GlueMiniSat2.2.5: A fast SAT solver with an aggressive acquiring strategy of glue clauses

Hidetomo NABESHI MA*
Koji IWANUMA*
Katsumi INOUYE**

* University of Yamanashi
** National Institute of Informatics
GlueMiniSat 2.2.5

Boolean satisfiability testing program (a SAT solver)

MiniSat2.2 + Glucose1.0 + $\alpha$

- Variant of LBD which is an evaluation criteria of learnt clauses
- Aggressive restart strategy to get good learnt clauses

- Application category of SAT 2011 competition
  - 1st in CPU time UNSAT class
  - 2nd in CPU time SAT+UNSAT class
  - 2nd in Wall-clock time UNSAT class (including parallel SAT solvers)
Outline

- SAT
- CDCL (Conflict Driven Clause Learning) Algorithm
  [Silva 99, Bayardo 97]
- Evaluation Criteria of Learnt Clauses
  - Literal Blocks Distance [Audemard 09]
- GlueMiniSat2.2 & 2.2.5
- Experimental Results
  - SAT 2009 Application
  - SAT 2011 Application
  - Covering Arrays
- Conclusion
SAT

- Boolean satisfiability testing
- First NP-complete problem [Cook, 1971]
- Usually, represented in CNF formula

\[(a \lor b \lor c) \land (\neg a \lor \neg c) \land (\neg a \lor c)\]

**Purpose**

Determines the satisfiability of a given formula
Progress in SAT Solvers

- Dramatic performance improvement from the late 90s
- Can handle problems consisting of millions of variables
- Various techniques in the state-of-the-art SAT solvers
  - Basic procedure: DPLL [Davis+ 62]
  - Conflict driven clause learning (CDCL) [Silva+ 99, Bayardo+ 97]
  - Backjumping [Silva+ 99, Bayardo+ 97]
  - Fast unit propagation by watched literals [Moskewicz +01]
  - Effective variable selection heuristics [Moskewicz+ 01]
  - Restart strategy [Gomes+ 98, Luby+ 93]
  - Phase caching [Pipatsrisawat+ 07]
  - Fast identification of satisfied clauses
    [Jain+ 07][Schubert+ 07][Sorensson+ 08]
- Canonical SAT solver: **MiniSat** [Eén+ 03]
Problem Solving by SAT

- Planning / scheduling
- Hardware / software verification
- Theorem proving
- Constraint satisfaction / optimization problems
  - Sugar [Tamura 08] which is a SAT-based CSP solver got first places in 3 categories of 2009 CSP solver competition
Outline

- SAT
- CDCL (Conflict Driven Clause Learning) Algorithm [Silva 99, Bayardo 97]
- Evaluation Criteria of Learnt Clauses
  - Literal Blocks Distance [Audemard 09]
- GlueMiniSat2.2.0 & 2.2.5
- Experimental Results
  - SAT 2009 Application
  - SAT 2011 Application
  - Covering Arrays
- Conclusion
CDCL Algorithm [Silva 99, Bayardo 97]

(1) Finds unit clauses and satisfies them (unit propagation)
(2) If no unit clause, selects an unassigned var and assigns 1 or 0
(3) If a conflict occurs, analyzes a cause of the conflict and learns the negation of the cause as a clause, and then backjumps to the level in which the learned clause becomes unit

\[
\begin{align*}
\{+1\} & \quad \text{← UC} \\
\{+1, +2, +4\} & \\
\{+2, +9\} & \quad \text{← UC} \\
\{-1, +4, +9\} & \\
\{-2, +3\} & \quad \text{← UC} \\
\{-2, -5, +6\} & \quad \text{← UC} \\
\{-1, -4, +5\} & \quad \text{← UC} \\
\{-5, -3, +7\} & \quad \text{← UC} \\
\{-6, +8\} & \quad \text{Conflict!} \\
\{-7, -8\} & 
\end{align*}
\]

**Decision Stack**

<table>
<thead>
<tr>
<th>Level</th>
<th>Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>+2 +3</td>
</tr>
<tr>
<td>2</td>
<td>+4 +5 +6 +7</td>
</tr>
</tbody>
</table>

2 is selected and assigned as true by heuristics
4 is selected and assigned as true by heuristics
Conflicts Driven Clause Learning

Decision Stack

If $+2 \land +5 \land +3$, then contradicts

Learns the clause $-2 \lor -5 \lor -3$
Management of Learnt Clauses

- Learnt clauses are useful to prevent same conflicts.
- However there is a trade-off:
  - It is difficult to preserve all learnt clauses since it consumes memory and unit propagations becomes slow.
  - If learnt clauses are not preserved, the search process repeats same conflicts and becomes slow.
- Hence, CDCL solvers reduce learnt clauses periodically.

