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

Hidetomo NABESHIMA[*]

Koji IWANUMA[*]

Katsumi INOUE[**]

[*] University of Yamanashi

[**] National Institute of Informatics

GlueMiniSat 2.2.5

Boolean satisfiability testing program (a SAT solver)

MiniSat2.2 + Glucose1.0 + α

[Eén and Sörensson 03] [Audemard and Simon 09]

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

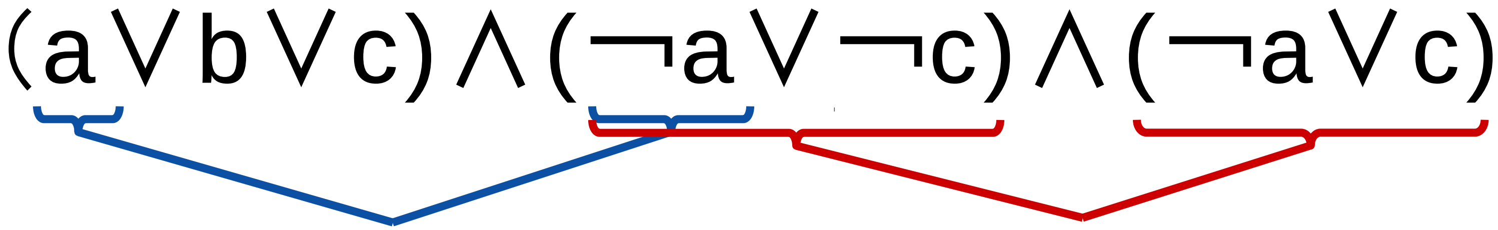
- Application category of SAT 2011 competition
  - 1st in CPU time UNSAT class
  - 2nd in CPU time SAT+UNSAT class
  - 2nd in Wall-clock time UNSAT class (including parallel SAT solvers)

