

Results on Weighted Partial Max SAT benchmarks, crafted category !

# SAT4J MAX-SAT : does it work ?

- ▶ Worst MAX-SAT solver in 2006
- ▶ Worst MAX-SAT solver in 2007
- ▶ Worst MAX-SAT solver in 2008 ? **No !**
  - ▶ CDCL solvers perform badly on random benchmarks
  - ▶ Using SAT4J Resolution instead of SAT4J CP does gives results similar to Toolbar.
  - ▶ Some good numbers : QCP, ...,.
  - ▶ **Without any SLS preprocessing nor bound reasoning !**
- ▶ Is it possible to reproduce the minisat+ case in MaxSAT ?

## Industrial Partial Max-SAT

| Instance_set            | #   | pm2         | SAT4JMaxsat | wpm1      | wbo        | MSUnCore   |
|-------------------------|-----|-------------|-------------|-----------|------------|------------|
| CircuitTraceCompaction/ | 4   | 442.99(3)   | 431.75(3)   | 0.00(0)   | 0.00(0)    | 0.00(0)    |
| HaplotypeAssembly/      | 6   | 34.52(5)    | 0.00(0)     | 31.70(5)  | 9.72(5)    | 31.52(5)   |
| PROTEIN_INS/            | 12  | 56.83(2)    | 319.63(2)   | 44.11(1)  | 11.96(1)   | 534.08(1)  |
| bcp-fir/                | 59  | 13.60(57)   | 7.99(10)    | 27.99(56) | 112.01(37) | 53.11(49)  |
| bcp-hipp-yRa1/SU/       | 38  | 101.65(28)  | 268.15(8)   | 21.44(11) | 109.84(10) | 57.08(10)  |
| bcp-hipp-yRa1/simp/     | 138 | 8.10(136)   | 19.95(132)  | 1.40(130) | 3.56(131)  | 1.35(130)  |
| bcp-msp/                | 148 | 91.31(94)   | 7.54(95)    | 12.25(42) | 45.35(24)  | 102.03(23) |
| bcp-mtg/                | 215 | 1.36(215)   | 67.03(196)  | 6.03(170) | 15.67(173) | 6.38(166)  |
| bcp-syn/                | 74  | 12.64(38)   | 98.81(20)   | 5.70(31)  | 31.76(30)  | 35.20(32)  |
| pbo-mqc/nencdr/         | 128 | 233.02(92)  | 318.54(114) | 82.62(50) | 73.63(63)  | 35.74(48)  |
| pbo-mqc/nlogencdr/      | 128 | 148.66(108) | 176.42(125) | 69.84(75) | 76.27(67)  | 56.44(67)  |
| pbo-routing/            | 15  | 1.18(15)    | 223.31(12)  | 0.99(15)  | 0.96(15)   | 0.63(15)   |
| Solved instances        | 965 | 793         | 717         | 586       | 556        | 546        |

| Instance_set            | WMaxSatZ-2.5 | WMaxSatZ-1.6 | IUT_BCMB_LSWM | Clone      | IUT_BCMB_WM |
|-------------------------|--------------|--------------|---------------|------------|-------------|
| CircuitTraceCompaction/ | 0.00(0)      | 0.00(0)      | 0.00(0)       | 0.00(0)    | 0.00(0)     |
| HaplotypeAssembly/      | 0.00(0)      | 0.00(0)      | 0.00(0)       | 0.00(0)    | 0.00(0)     |
| PROTEIN_INS/            | 0.26(1)      | 0.22(1)      | 0.29(1)       | 26.99(1)   | 1.49(1)     |
| bcp-fir/                | 6.99(7)      | 14.83(7)     | 7.34(7)       | 118.32(8)  | 10.53(7)    |
| bcp-hipp-yRa1/SU/       | 0.00(0)      | 0.00(0)      | 0.00(0)       | 0.00(0)    | 0.00(0)     |
| bcp-hipp-yRa1/simp/     | 218.83(44)   | 242.21(43)   | 202.00(44)    | 105.18(74) | 187.72(40)  |
| bcp-msp/                | 30.18(94)    | 38.61(94)    | 30.35(93)     | 77.38(91)  | 32.50(90)   |
| bcp-mtg/                | 118.80(153)  | 48.11(143)   | 84.52(119)    | 45.70(104) | 58.06(102)  |
| bcp-syn/                | 148.71(22)   | 133.83(22)   | 139.79(22)    | 183.55(23) | 127.35(24)  |
| pbo-mqc/nencdr/         | 544.37(59)   | 597.77(58)   | 512.29(54)    | 0.00(0)    | 0.00(0)     |
| pbo-mqc/nlogencdr/      | 331.39(95)   | 383.59(92)   | 353.89(95)    | 333.79(86) | 0.00(0)     |
| pbo-routing/            | 5.82(5)      | 7.27(5)      | 5.63(2)       | 10.97(5)   | 13.64(2)    |
| Solved instances        | 965          | 480          | 465           | 437        | 392         |