How to select learnt clauses which will be preserved?
Evaluation Criteria of Learnt Clauses

- **Length**
  - Short learnt clauses have high pruning power

- **Activity**
  - **Chaff** [Moskewicz+ 01], **MiniSat** [Eén+ 03]
  - Defines *activity* for each learnt clause, removes clauses whose activity is less than a certain threshold
  - Activity is raised when the clause is used to produce a contradiction
  - Least recently used (LRU) learnt clauses are removed

- **LBD (Literal Blocks Distance)**
  - LBD is a measure to evaluate the possibility of use of learnt clauses in the future
  - **Glucose1.0** [Audemard and Simon, 09]
    - 1\textsuperscript{st} in UNSAT class and 2\textsuperscript{nd} in SAT+UNSAT class at Application category of SAT 2009 Competition
LBD

Learnt clause \(\{ L_1, L_2, L_3, L_4, L_5, L_6 \}\)

Decision level

\[
\begin{array}{ccccccc}
Lv 0 & & & & & & \\
Lv 1 & & & & & & \\
Lv 2 & -L_4 & -L_6 & -L_5 & & & \\
Lv 3 & & & & & & \\
Lv 4 & & & & & & \\
Lv 5 & -L_2 & -L_3 & & & & \\
Lv 6 & & & & & & \\
Lv 7 & & & & & & \\
\end{array}
\]

Implication

Decision

Conflict
A set of variables assigned at the same DLV is called a block. LBD of a clause \( C \) is defined as \# blocks in \( C \).

- Variables in a block have possibility that they will be assigned as false at the same time by unit propagations.
- LBD can be considered as a generalization of length criteria.
Glue Clauses

- Especially, clauses whose LBD=2 are called **glue clauses**.
- Glue clauses promote unit propagations even if they are long.

\[ \{ L_1, L_2, L_3, L_4, L_5, L_6 \} \]

**Decision level**

\[
\begin{array}{cccccc}
7 & 5 & 5 & 5 & 5 & 5 \\
\end{array}
\]

*L1* is propagated when \( L_2 \sim L_6 \) are assigned as false.

**Glucose** never removes **glue clauses**.
# GlueMiniSat2.2

GlueMiniSat2.2 = MiniSat2.2 + Glucose1.0

<table>
<thead>
<tr>
<th></th>
<th>MiniSat 2.0</th>
<th>MiniSat 2.2</th>
<th>Glucose 1.0</th>
<th>GlueMiniSat 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Var selection heuristics</strong></td>
<td>VSIDS</td>
<td>VSIDS</td>
<td>VSIDS</td>
<td>VSIDS</td>
</tr>
<tr>
<td><strong>Randomness</strong></td>
<td>2%</td>
<td>0%</td>
<td>2%</td>
<td>0%</td>
</tr>
<tr>
<td><strong>Evaluation criteria of learnts</strong></td>
<td>LRU</td>
<td>LRU</td>
<td>LBD</td>
<td>LBD</td>
</tr>
<tr>
<td><strong>Reduction strategy of learnts</strong></td>
<td>Exponential (#C/3)*1.1(^r)</td>
<td>Exponential (#C/3)*1.1(^d)</td>
<td>Linear 20000+500x</td>
<td>Linear 20000+10000x</td>
</tr>
<tr>
<td><strong>Restart strategy</strong></td>
<td>Exponential 100 * 1.5(^r)</td>
<td>Luby</td>
<td>Dynamic (LBD)</td>
<td>Dynamic (LBD)</td>
</tr>
<tr>
<td><strong>Phase caching</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Fast identification of satisfied clauses</strong></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Memory management</strong></td>
<td>malloc</td>
<td>Single area</td>
<td>malloc</td>
<td>Single area</td>
</tr>
</tbody>
</table>
Experimental Results in SAT 2009 Application

CPU time limit [sec]

# solved

- **MiniSat2.0**
- **MiniSat2.2**
- **glucose1.0**
- **GlueMiniSat2.2.0**
From Development of GlueMiniSat 2.2

We got the following assumptions:

(a) Important to promote acquiring clauses with small LBD

(b) For unsatisfiability proof, important to preserve useful learned clauses as many as possible

(a) Aggressive restart strategy

(b) Expanding a set of preserved learnt clauses which are never removed
Expanding Preserved Learnts

- Low performance if it holds learnts with \( \text{LBD} \leq 3 \)
A clause $C$ is glue

1. when $C$ is generated from a conflict and the LBD is 2
2. when $C$ is used in unit propagations and the LBD is 2 (LBD is recalculated by the current truth assignment)
### Pseudo LBD