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



$$(a \lor b \lor c) \land (\neg a \lor \neg c) \land (\neg a \lor c)$$

Literals

Boolean variables or their negations

Clauses

Disjunctions of literals

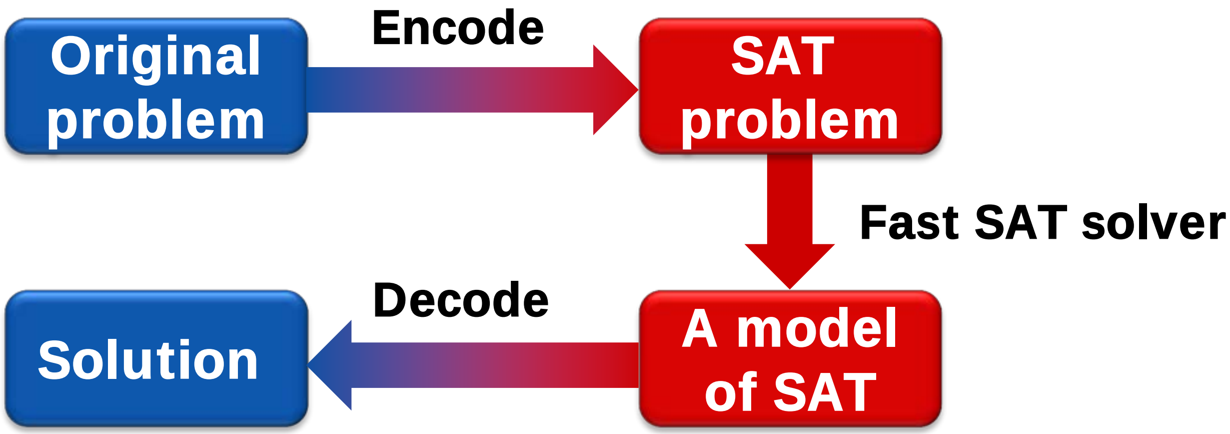
Purpose

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



Original problem

Encode

SAT problem

Fast SAT solver

A model of SAT

Decode

Solution

- 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

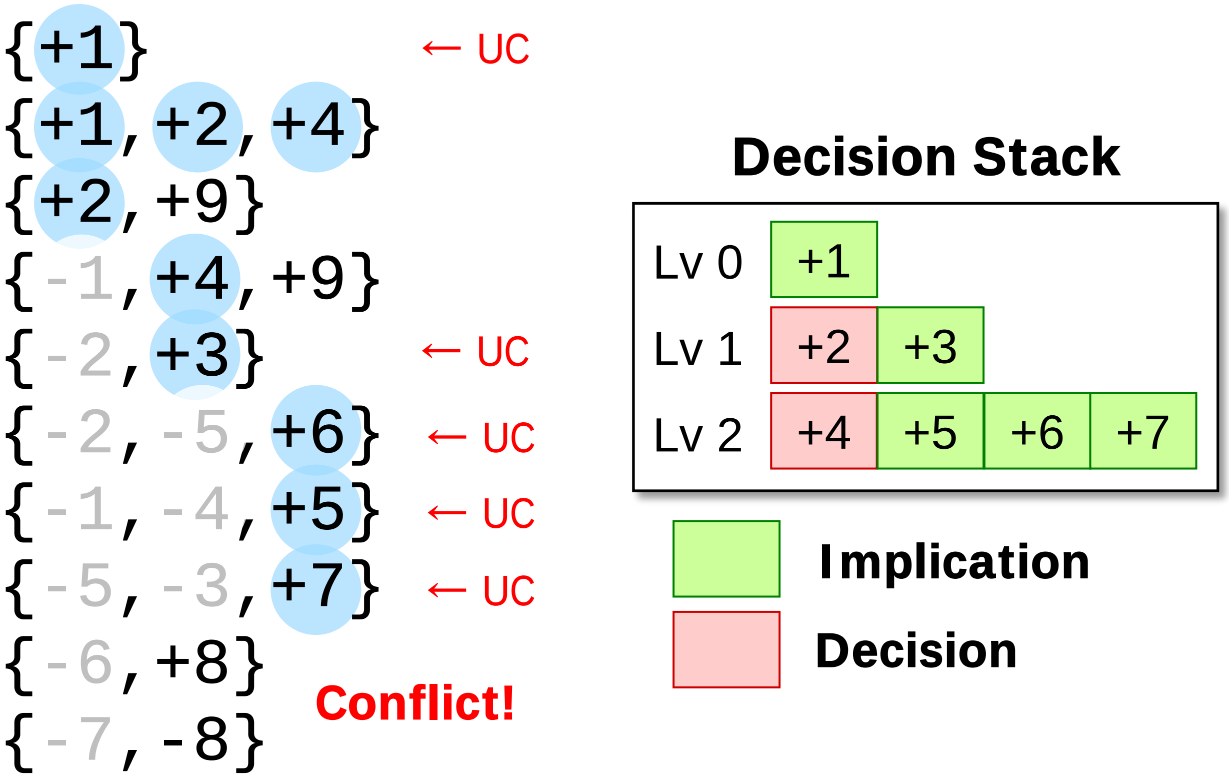
- SAT
- CDCL (Conflict Driven Clause Learning) Algorithm [Silva 99, Bayardo 97]
- Evaluation Criteria of Learnt Clauses
  - Literal Blocks Distance [Audemard 09]
- GlueMiniSat2.2.0 & 2.2.5
- Experimental Results
  - SAT 2009 Application
  - SAT 2011 Application
  - Covering Arrays
- Conclusion

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



{+1} ← UC
{+1, +2, +4}
{+2, +9}
{-1, +4, +9}
{-2, +3} ← UC
{-2, -5, +6} ← UC
{-1, -4, +5} ← UC
{-5, -3, +7} ← UC
{-6, +8}
{-7, -8}

Conflict!

Decision Stack

Lv 0	+1			
Lv 1	+2	+3		
Lv 2	+4	+5	+6	+7

🟩 **Implication**

🟥 **Decision**

2 is selected and assigned as true by heuristics

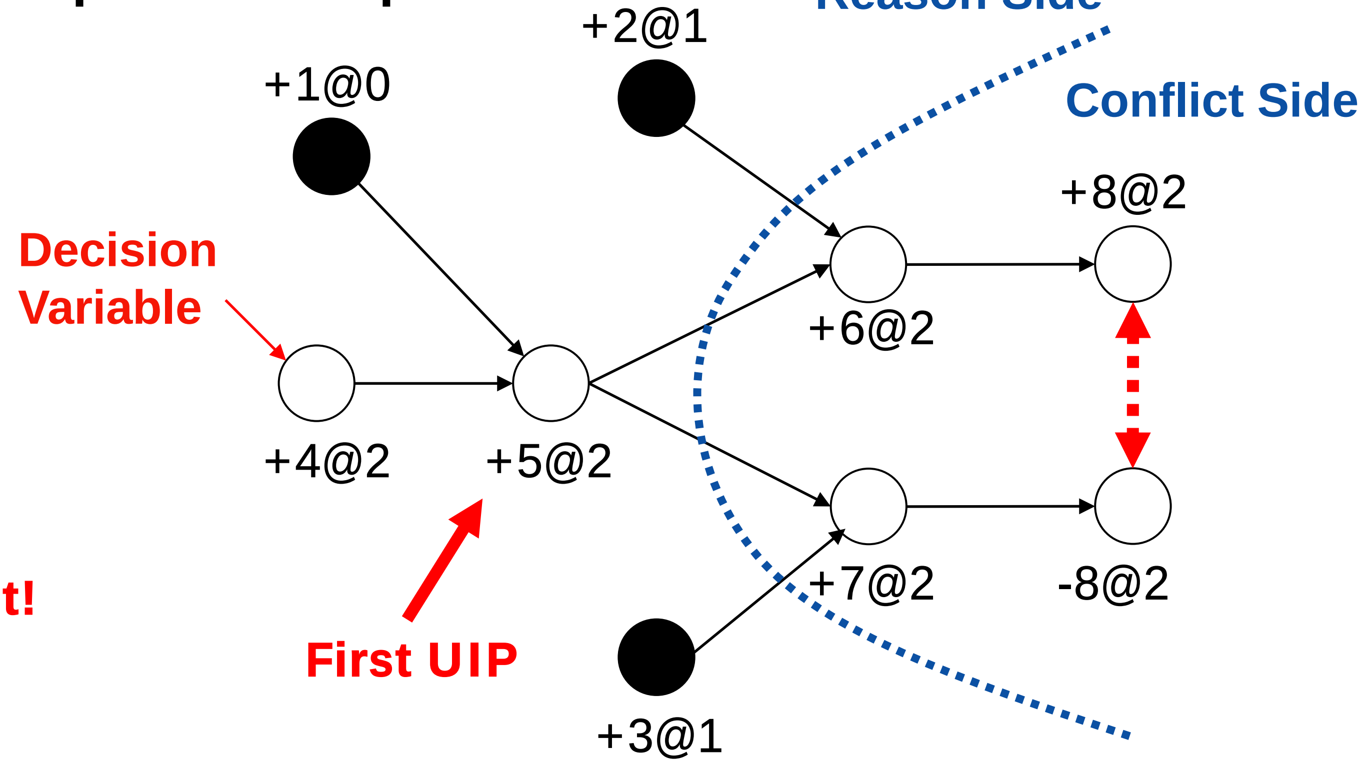
4 is selected and assigned as true by heuristics