# Another recent approach to tackle MAXSAT : MSU

Z. Fu and S. Malik, On solving the partial MAX-SAT problem, in International Conference on Theory and Applications of Satisfiability Testing, August 2006, pp. 252-265.

- ▶ Based on unsat core detection
- ▶ Introduction of new variables
- ▶ Use of cardinality constraints
- ▶ Several improved algorithms developed since then :
  - ▶ Marques-Silva&Planes'07/'08
  - ▶ Marques-Silva&Manquinho'08
  - ▶ Ansótegui, Bonet& Levy'09

# Fu&Malik's Algorithm : msu1.0

$x_6, x_2$

$\neg x_6, x_2$

$\neg x_2, x_1$

$\neg x_1$

$\neg x_6, x_8$

$x_6, \neg x_8$

$x_2, x_4$

$\neg x_4, x_5$

$x_7, x_5$

$\neg x_7, x_5$

$\neg x_5, x_3$

$\neg x_3$

Example CNF formula

# Fu&Malik's Algorithm : msu1.0

$x_6, x_2$

$\neg x_6, x_2$

$\neg x_6, x_8$

$x_6, \neg x_8$

$x_7, x_5$

$\neg x_7, x_5$

$\neg x_2, x_1$

$\neg x_1$

$x_2, x_4$

$\neg x_4, x_5$

$\neg x_5, x_3$

$\neg x_3$

Formula is **UNSAT** ; Get unsat core

# Fu&Malik's Algorithm : msu1.0

$x_6, x_2$

$\neg x_6, x_2$

$\neg x_2, x_1, b_1$

$\neg x_1, b_2$

$\neg x_6, x_8$

$x_6, \neg x_8$

$x_2, x_4, b_3$

$\neg x_4, x_5, b_4$

$x_7, x_5$

$\neg x_7, x_5$

$\neg x_5, x_3, b_5$

$\neg x_3, b_6$

$$\sum_{i=1}^6 b_i \leq 1$$

Add **blocking variables** and AtMost1 constraint

# Fu&Malik's Algorithm : msu1.0

$$x_6, x_2$$

$$\neg x_6, x_2$$

$$\neg x_2, x_1, b_1$$

$$\neg x_1, b_2$$

$$\neg x_6, x_8$$

$$x_6, \neg x_8$$

$$x_2, x_4, b_3$$

$$\neg x_4, x_5, b_4$$

$$x_7, x_5$$

$$\neg x_7, x_5$$

$$\neg x_5, x_3, b_5$$

$$\neg x_3, b_6$$

$$\sum_{i=1}^6 b_i \leq 1$$

Formula is (again) **UNSAT** ; Get unsat core

# Fu&Malik's Algorithm : msu1.0

$$x_6, x_2, b_7$$

$$\neg x_6, x_2, b_8$$

$$\neg x_2, x_1, b_1, b_9$$

$$\neg x_1, b_2, b_{10}$$

$$\neg x_6, x_8$$

$$x_6, \neg x_8$$

$$x_2, x_4, b_3$$

$$\neg x_4, x_5, b_4$$

$$x_7, x_5, b_{11}$$

$$\neg x_7, x_5, b_{12}$$

$$\neg x_5, x_3, b_5, b_{13}$$

$$\neg x_3, b_6, b_{14}$$

$$\sum_{i=1}^6 b_i \leq 1 \quad \sum_{i=7}^{14} b_i \leq 1$$

Add new **blocking variables** and AtMost1 constraint

# Fu&Malik's Algorithm : msu1.0

$$x_6, x_2, b_7$$

$$\neg x_6, x_2, b_8$$

$$\neg x_2, x_1, b_1, b_9$$

$$\neg x_1, b_2, b_{10}$$

$$\neg x_6, x_8$$

$$x_6, \neg x_8$$

$$x_2, x_4, b_3$$

$$\neg x_4, x_5, b_4$$

$$x_7, x_5, b_{11}$$

$$\neg x_7, x_5, b_{12}$$

$$\neg x_5, x_3, b_5, b_{13}$$

$$\neg x_3, b_6, b_{14}$$

$$\sum_{i=1}^6 b_i \leq 1 \quad \sum_{i=7}^{14} b_i \leq 1$$

Instance is now SAT

# Fu&Malik's Algorithm : msu1.0

$$x_6, x_2, b_7$$

$$\neg x_6, x_2, b_8$$

$$\neg x_2, x_1, b_1, b_9$$

$$\neg x_1, b_2, b_{10}$$

$$\neg x_6, x_8$$

$$x_6, \neg x_8$$

$$x_2, x_4, b_3$$

$$\neg x_4, x_5, b_4$$

$$x_7, x_5, b_{11}$$

$$\neg x_7, x_5, b_{12}$$

