GLUCOSE 2.1
Aggressive – but Reactive – Clause Database Management, Dynamic Restarts

Gilles Audemard – Laurent Simon

POS – 16 june 2012

A short history of GLUCOSE

- **2009 – Version 1.0**
  - Built on top of MINISAT 2.0
  - Learnt clause measure usefulness: LBD
  - Agressive cleaning strategy
  - Dynamic restarts

A short history of GLUCOSE

- **2009 – Version 1.0**
  - Built on top of MINISAT 2.0
  - Learnt clause measure usefulness: LBD
  - Agressive cleaning strategy
  - Dynamic restarts

- **2011 – Version 2.0**
  - Built on top of MINISAT 2.2 ($\approx$ 30% faster)
  - Focus on cleaning strategy
    - More agressive cleaning strategy
    - Dynamic
    - Protect promising clauses
  - Reducing learnt clauses

A short history of GLUCOSE

- **2009 – Version 1.0**
  - Built on top of MINISAT 2.0
  - Learnt clause measure usefulness: LBD
  - Agressive cleaning strategy
  - Dynamic restarts

- **2011 – Version 2.0**
  - Built on top of MINISAT 2.2 ($\approx$ 30% faster)
  - Focus on cleaning strategy
    - More agressive cleaning strategy
    - Dynamic
    - Protect promising clauses
  - Reducing learnt clauses

- **2012 – Version 2.1**
  - Focus on restarts

Litteral Block Distance

An observation

- Before CDCL solvers: solvers implement ideas (lookahead, Mom's heuristics...)

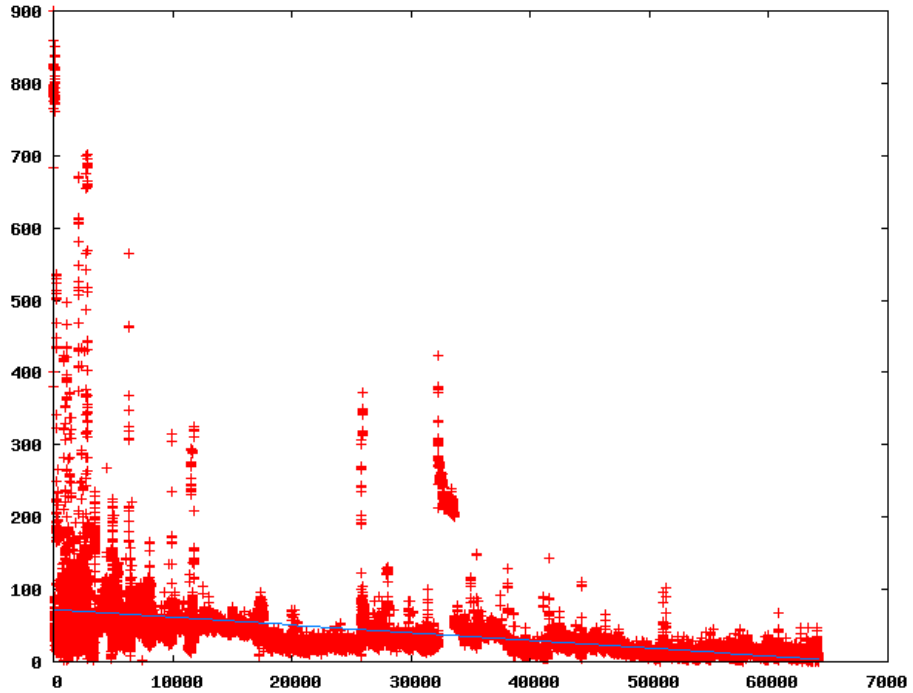
  explaining performances was simple

- With CDCL: lookback solvers (VSIDS heuristics, learning,...)

  explaining performances is hard

We need strong empirical studies in order to understand and improve performances

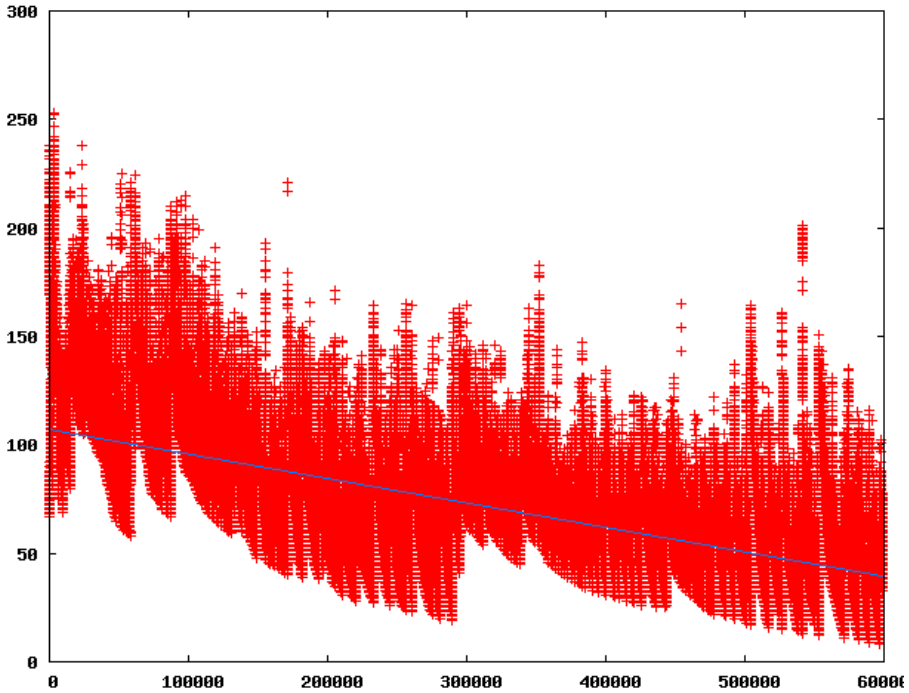
Some plots . . .



een-pico-prop05-50 – UNSAT – 13,000 vars and 65,000 clauses

- For each conflict, we store the decision level where it occurs
- We also compute the linear regression on these points
- Gives an idea of the global behavior of the computation

