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

(presented by Michalis Famelis)

February 3, 2012
Introduction

• Paper published in SAT’11.
• H.Katebi, student of K.Sakallah.
• K.Sakallah and J.Marques-Silva: authors of GRASP, that introduced Conflict-Driven Clause Learning.
• SAT: a hard problem, but with very important applications.

• SAT Solvers:
  • Major advancements in the last 15 years!
  • Contemporary solvers are based on DPLL.
  • Generally can scale up to several million vars/clauses.

• Most modern SAT solvers share a few important features*.

• Despite dramatic progress:
  • Unpredictable failures in practical problems.
  • Usually not clear which features make problems tractable.
  • Researchers unclear about features’ relative importance.
Contributions

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

DPLL algorithm:

- Branching.
- Unit propagation.
- Backtracking.

Modern SAT solvers extend this basic framework with algorithmic enhancements and heuristics.
Features in MiniSAT (1/2)

Conflict-driven clause learning

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

Random search restarts

- Complete search algorithms: heavy-tailed cost distribution.
- Unpredictable effect on run times.
- Solve by randomization. In SAT solvers: randomly backtrack to root (MiniSAT uses the Luby sequence).
Features in MiniSAT (2/2)

Two-Literal Watching

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

Conflict-based adaptive branching

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

MiniSAT can be configured for:

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

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

Two experiments:

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

Experimental approach:

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

- 10000 benchmarks from 12 application areas, since early 1990s.

- Criteria for selection:
  - Domains where SAT solving has been successful.
  - Benchmarks from SAT Competitions and Races.
  - A “reasonable” number of “easy” instances.
  - Weighting numbers in each area by relative success of application of SAT.

- Excluded random benchmarks: well studied elsewhere and real-word not as random.
# Subjects By Area

## Table 1. Benchmark families

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

(a) Number of variables

(b) Number of clauses

(c) Clause-to-variable ratio
Experiment 1: Results

![Graph showing CPU time vs instances for different algorithms.](image)
## Experiment 1: By Application Area

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

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


**Experiment 2: Results**

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

<table>
<thead>
<tr>
<th>Family</th>
<th>CDCL</th>
<th>rnd-freq</th>
<th></th>
<th></th>
<th></th>
<th>phase-saving</th>
<th>no-</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>50</td>
<td>75</td>
<td>100</td>
<td>init</td>
<td>none</td>
<td>basic</td>
</tr>
<tr>
<td>atpg</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>bioinf</td>
<td>150</td>
<td>133</td>
<td>107</td>
<td>72</td>
<td>46</td>
<td>150</td>
<td>139</td>
<td>149</td>
</tr>
<tr>
<td>config</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>500</td>
<td>50</td>
<td>500</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>crypto</td>
<td>237</td>
<td>67</td>
<td>63</td>
<td>49</td>
<td>35</td>
<td>228</td>
<td>214</td>
<td>223</td>
</tr>
<tr>
<td>equiv</td>
<td>231</td>
<td>221</td>
<td>216</td>
<td>181</td>
<td>162</td>
<td>231</td>
<td>220</td>
<td>222</td>
</tr>
<tr>
<td>fpga</td>
<td>470</td>
<td>456</td>
<td>453</td>
<td>444</td>
<td>421</td>
<td>470</td>
<td>471</td>
<td>468</td>
</tr>
<tr>
<td>h BMC</td>
<td>2333</td>
<td>2328</td>
<td>2322</td>
<td>2225</td>
<td>2057</td>
<td>2328</td>
<td>2328</td>
<td>2333</td>
</tr>
<tr>
<td>netcfg</td>
<td>87</td>
<td>76</td>
<td>75</td>
<td>60</td>
<td>72</td>
<td>80</td>
<td>76</td>
<td>77</td>
</tr>
<tr>
<td>plan</td>
<td>650</td>
<td>619</td>
<td>593</td>
<td>526</td>
<td>490</td>
<td>647</td>
<td>637</td>
<td>640</td>
</tr>
<tr>
<td>sverif</td>
<td>1006</td>
<td>915</td>
<td>858</td>
<td>762</td>
<td>302</td>
<td>1004</td>
<td>1003</td>
<td>996</td>
</tr>
<tr>
<td>termrw</td>
<td>420</td>
<td>416</td>
<td>407</td>
<td>378</td>
<td>291</td>
<td>420</td>
<td>416</td>
<td>417</td>
</tr>
<tr>
<td>Total:</td>
<td>9068</td>
<td>8720</td>
<td>8587</td>
<td>8194</td>
<td>7325</td>
<td>9042</td>
<td>8997</td>
<td>9016</td>
</tr>
</tbody>
</table>

(Bold: better performance than default.)
1 Introduction

2 Survey of CDCL Solver features

3 Experimental Setup

4 Experimental Results

5 Insights Gained

6 Conclusion
Experiment 1

- **Importance of major features:**
  
  \[ CL > VSIDS > 2WL > RST \]

- **Algorithmic optimizations vs heuristics:**
  - Enabling CL and 2WL consistently improves results.
  - The performance of VSIDS and RST was more variable.

- **Restarts are interesting:**
  - Relatively moderate performance gain.
  - Surprisingly restarts also improve things for UNSAT problems.
  - Not as reliable (in some cases \(\neg\)RS worked better).

- **Combination of all four features yields best performance.**
Experiment 2

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

2 Survey of CDCL Solver features

3 Experimental Setup

4 Experimental Results

5 Insights Gained

6 Conclusion
Paper’s Conclusions

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

• Methodology and results are definitely convincing.

• Research questions and relation to experiments are not clearly explicated.

• But where is the “Threats to Validity” section?

• E.g., why was MiniSAT specifically chosen?

• Very interesting also from the reverse perspective: what domains SAT is good for?