$$\neg x_5, x_3, b_5, b_{13}$$

$$\neg x_3, b_6, b_{14}$$

$$\sum_{i=1}^6 b_i \leq 1 \quad \sum_{i=7}^{14} b_i \leq 1$$

MaxSATsolution is  $|\varphi| - \mathcal{I} = 12 - 2 = 10$

- ▶ Clauses characterized as :
  - ▶ **Initial** : derived from clauses in  $\varphi$
  - ▶ **Auxiliary** : added during execution of algorithm
    - ▶ E.g. clauses from cardinality constraints
- ▶ While exist unsatisfiable cores
  - ▶ Add **fresh** set  $B$  of blocking variables to **non-auxiliary soft** clauses in core
  - ▶ Add **new** AtMost1 constraint

$$\sum_{b_i \in B} b_i \leq 1$$

- ▶ At most 1 blocking variable from set  $B$  can take value 1
- ▶ MaxSATsolution is  $|\varphi| - \mathcal{I}$ , where  $\mathcal{I}$  is number of iterations

# Main interest of the approach

- ▶ Takes advantage of unsat core computation
- ▶ Works well in practice on real MAXSAT problems
- ▶ Completely orthogonal to “reasoning-based” MAX SAT approaches.

# Agenda

Anatomy of a modern CDCL SAT solver (continued)

Certifiable Unsat answers

A note on SAT solving on multi-core processors

Some results from the SAT Competition 2009

Can we extend CDCL with a new proof system ?

A note about MAXSAT

Practicing SAT



# I am convinced : where should I start ?

- ▶ The first step is to pick up a SAT solver.
- ▶ Play with simple examples
- ▶ Translate your problem into SAT
- ▶ Experiment
- ▶ If it does not work, try another encoding !

# Where to find SAT solvers ?

Several open source SAT solvers exist :

**Minisat (C++)** [www.minisat.se](http://www.minisat.se) Presumably the most widely used within the SAT community. Used to be the best general purpose SAT solver. A large community around the solver.

**Picosat (C)/Precosat (C++)**  
<http://fmv.jku.at/software/index.html>  
Award winner in 2007 and 2009 of the SAT competition, industrial category.

**SAT4J (Java)** <http://www.sat4j.org>. For Java users. Far less efficient than the two others.

**UBCSAT (C)** <http://www.satlib.org/ubcsat/> Very efficient stochastic local search for SAT.

<http://www.satcompetition.org> Both the binaries and the source code of the solvers are made available for research purposes.



# Where to find examples ?

- ▶ The Satisfiability Library : <http://www.satlib.org/>
- ▶ The SAT competition web site :  
<http://www.satcompetition.org/>
- ▶ Google “SAT benchmarks”

# How to feed a SAT solver ?

All SAT solvers support the very simple DIMACS CNF input format :

$$(a \vee b \vee \neg c) \wedge (\neg b \vee \neg c)$$

will be translated into

```
p cnf 3 2
1 2 -3 0
-2 -3 0
```

The first line is of the form

```
p cnf <maxVarId> <numberOfClauses>
```

Each variable is represented by an integer, negative literals as negative integers, 0 is the clause separator.