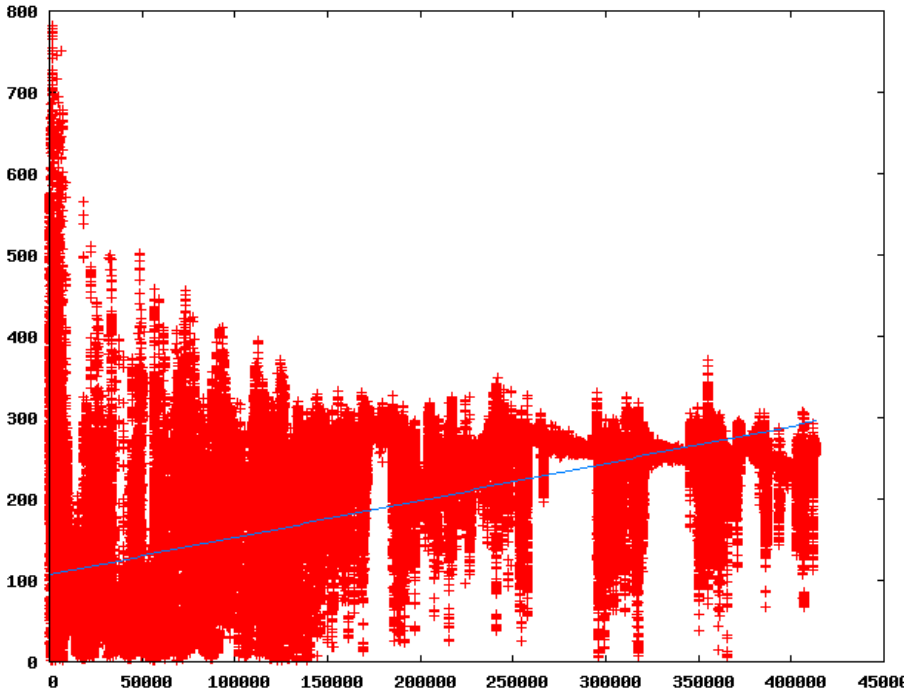
Some plots . . .



grieu-vmpc-s05-25 – SAT – 625 vars and 76,000 clauses

- For each conflict, we store the decision level where it occurs
- We also compute the linear regression on these points
- Gives an idea of the global behavior of the computation

Some plots . . .



mizh-sha0-35-3 – SAT – 20,000 vars and 120,000 clauses

- For each conflict, we store the decision level where it occurs
- We also compute the linear regression on these points
- Gives an idea of the global behavior of the computation

Remarks

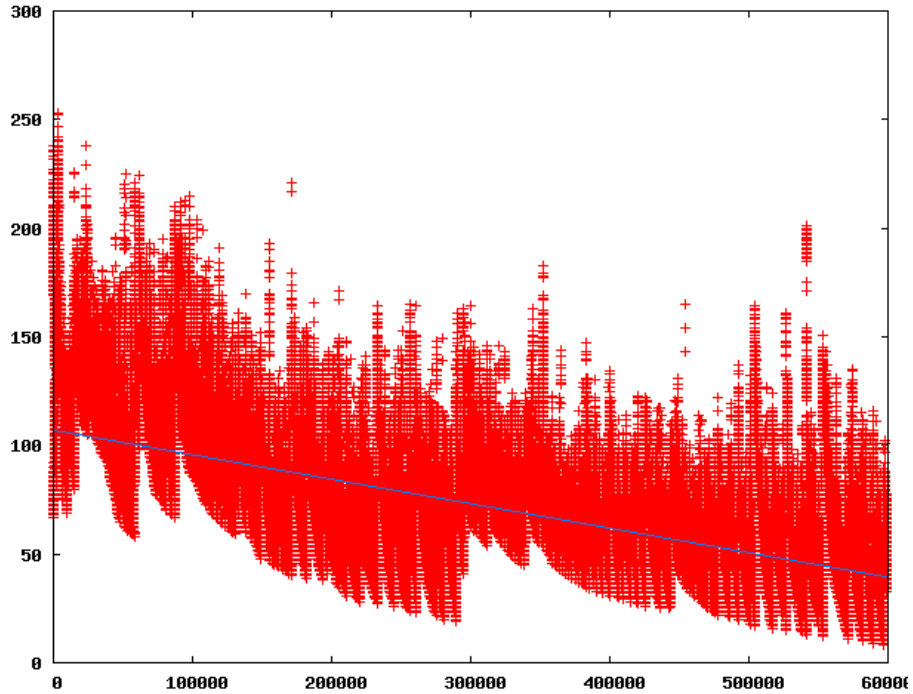
- Of course, we do not expect to feet curves
- We try to make observations of the behavior of a CDCL solver

AND...