Conflict Driven Clause Learning

{+1}
{+1, +2, +4}
{+2, +9}
{-1, +4, +9}
{-2, +3}
{-2, -5, +6}
{-1, -4, +5}
{-5, -3, +7}
{-6, +8}
{-7, -8}

Conflict!

Implication Graph



Decision Stack

Lv 0	+1			
Lv 1	+2	+3		
Lv 2	+4	+5	+6	+7

If +2 ∧ +5 ∧ +3, then contradicts

⬇

Learns the clause -2 ∨ -5 ∨ -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?

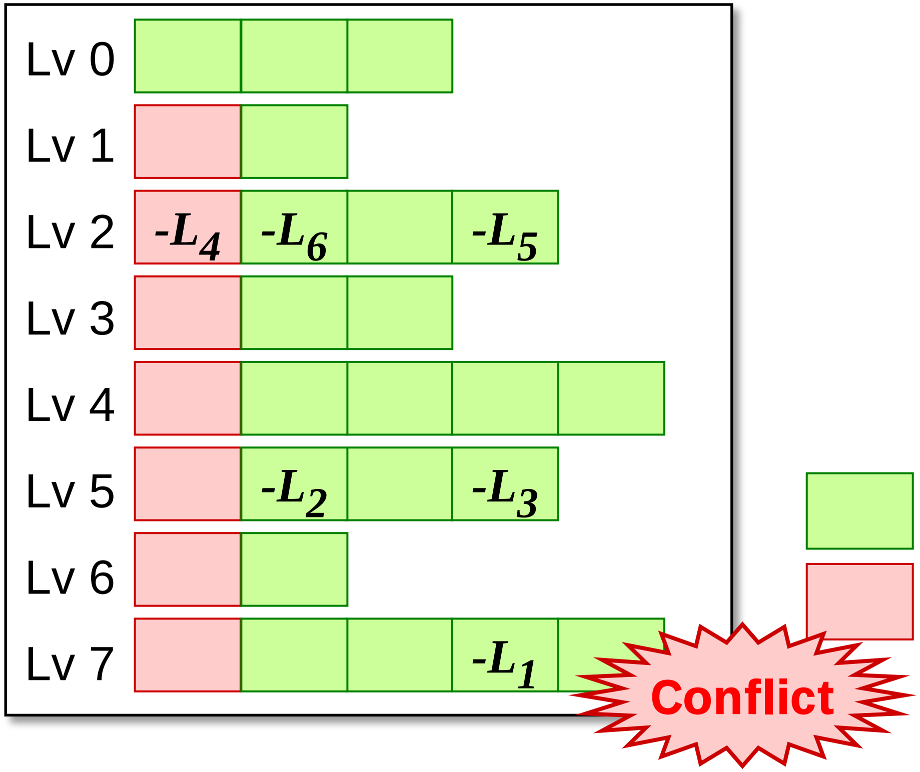
Evaluation Criteria of Learnt Clauses

- **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]
    - ✓ 1st in UNSAT class and 2nd 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	**7**	**5**	**5**	**2**	**2**	**2**

