Algorithm Configuration for Portfolio-based Parallel SAT-Solving

Holger Hoos\textsuperscript{1}, Kevin Leyton-Brown\textsuperscript{1}, Torsten Schaub\textsuperscript{2}, Marius Schneider\textsuperscript{2}

\textsuperscript{1} Department of Computer Science
University of British Columbia
Vancouver, BC, Canada

\textsuperscript{2} Institute of Computer Science
University of Potsdam
Potsdam, Germany
The rise of main-stream multi-core computing

Parallel SAT Solvers

<table>
<thead>
<tr>
<th>2008 SAT RACE</th>
<th>2009 SAT Comp.</th>
<th>2010 SAT RACE</th>
<th>2011 SAT Comp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManySat</td>
<td>gNovelty+-T</td>
<td>antom</td>
<td>borg-sat</td>
</tr>
<tr>
<td>MiraXT</td>
<td>ManySAT</td>
<td>ManySAT</td>
<td>clasp</td>
</tr>
<tr>
<td>pMiniSat</td>
<td>satake</td>
<td>pMiniSAT</td>
<td>csls</td>
</tr>
<tr>
<td></td>
<td>ttsth</td>
<td>SArTagnan</td>
<td>CryptoMiniSAT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Plingeling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ppfolio</td>
</tr>
</tbody>
</table>

- (Slowly) increasing number of parallel SAT solvers
- Designing effective parallel solvers is challenging!
**Key idea:** Automatically generate parallel solver from highly parametric sequential design / sources
Parallel SAT Solving

▶ Search space splitting
e.g., clasp with guiding path

▶ Competitive parallel portfolio with clause sharing
e.g., Plingeling and CryptoMiniSat

▶ Competitive parallel portfolio w/o clause sharing
e.g., Plingeling and CryptoMiniSat
Ingredients for parallel SAT solver
based on competitive parallel portfolio

- Parametric solver $A$
- Configuration space $C$
- Instance set $I$
- Algorithm configurator $AC$

That’s all!
Recipe for parallel SAT solver
based on competitive parallel portfolio

1. Use algorithm configurator to produce multiple configurations of given solver that work well together
2. Run configurations in parallel until one solves given instance

Fully automatic method!
Recipe: **GLOBAL**

for parallel SAT solver based on competitive parallel portfolio

- For $k$ portfolio components (≡ processors/threads), consider combined configuration space $C^k$ of $k$ copies of given parametric solver

- Use configurator $AC$ to find good joint configuration in $C^k$ (standard protocol for current configurators: pick best result from multiple independent runs)

- Configurations are assessed using (training) instance set $I$

**Challenge:** Large configuration spaces (exponential in $k$)
Recipe: Greedy
for parallel SAT solver based on competitive parallel portfolio

- Add portfolio components, one at a time, starting from single solver

- **Iteration 1**: Configure given solver $A$ using configurator $AC$ to achieve optimised performance of single-component portfolio $A^1$

- **Iteration $j = 2 \ldots k$**: Configure given solver $A$ using $AC$ to achieve optimised performance of extended portfolio $A^j := A^{j-1} || A$
  i.e., optimise improvement in $A^j$ over $A^{j-1}$

**Note:** Similar idea to many greedy constructive algorithms (including Hydra, Xu et al. 2010)
**Product:** parallel *Lingeling* (v.276)  
on *SAT Comp. Application* instances

<table>
<thead>
<tr>
<th></th>
<th>PAR10</th>
<th>Overall Speedup vs Configured-SP</th>
<th>Avg. Speedup vs Configured-SP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default-SP</strong></td>
<td>3747</td>
<td>0.93</td>
<td>1.44</td>
</tr>
<tr>
<td><strong>Configured-SP</strong></td>
<td>3499</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td><strong>Plingeling</strong></td>
<td>3066</td>
<td>1.14</td>
<td>7.39</td>
</tr>
<tr>
<td><strong>Global-MP4</strong></td>
<td>2734</td>
<td>1.27</td>
<td>10.47</td>
</tr>
<tr>
<td><strong>Greedy-MP4</strong></td>
<td>1341</td>
<td>2.61</td>
<td>3.52</td>
</tr>
</tbody>
</table>
Product: parallel Lingeling (v.276) on SAT Comp. Application instances
**Product:** parallel *clasp* (v.2.0.2)

on *SAT Comp. Application* instances

<table>
<thead>
<tr>
<th></th>
<th>PAR10</th>
<th>Overall Speedup vs Configured-SP</th>
<th>Avg. Speedup vs Configured-SP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default-SP</strong></td>
<td>7560</td>
<td>0.82</td>
<td>4.46</td>
</tr>
<tr>
<td><strong>Configured-SP</strong></td>
<td>6170</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td><strong>Default-MP4</strong></td>
<td>2324</td>
<td>2.65</td>
<td>7.58</td>
</tr>
<tr>
<td><strong>Global-MP4</strong></td>
<td>3604</td>
<td>1.71</td>
<td>6.36</td>
</tr>
<tr>
<td><strong>Greedy-MP4</strong></td>
<td>2277</td>
<td>2.71</td>
<td>9.47</td>
</tr>
</tbody>
</table>
Product: parallel clasp (v.2.0.2) on SAT Comp. Application instances
Conclusions

- Parallel SAT solving is gaining importance
- Designing high-performance parallel SAT solvers can be challenging
- Fully automatic method:
  1. Use algorithm configurator to produce multiple configurations of given solver that work well together
  2. Run configurations in parallel until one solves given instance
- At least as good as hand-crafted parallel SAT solvers

Hoos, Leyton-Brown, Schaub, Schneider: Algorithm Configuration for Portfolio-based Parallel SAT-Solving
Future Work

- Parallel portfolios with different solvers
- New configuration protocols / configurators
- Generic mechanisms for cooperation between portfolio members