Decreasing appear in a lot of problems

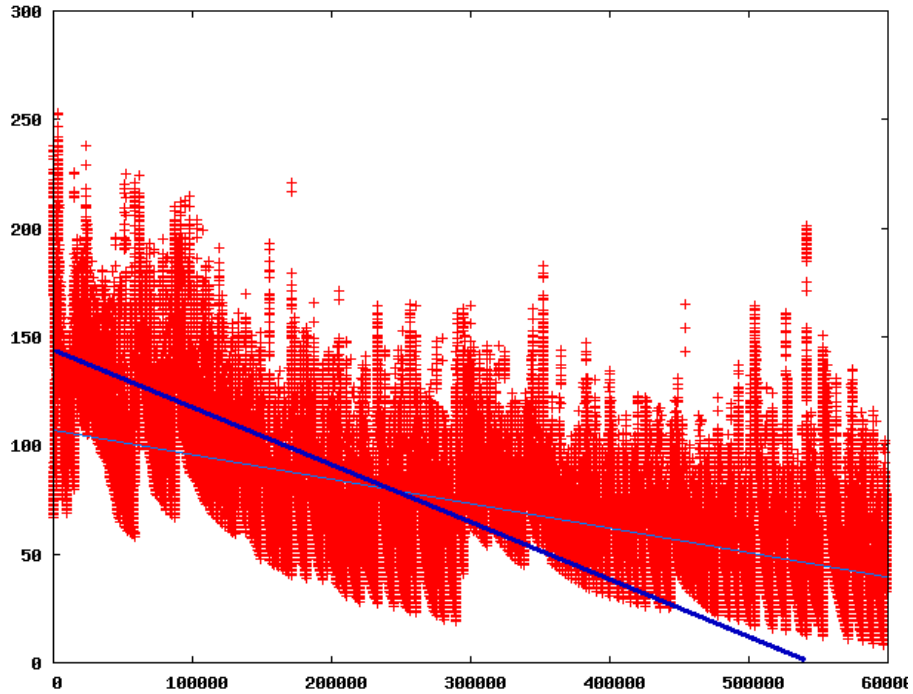
Series	#Benchs	% Decr.
een	8	62%
goldb	11	100%
grieu	7	71%
hoons	5	100%
ibm-2002	7	71%
ibm-2004	13	92%
manol-pipe	55	91%
miz	13	0%
schup	5	80%
simon	10	90%
vange	3	66%
velev	54	92%
all	199	83%

The goal



grieu-vmpc-s05-25 – SAT – 625 vars and 76,000 clauses

The goal



grieu-vmpc-s05-25 – SAT – 625 vars and 76,000 clauses

Intuitions

- A lot of dependencies between variables

  During search those variables will probably be propagated together inside blocks of propagations
- One needs to collapse independent blocks of propagated literals in order to reduce the decision level

The LBD score of a nogood is the number of different blocks of propagated literals

Intuitions

- A lot of dependencies between variables

  During search those variables will probably be propagated together inside blocks of propagations
- One needs to collapse independent blocks of propagated literals in order to reduce the decision level

The LBD score of a nogood is the number of different blocks of propagated literals

- LBD=2
  - Only one literal from the last decision level (the assertive one)
  - This literal will be **glued** to the other block
  - binary clauses have LBD equal to 2
- VSIDS + progress saving: this should occurs a lot!!!

Good clauses are GLUE clauses

Managing Learnt Clauses

Previous works

- Before GLUCOSE , managing learnt clauses was not considered as an important component of CDCL solvers
- Previous measures were not so accurate
- Clause database size followed a geometric progression
- Dependent of the size of the input formula: No cleaning are performed for huge formulas

Use the LBD measure

Agressive strategies

- Small LBD are good ones
- In case of equality, prefer clauses with recent activity (VSIDS like)
- No matter the size of the initial formula
- Remove half of learnt clauses every :
  - GLUCOSE 1.0 (2009): $20000 + 500 \times x$ conflicts
  - GLUCOSE 2.X (2011): $4000 + 300 \times x$ conflicts

A first step towards a dynamic management

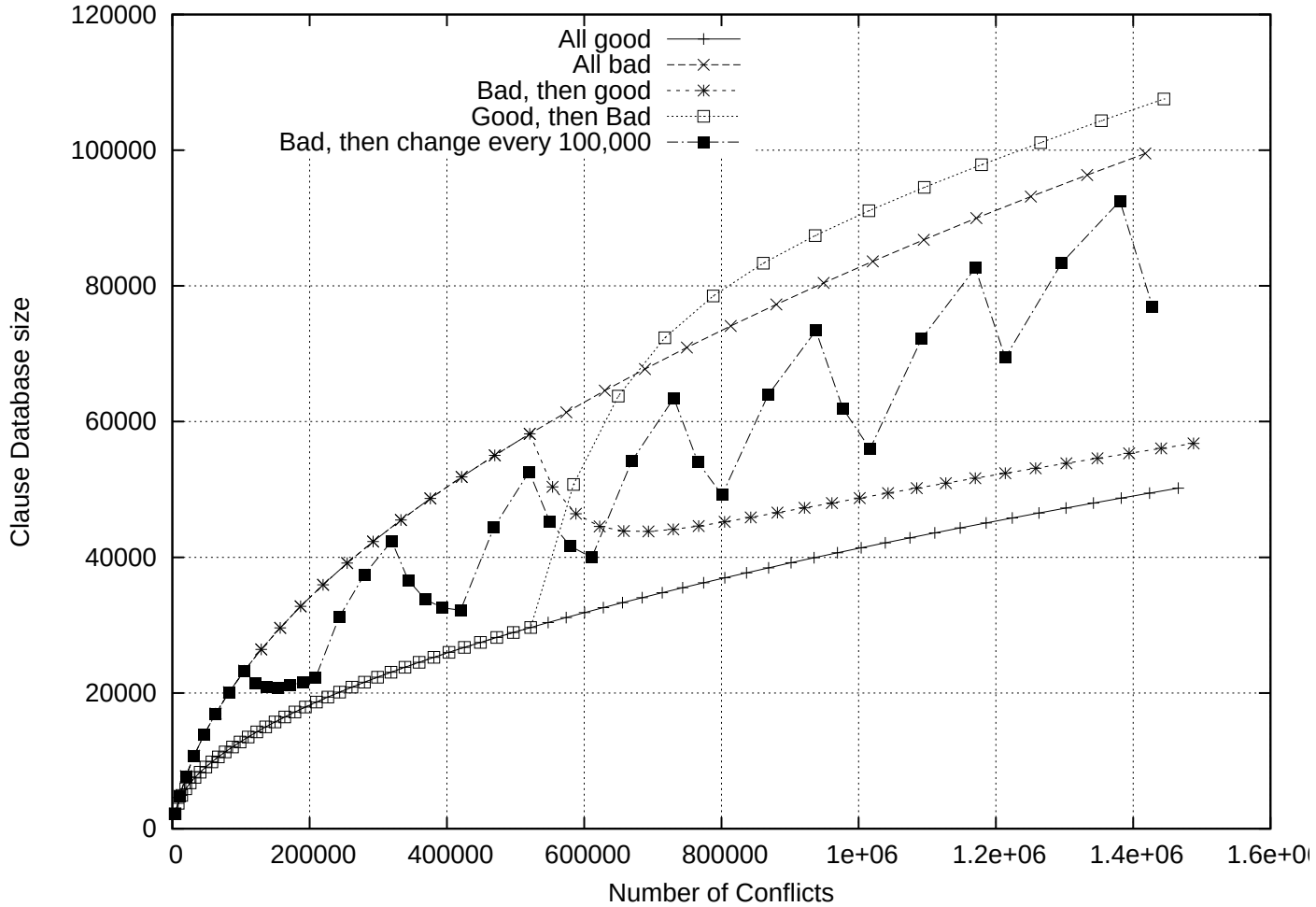
- Performances of GLUCOSE heavily depends on the quality of LBD
- A very good indicator on many instances
- However, it may not be discriminating enough
- A special case:
  - Half of clauses have a LBD less than 3 (we are going to remove potentially good clauses)
  - Too much good clauses
- We need to keep more of them
- We postpone the next cleaning by a constant of 1000