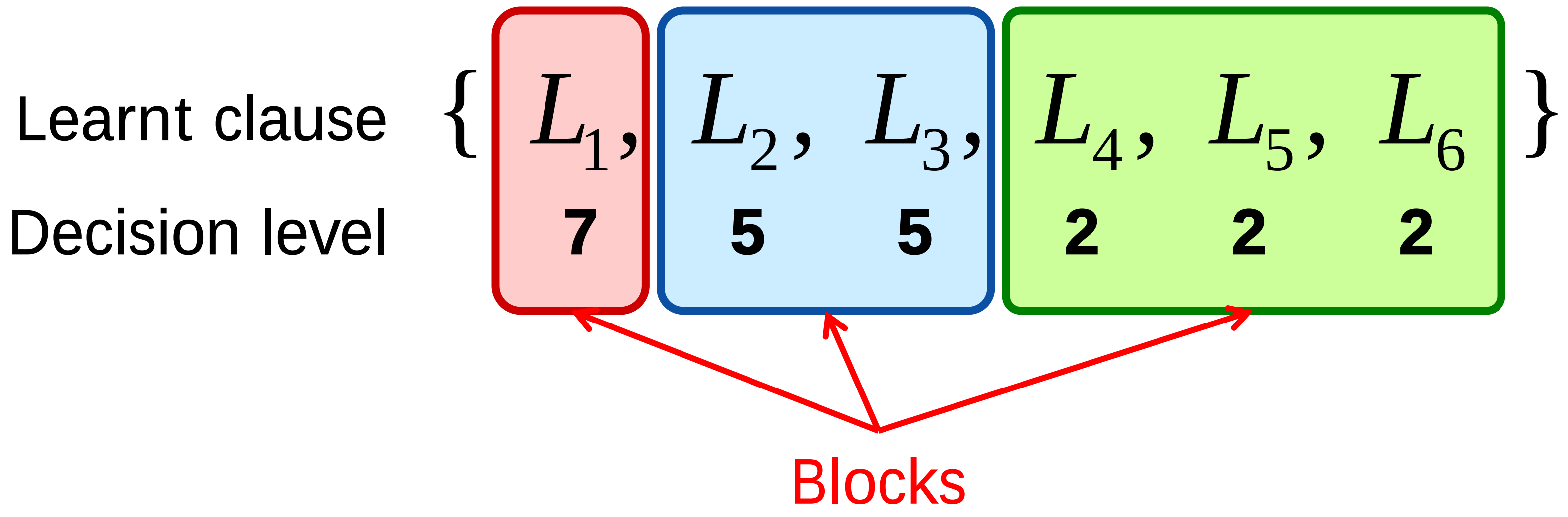
Decision Stack



🟩 **Implication**

🟥 **Decision**

LBD



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

Decision level **7** **5** **5** **2** **2** **2**

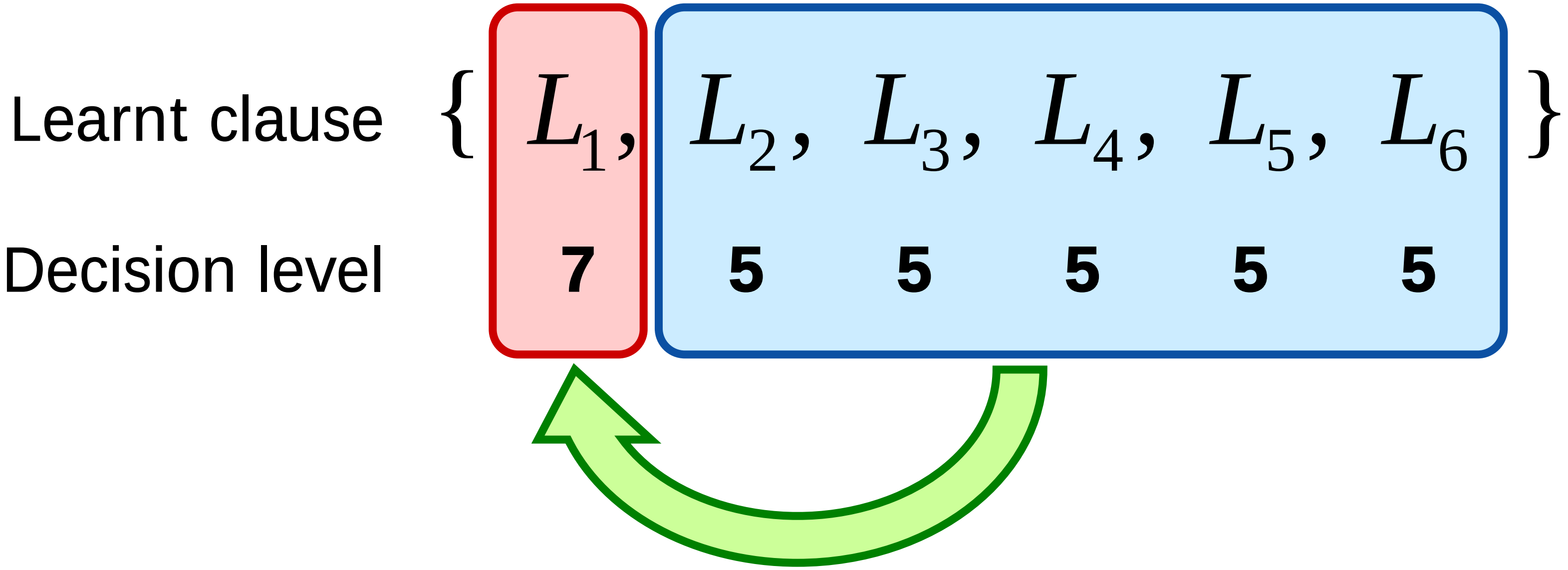
Blocks

- 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**



Learnt clause	$\{ L_1,$	$L_2,$	$L_3,$	$L_4,$	$L_5,$	$L_6 \}$
Decision level	**7**	**5**	**5**	**5**	**5**	**5**