# My original problem is not in CNF !

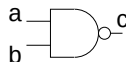
- ▶ If it is a propositional formula/circuit : encode it into a CNF
- ▶ Even for numerical constraints, a translation into CNF may be worthwhile (c.f. Minisat+).
- ▶ Else move to 0-1 LP, SMT, etc.

# Representing Boolean Circuits / Formulas I

- ▶ Satisfiability problems can be defined on Boolean circuits/formulas
- ▶ Can represent circuits/formulas as CNF formulas [Tseitin'68]
  - ▶ For each (simple) gate, CNF formula encodes the **consistent** assignments to the gate's inputs and output
    - ▶ Given  $z = OP(x, y)$ , represent in CNF  $z \leftrightarrow OP(x, y)$
  - ▶ CNF formula for the circuit is the conjunction of CNF formula for each gate

# Representing Boolean Circuits / Formulas II

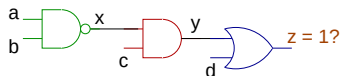
$$\begin{aligned}\varphi_c &= (a \wedge b \leftrightarrow \neg c) \\ &\equiv (a \wedge b \rightarrow \neg c) \wedge (\neg c \rightarrow a \wedge b) \\ &\equiv (\neg a \vee \neg b \vee \neg c) \wedge (a \vee c) \wedge (b \vee c)\end{aligned}$$



$$\varphi_t = (\neg r \vee t) \wedge (\neg s \vee t) \wedge (r \vee s \vee \neg t)$$

# Representing Boolean Circuits / Formulas III

- ▶ CNF formula for the circuit is the conjunction of the CNF formula for each gate
  - ▶ Can specify objectives with additional clauses



$$\begin{aligned}\varphi = & (a \vee x) \wedge (b \vee x) \wedge (\neg a \vee \neg b \vee \neg x) \wedge \\ & (x \vee \neg y) \wedge (c \vee \neg y) \wedge (\neg x \vee \neg c \vee y) \wedge \\ & (\neg y \vee z) \wedge (\neg d \vee z) \wedge (y \vee d \vee \neg z) \wedge \\ & (z)\end{aligned}$$

- ▶ Note :  $z = d \vee (c \wedge (\neg(a \wedge b)))$ 
  - ▶ No distinction between Boolean circuits and formulas



Peter Barth.

A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization.

Technical Report MPI-I-95-2-003, Max-Planck-Institut für Informatik, Saarbrücken, 1995.



Donald Chai and Andreas Kuehlmann.

A fast pseudo-boolean constraint solver.

In *ACM/IEEE Design Automation Conference (DAC'03)*, pages 830–835, Anaheim, CA, 2003.



Heidi E. Dixon Matthew L. Ginsberg.

Combining satisfiability techniques from AI and OR.

In *The Knowledge Engineering Review* 15, page 53, 2000.



Heidi E. Dixon Matthew L. Ginsberg.

Inference methods for a pseudo-boolean satisfiability solver.

In *Proceedings of The Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 635–640, 2002.



Heidi E. Dixon Matthew L. Ginsberg Andrew J. Parkes.

Generalizing boolean satisfiability i : Background and survey of existing work.

In *Journal of Artificial Intelligence Research* 21, 2004.



S. Prestwich.

Randomised backtracking for linear pseudo-boolean constraint problems.

In *Proceedings of Fourth International Workshop on Integration of AI and OR techniques in Constraint Programming for Combinatorial Optimisation Problems (CP-AI-OR'2002)*, pages 7–20, 2002.



S. Prestwich.

Incomplete dynamic backtracking for linear pseudo-boolean problems : Hybrid optimization techniques.

*Annals of Operations Research*, 130(1-4) :57–73, August 2004.



Fadi A. Aloul Arathi Ramani Igor L. Markov Karem A. Sakallah.

Symmetry-breaking for pseudo-boolean formulas.

In *International Workshop on Symmetry on Constraint Satisfaction Problems (SymCon)*, pages 1–12, County Cork, Ireland, 2003.



Christian Thiffault, Fahiem Bacchus, and Toby Walsh.  
Solving Non Clausal Formulas with DPLL Search.

In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming (CP'2004)*, Toronto, Canada, September 2004.



J. P. Walser.

# Solving Linear Pseudo-Boolean Constraint Problems with Local Search.

In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 269–274, 1997.