A first step towards a dynamic management

- Performances of GLUCOSE heavily depends on the quality of LBD
- A very good indicator on many instances
- However, it may not be discriminating enough
- A special case:
  - Half of clauses have a LBD less than 3 (we are going to remove potentially good clauses)
  - Too much good clauses
- We need to keep more of them
- We postpone the next cleaning by a constant of 1000

When performing cleaning??

Behavior


All good
All bad
Bad, then good
Good, then Bad
Bad, then change every 100,000
Clause Database size
Number of Conflicts
0
20000
40000
60000
80000
100000
120000
0
200000
400000
600000
800000
1e+06
1.2e+06
1.4e+06

Protect promising clauses

- Reminder: LBD is computed when the clause is learnt
- We computed it again it when a clause is used during BCP
- We change it, if it becomes smaller
- Such clauses seem interesting
- They are protected for one round

Restarts

Introduction

- Initially, restarts were introduced to prevent trashing
- Now, restarts must be seen as dynamic rearrangements of variables dependencies
- Restarts are more and more frequent
- GLUCOSE uses a dynamic restart strategy

Targetting UNSAT

- GLUCOSE aims to produce glue clauses
- If recent learnt clauses are bad (big LBD) a restart is performed
- We use
  - bounded queue (of size X) called `queueLBD`
  - the sum of all LBD clauses `sumLBD`

Targetting UNSAT

- GLUCOSE aims to produce glue clauses
- If recent learnt clauses are bad (big LBD) a restart is performed
- We use
  - bounded queue (of size X) called `queueLBD`
  - the sum of all LBD clauses `sumLBD`

```
// In case of conflict
compute learnt clause c;
sumLBD+=c.lbd();
queueLBD.push(c.lbd());
if(queueLBD.isFull() && queueLBD.avg()*K>sumLBD/nbConflicts) {
    queueLBD.clear();
    restart();
}
```

Targetting UNSAT

- GLUCOSE aims to produce glue clauses
- If recent learnt clauses are bad (big LBD) a restart is performed
- We use
  - bounded queue (of size X) called `queueLBD`
  - the sum of all LBD clauses `sumLBD`

```
// In case of conflict
compute learnt clause c;
sumLBD+=c.lbd();
queueLBD.push(c.lbd());
if(queueLBD.isFull() && queueLBD.avg()*K>sumLBD/nbConflicts) {
    queueLBD.clear();
    restart();
}
```

- Perform at least X conflicts before restarting

Targetting UNSAT

- GLUCOSE aims to produce glue clauses
- If recent learnt clauses are bad (big LBD) a restart is performed
- We use
  - bounded queue (of size X) called `queueLBD`
  - the sum of all LBD clauses `sumLBD`

```
// In case of conflict
compute learnt clause c;
sumLBD+=c.lbd();
queueLBD.push(c.lbd());
if(queueLBD.isFull() && queueLBD.avg()*K>sumLBD/nbConflicts) {
    queueLBD.clear();
    restart();
}
```

- Perform at least X conflicts before restarting
- Average over last X LBD become too big wrt total average