L1 is propagated when ***L2*** ～ ***L6*** are assigned as false

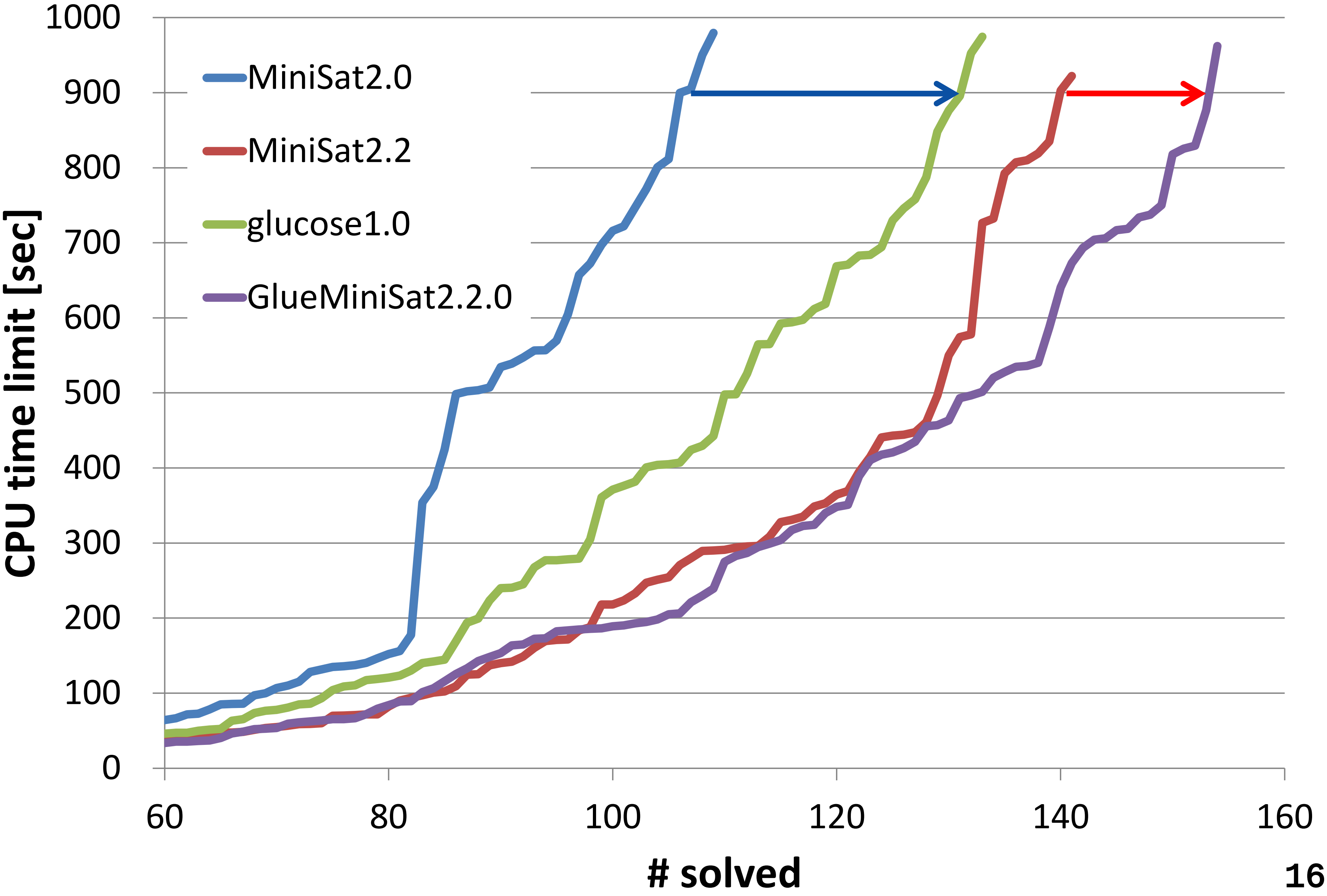
Glucose never removes **glue clauses**

GlueMiniSat2.2

GlueMiniSat2.2 = MiniSat2.2 + Glucose1.0

	MiniSat 2.0	**MiniSat 2.2**	**Glucose 1.0**	**GlueMiniSat 2.2**
Var selection heuristics	VSIDS	VSIDS	VSIDS	VSIDS
Randomness	2%	0%	2%	0%
Evaluation criteria of learnts	LRU	LRU	LBD	LBD
Reduction strategy of learnts	Exponential $(\#C/3)*1.1^r$	Exponential $(\#C/3)*1.1^d$	Linear 20000+500x	Linear 20000+10000x
Restart strategy	Exponential $100 * 1.5^r$	Luby	Dynamic (LBD)	Dynamic (LBD)
Phase caching		✓	✓	✓
Fast identification of satisfied clauses		✓	✓	✓
Memory management	malloc	Single area	malloc	Single area

Experimental Results in SAT 2009 Application



From Development of GlueMiniSat 2.2

We got the following assumptions:

(a) Important to promote acquiring clauses with small LBD

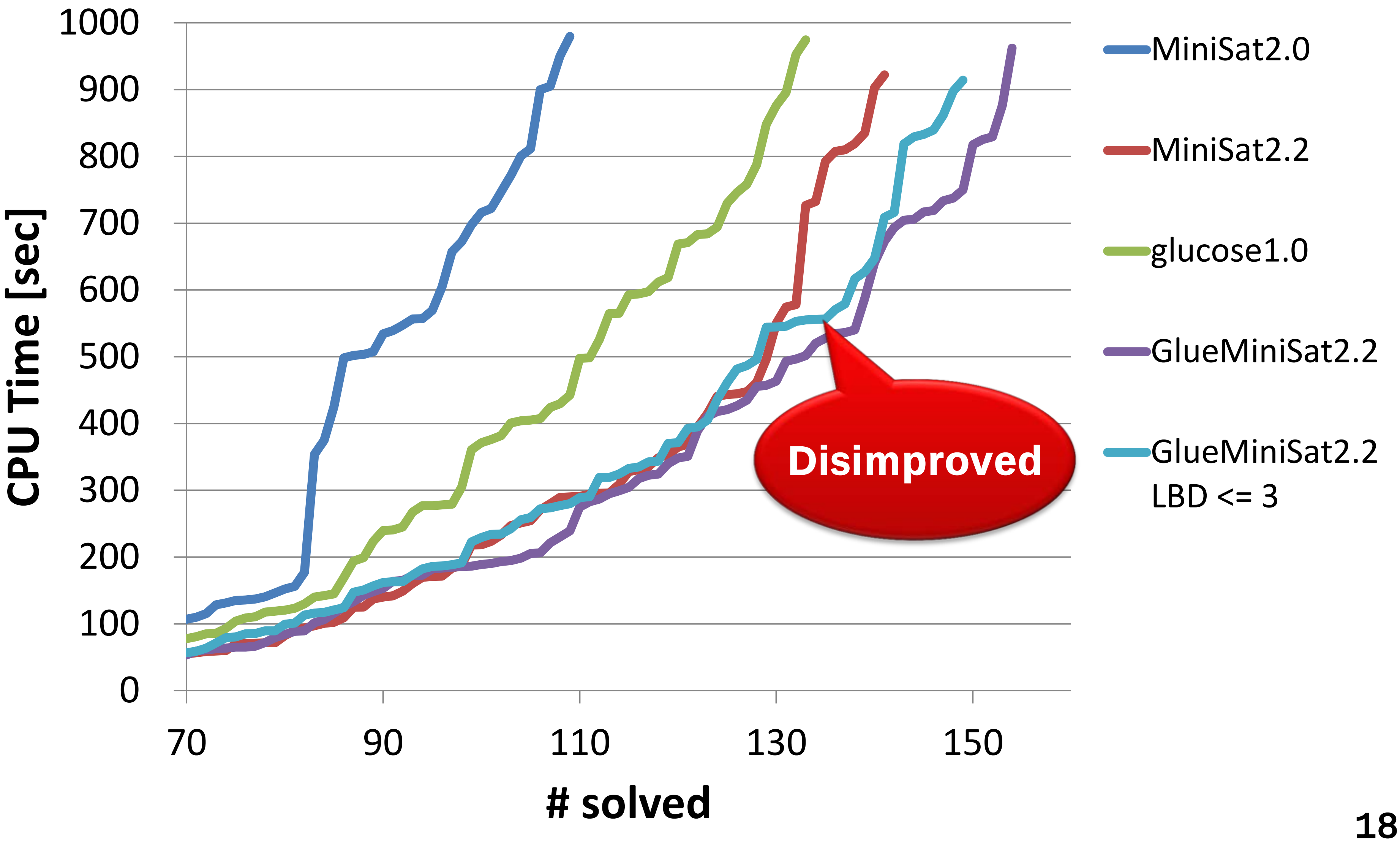
(b) For unsatisfiability proof, important to preserve useful learned clauses as many as possible

(a) Aggressive restart strategy

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

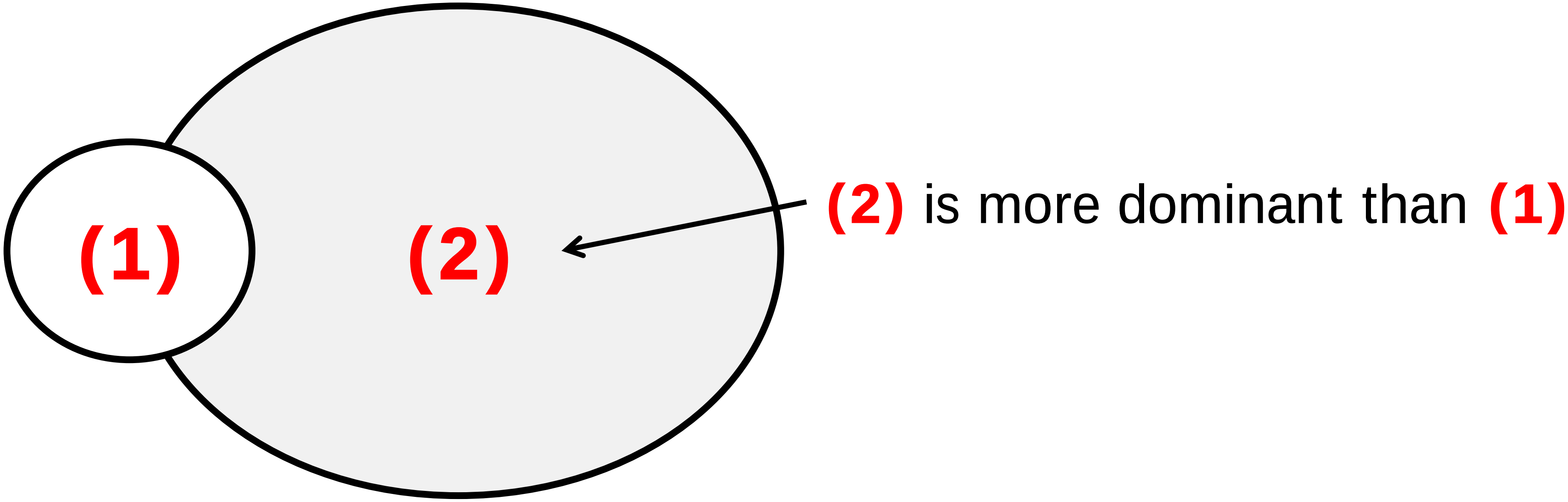
Expanding Preserved Learnts

- Low performance if it holds learnts with **LBD ≦ 3**



Details of LBD Computation

- 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)