<table>
<thead>
<tr>
<th>When?</th>
<th>(1) generated from a conflict</th>
<th>(2) used in unit propagations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leant clause</td>
<td>{(L_1, L_2, L_3, L_4, L_5, L_6)}</td>
<td>{(L_1, L_2, L_3, L_4, L_5, L_6)}</td>
</tr>
<tr>
<td>Decision Lv</td>
<td>7 5 5 5 4 4</td>
<td>7 7 5 5 4 4</td>
</tr>
<tr>
<td>LBD</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- **GlueMiniSat holds learnt clauses with pseudo LBD \(\leq 3\)**
  - A learnt clause from (1) always contains unit literal block. Hence, the clause somewhat promotes unit propagations even if LBD is 3.
  - A learnt clause from (2) may not contain unit literal block. Hence, GlueMiniSat holds learnts with pseudo LBD \(\leq 3\) (LBD \(\leq 2\)).
LBD vs Pseudo LBD

CPU Time [sec] vs # solved

- MiniSat2.0
- MiniSat2.2
- glucose1.0
- GlueMiniSat2.2
- LBD <= 3
- Pseudo LBD <= 3

Slightly improved
From Development of GlueMiniSat 2.2

We got the following assumptions:

(a) Important to promote acquiring glue clauses

(b) For unsatisfiability proof, important to preserve learned clauses which will be used in the future

(a) Aggressive restart strategy

(b) Expanding a set of preserved learnt clauses
Restart Strategy of GlueMiniSat

- Restart strategy for DLVs
  
  Local avg. of DLVs over the last 50 conflicts $\times 1.0 >$ Global avg. of DLVs

- Restart strategy for LBDs (same as Glucose1.0)
  
  Local avg. of LBDs over the last 50 learnt clauses $\times 0.8 >$ Global avg. of LBDs

Restarts if either condition is satisfied
Purpose is to reduce DLVs and get small LBD clauses
Results of Restart by DLV and LBD

![Graph showing CPU time vs. number of solved problems for different solvers with annotations indicating slight improvement with restart by DLV and LBD.]
<table>
<thead>
<tr>
<th></th>
<th>MiniSat 2.0</th>
<th>Glucose 1.0</th>
<th>MiniSat 2.2</th>
<th>GlueMiniSat 2.2</th>
<th>Pseudo LBD + AR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong># solved</strong></td>
<td>109 (49 / 60)</td>
<td>133 (52 / 81)</td>
<td>141 (60 / 81)</td>
<td>154 (60 / 94)</td>
<td>161 (61 / 100)</td>
</tr>
<tr>
<td><strong>Average time [sec]</strong></td>
<td>193</td>
<td>206</td>
<td>167</td>
<td>197</td>
<td>199</td>
</tr>
<tr>
<td><strong>Restart speed [confs/restart]</strong></td>
<td>14229</td>
<td>1152</td>
<td>528</td>
<td>456</td>
<td>117</td>
</tr>
</tbody>
</table>

- Enhanced the strength for UNSAT
- Restarts very aggressively

Environment: Mac mini, Core 2 Duo 1.83GHz, 2GB RAM 1000 CPU sec / instance
## SAT 2011 Application

### CPU Time

<table>
<thead>
<tr>
<th>Type</th>
<th>Gold</th>
<th>Silver</th>
<th>Bronze</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAT+UNSAT</strong></td>
<td>Glucose2.0</td>
<td><strong>GlueMiniSat</strong></td>
<td>Lingeling</td>
</tr>
<tr>
<td><strong>UNSAT</strong></td>
<td><strong>GlueMiniSat</strong></td>
<td>Glucose2.0</td>
<td>QuteRSat</td>
</tr>
</tbody>
</table>

- **Strong for UNSAT**
- **Weak for SAT** (19th of 20 solvers in final stage)

### WC Time

<table>
<thead>
<tr>
<th>Type</th>
<th>Gold</th>
<th>Silver</th>
<th>Bronze</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAT+UNSAT</strong></td>
<td>Plingeling //</td>
<td>CryptoMiniSat //</td>
<td>ppfolio //</td>
</tr>
<tr>
<td><strong>SAT</strong></td>
<td>ppfolio //</td>
<td>Plingeling //</td>
<td>contrasat (MiniSat hack)</td>
</tr>
<tr>
<td><strong>UNSAT</strong></td>
<td>CryptoMiniSat //</td>
<td><strong>GlueMiniSat</strong></td>
<td>Plingeling //</td>
</tr>
</tbody>
</table>

// means a parallel solver which uses multiple CPU cores
Conclusion

- **GlueMiniSat** is strong for UNSAT proof
  - GlueMiniSat holds more glue clauses than Glucose
    - Prevents losing useful clauses required to prove unsatisfiability
  - GlueMiniSat restarts more aggressively than Glucose
    - Contributes to acquire good learnt clauses

Future Work

- Comparison with strong algorithms for SAT
- Extension from sequential to parallel