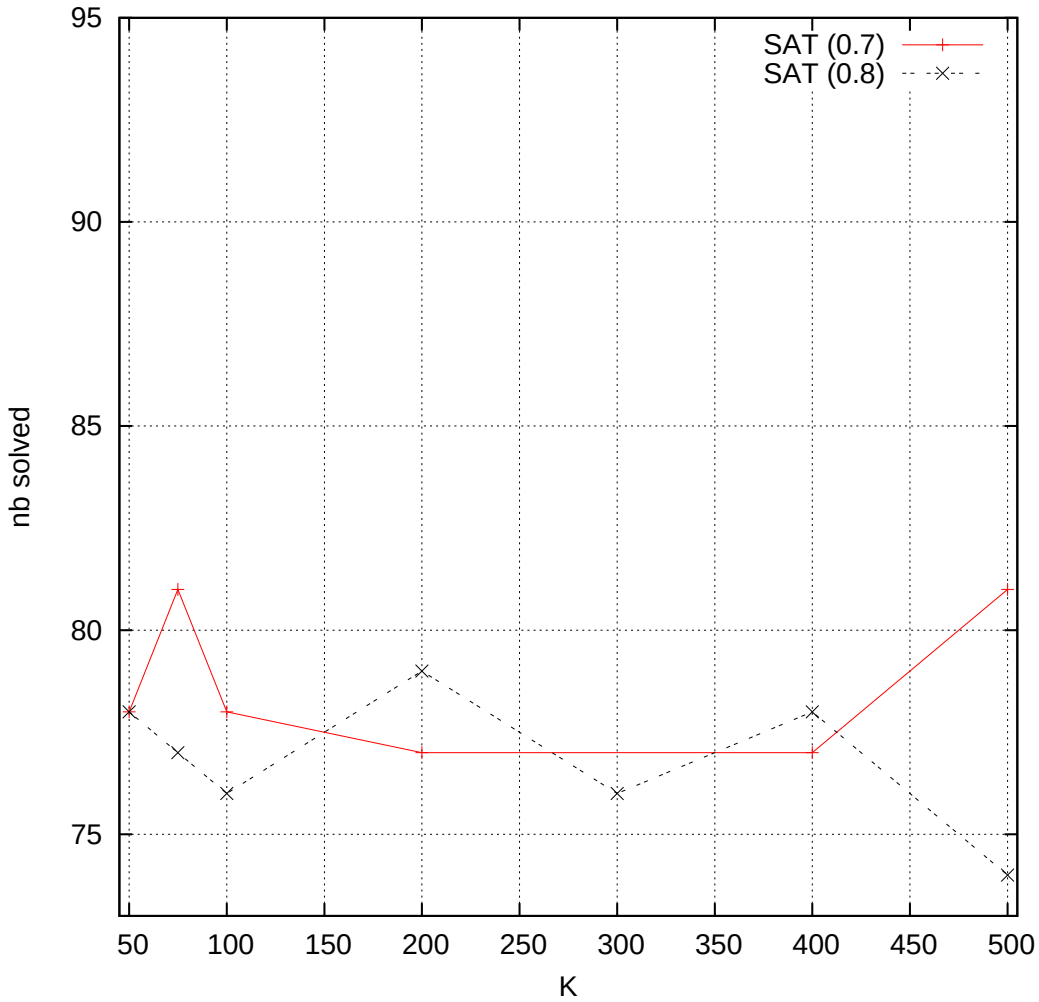
Targetting UNSAT

- GLUCOSE aims to produce glue clauses
- If recent learnt clauses are bad (big LBD) a restart is performed
- We use
  - bounded queue (of size X) called `queueLBD`
  - the sum of all LBD clauses `sumLBD`

```
// In case of conflict
compute learnt clause c;
sumLBD+=c.lbd();
queueLBD.push(c.lbd());
if(queueLBD.isFull() && queueLBD.avg()*K>sumLBD/nbConflicts) {
    queueLBD.clear();
    restart();
}
```

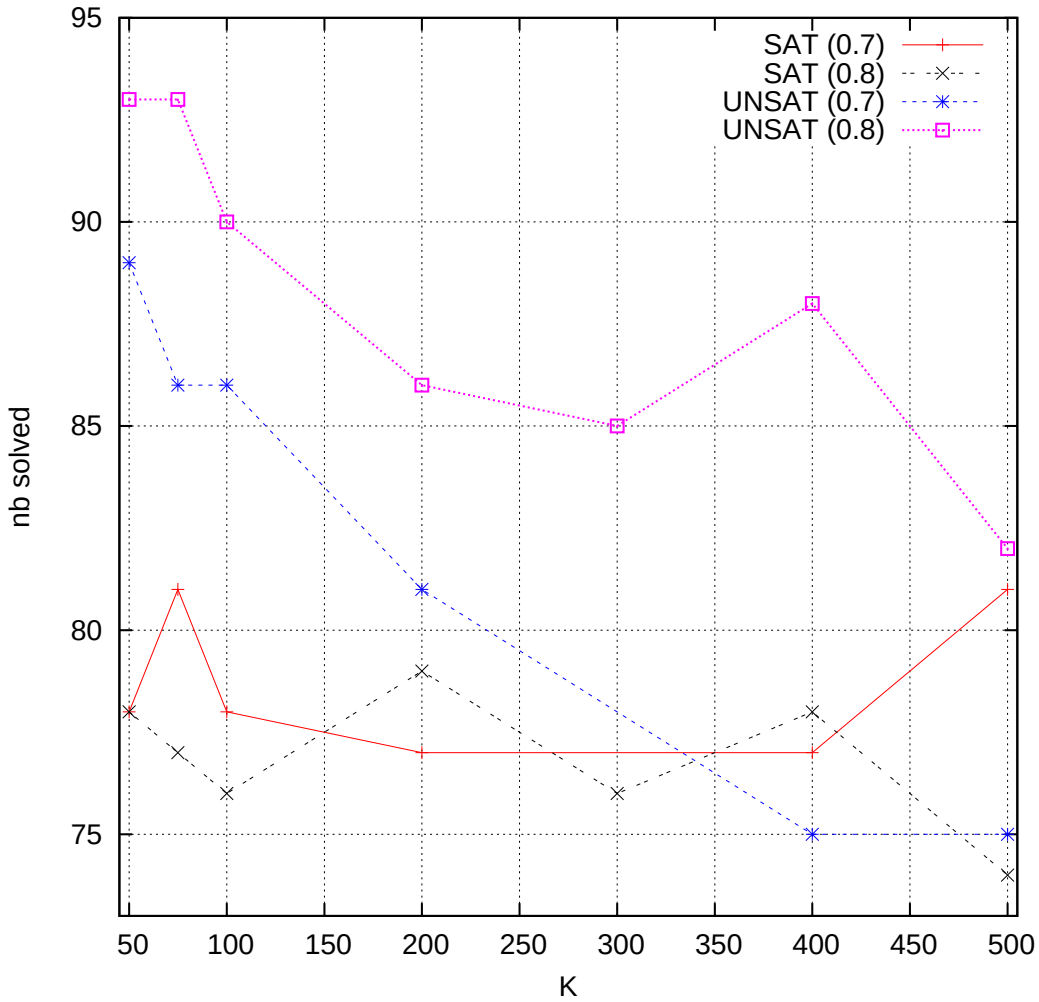
- GLUCOSE 1.0 and 2.0: X=100 and K=0.7
- GLUEMINISAT and GLUCOSE 2.1 : X=50 and K=0.8

Impact of different K and X



SAT 2011 Application benchmarks (limit 900 seconds)

Impact of different K and X



SAT 2011 Application benchmarks (limit 900 seconds)

Targeting SAT too (NEW in GLUCOSE 2.1)