(2) is more dominant than **(1)**

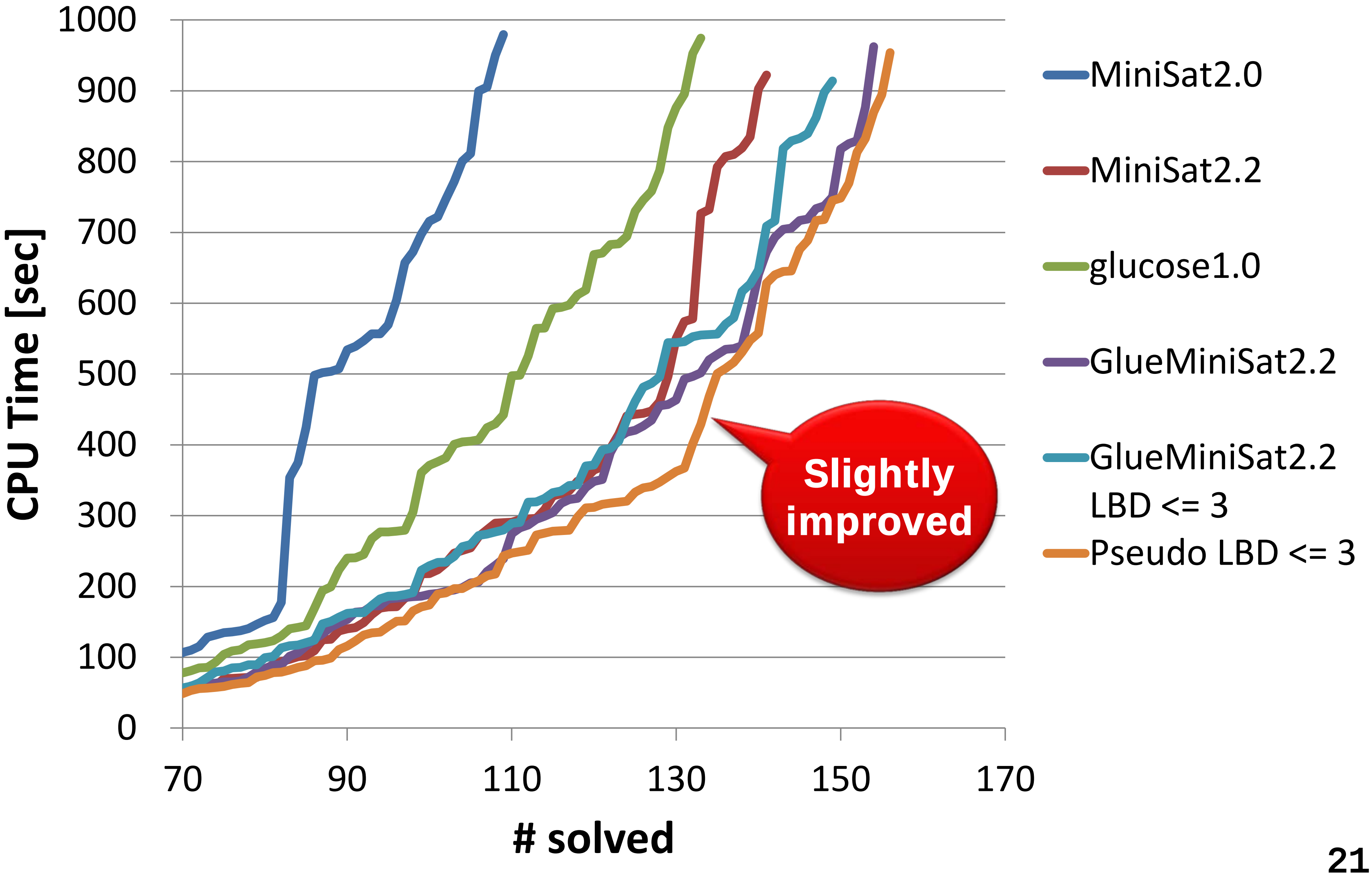
Glue clauses

Pseudo LBD

When?	**(1) generated from a conflict**	**(2) used in unit propagations**
Leant clause	$\{L_1, L_2, L_3, L_4, L_5, L_6\}$	$\{L_1, L_2, L_3, L_4, L_5, L_6\}$
Decision Lv	**7 5 5 5 4 4**	**7 7 5 5 4 4**
LBD	**3**	**3**

- GlueMiniSat holds learnt clauses with pseudo LBD ≦ 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 ≦ 3 (LBD ≦ 2)