- Frequent restarts seems not very good in case of SAT instances
- Some lessons of SAT 2011 competition – Second Phase, SAT instances
  - CONTRASAT: 1^{st} with 99 instances
  - GLUCOSE : 10^{th} with 94 instances
  - 6 of first ten solvers come from minisat hack (luby restarts)
  - 18 instances separate 1^{st} and 10^{th} in UNSAT

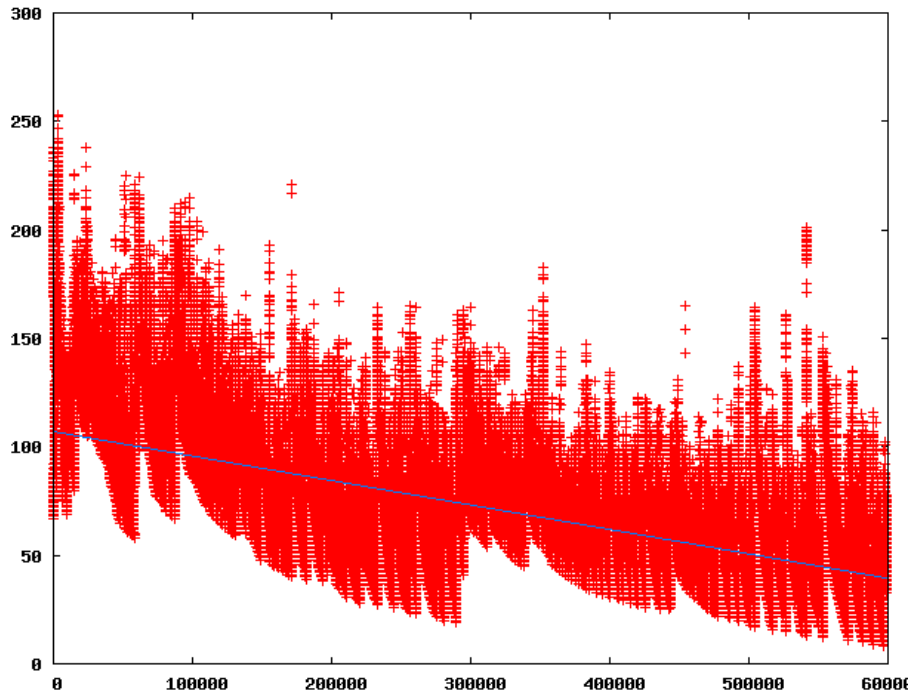
Targeting SAT too (NEW in GLUCOSE 2.1)

- Frequent restarts seems not very good in case of SAT instances
- Some lessons of SAT 2011 competition – Second Phase, SAT instances
  - CONTRASAT: 1^{st} with 99 instances
  - GLUCOSE : 10^{th} with 94 instances
  - 6 of first ten solvers come from minisat hack (luby restarts)
  - 18 instances separate 1^{st} and 10^{th} in UNSAT
- Agressive clauses deletion: some clauses may be bad for UNSAT but good for SAT
- Agressive restarts: some global assignments can be dropped!!

Delay restarts if total of assignments suddenly increase

Example

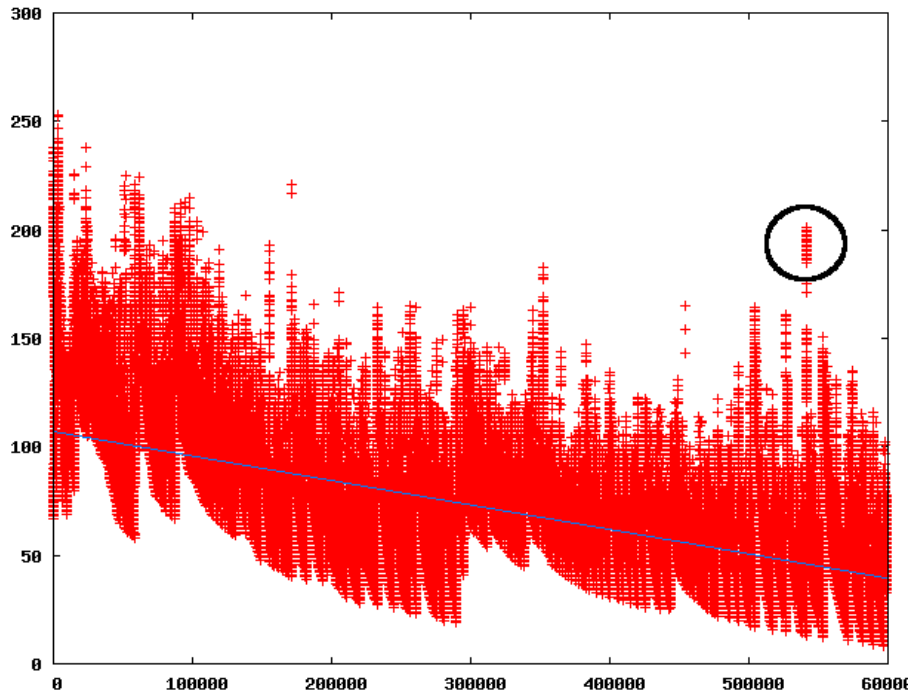
- Now, suppose dots represent trail stack assignment size



grieu-vmpc-s05-25 – SAT – 625 vars and 76,000 clauses

Example

- Now, suppose dots represent trail stack assignment size



grieu-vmpc-s05-25 – SAT – 625 vars and 76,000 clauses

- GLUCOSE is unlucky, a restart is performed!

Targeting SAT too (NEW in GLUCOSE 2.1)

- We use
  - `trail` the assignment stack
  - Bounded queue of the last Y trail size when reaching a conflict (`queueTrail`)

Targeting SAT too (NEW in GLUCOSE 2.1)

- We use
  - `trail` the assignment stack
  - Bounded queue of the last Y trail size when reaching a conflict (`queueTrail`)

```
// In case of conflict
 queueTrail.push(trail.size());
 if(queueLBD.isFull() && queueTrail.isFull() &&
     ➥trail.size()>T*queueTrail.avg()) {
      queueLBD.clear();
}

compute learnt clause c
 ...
```

Targeting SAT too (NEW in GLUCOSE 2.1)

- We use
  - `trail` the assignment stack
  - Bounded queue of the last Y trail size when reaching a conflict (`queueTrail`)

```
// In case of conflict
 queueTrail.push(trail.size());
 if(queueLBD.isFull() && queueTrail.isFull() &&
     ➥trail.size()>T*queueTrail.avg()) {
      queueLBD.clear();
}

compute learnt clause c
 ...
```

- The total number of assignments suddenly increase

Targeting SAT too (NEW in GLUCOSE 2.1)

- We use
  - `trail` the assignment stack
  - Bounded queue of the last Y trail size when reaching a conflict (`queueTrail`)

```
// In case of conflict
 queueTrail.push(trail.size());
 if(queueLBD.isFull() && queueTrail.isFull() &&
     ➥trail.size()>T*queueTrail.avg()) {
      queueLBD.clear();
}

compute learnt clause c
 ...
```

- The total number of assignments suddenly increase
- Postpone restart

Targeting SAT too (NEW in GLUCOSE 2.1)

- We use
  - `trail` the assignment stack
  - Bounded queue of the last Y trail size when reaching a conflict (`queueTrail`)

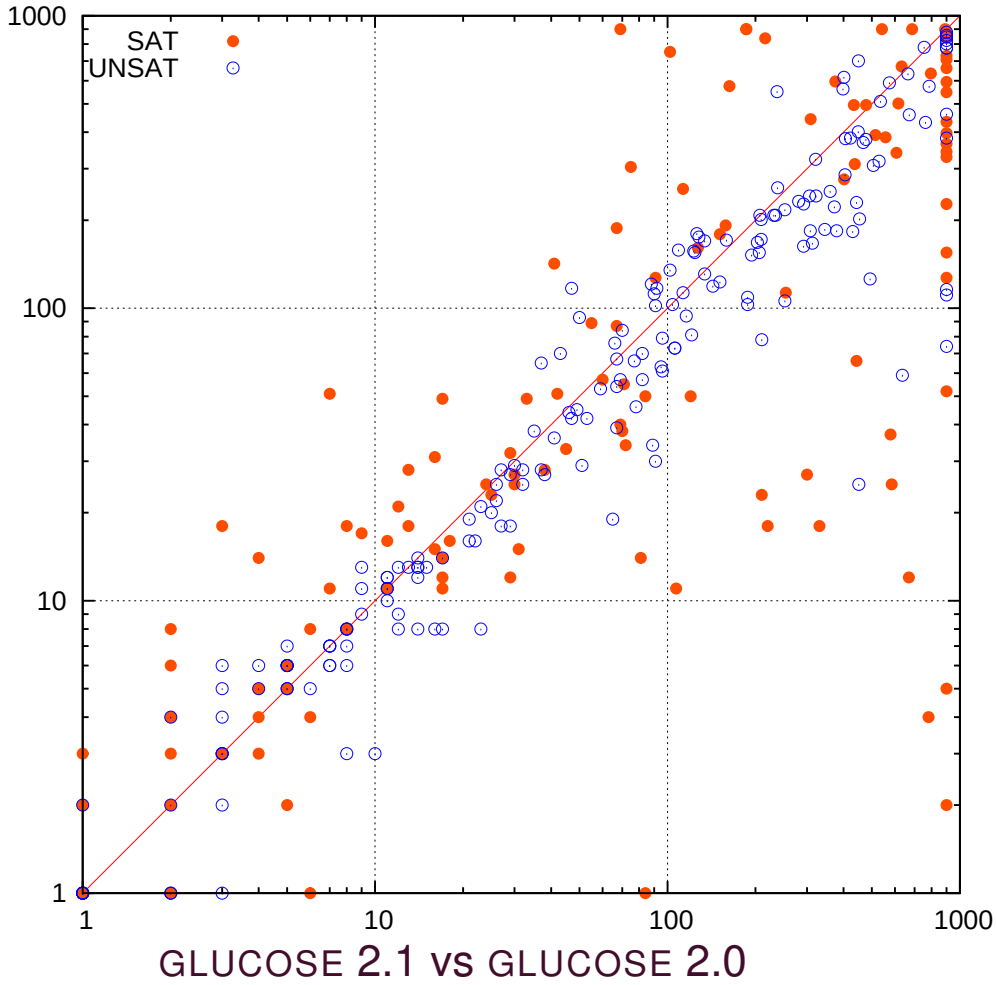
```
// In case of conflict
 queueTrail.push(trail.size());
 if(queueLBD.isFull() && queueTrail.isFull() &&
     ➥trail.size()>T*queueTrail.avg()) {
      queueLBD.clear();
}

compute learnt clause c
 ...
```

- The total number of assignments suddenly increase
- Postpone restart
- Y=5000 and T=1.4 appears to be good