LBD vs Pseudo LBD



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 * **1.0** > Global avg. of DLVs

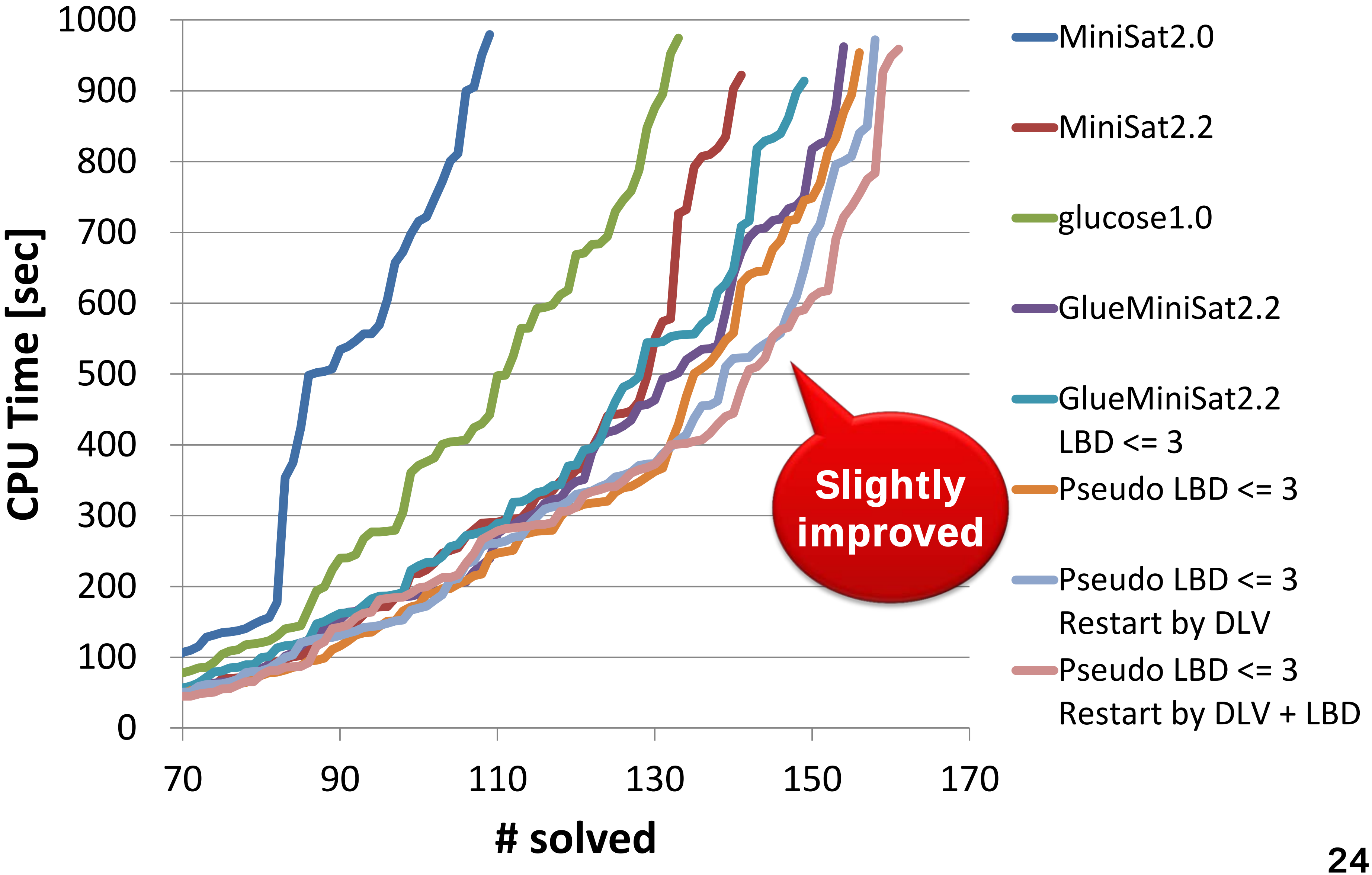
- Restart strategy for LBDs (same as Glucose1.0)

Local avg. of LBDs over
the last 50 learnt clauses * **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



Slightly improved

Experimental Results in SAT 2009 Application

	MiniSat 2.0	Glucose 1.0	MiniSat 2.2	GlueMiniSat	
				2.2	Pseudo LBD + AR
# solved (SAT / UNSAT)	109 (49 / 60)	133 (52 / 81)	141 (60 / 81)	154 (60 / 94)	161 (61 / 100)
Average time [sec]	193	206	167	197	199
Restart speed [confs/restart]	14229	1152	528	456	117

- 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

	Gold	Silver	Bronze
SAT+UNSAT	**Glucose2.0**	**GlueMiniSat**	**Lingeling**
S			**at64**
UNSAT	**GlueMiniSat**	**Glucose2.0**	**QuteRSat**

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

WC Time

	Gold	Silver	Bronze
SAT+UNSAT	Plingeling //	CryptoMiniSat //	ppfolio //
SAT	ppfolio //	Plingeling //	contrasat (MiniSat hack)
UNSAT	CryptoMiniSat //	GlueMiniSat	Plingeling //

// 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