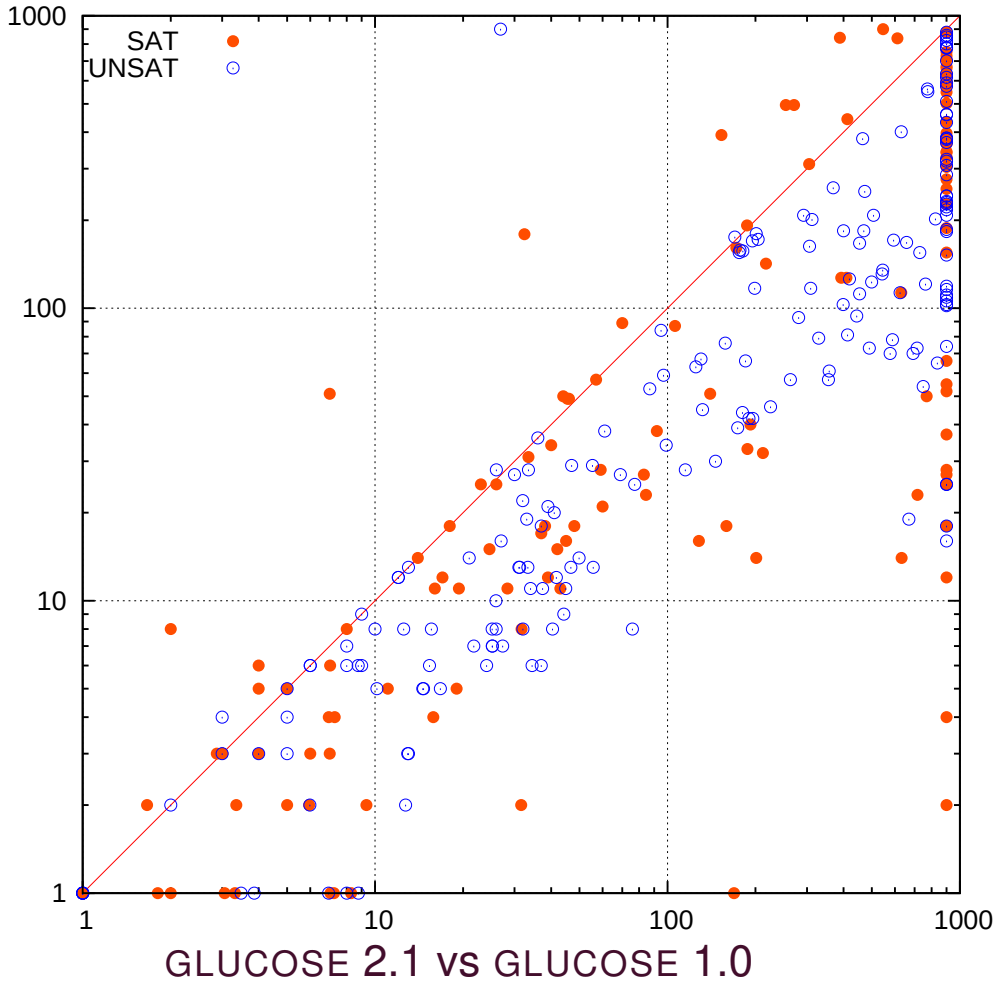
Conclusion

Evolution of GLUCOSE



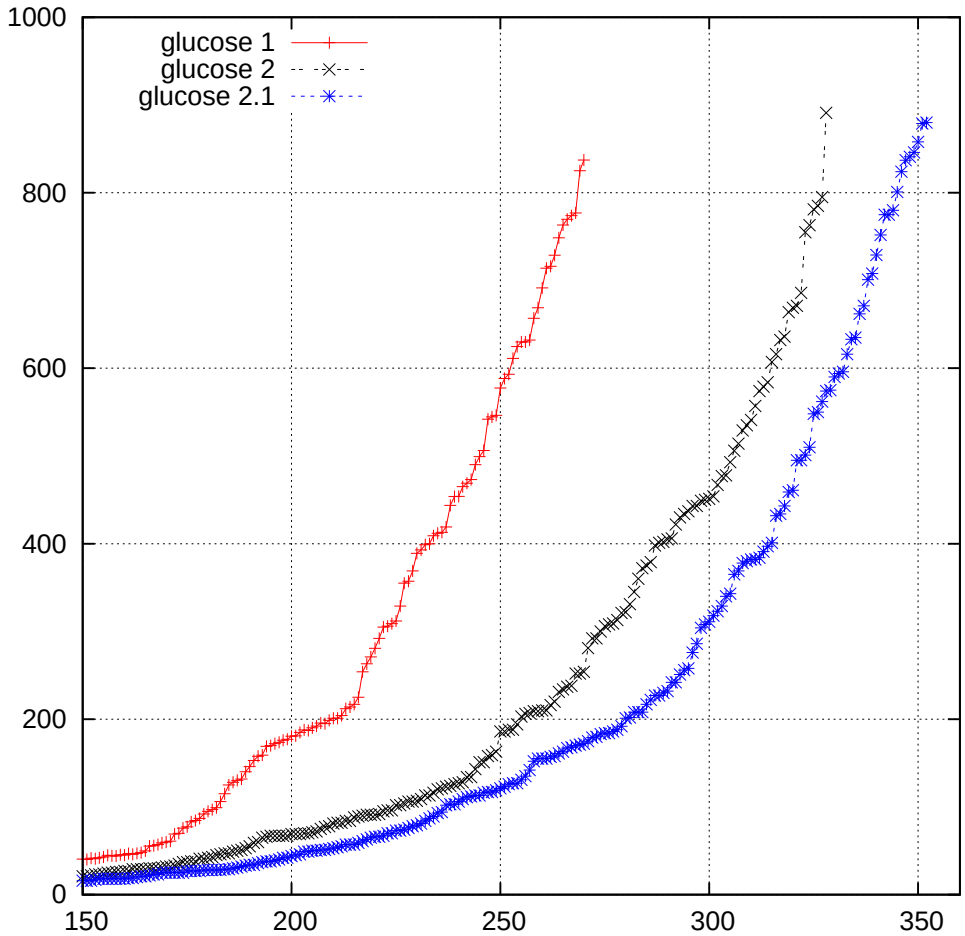
SAT 2011 application benchmarks (limit 900 seconds)

Evolution of GLUCOSE



SAT 2011 application benchmarks (limit 900 seconds)

Evolution of GLUCOSE



SAT 2009 + SAT 2011 application benchmarks (536, non redundant)

Evolution of GLUCOSE

Version	SAT	UNSAT	TOTAL
1.0	113	157	270
2.0	136	192	328
2.1	**148**	**204**	**352**

SAT 2009 + SAT 2011 application benchmarks (536, non redundant)

Conclusion

- The future of GLUCOSE . . . It is a secret :-)
  Current work with Daniel and Laurent

Conclusion

- The future of GLUCOSE . . . It is a secret :-)

  Current work with Daniel and Laurent

- A possible controversy
  - Are CDCL solvers still complete?
    - Very frequent restarts
    - Many deleted clauses (more than 93% for GLUCOSE (total for SAT 2011 Application benchmarks))
  - Are CDCL solvers closer to DPLL62 or local search??