Snappy
A Simple Neighborhood-based Algorithm Portfolio in PYTHON

Horst Samulowitz, Chandra Reddy, Ashish Sabharwal, Meinolf Sellman
IBM TJ Watson Research Center, New York
Need for an Easy-to-Use Portfolio “Tool”!

- Despite proven success, users of most portfolios are:
  - their own creators, or
  - creators of other portfolios: need comparison to publish!

- Not yet adopted by SAT/CSP/MIP communities at large
  - “portfolio builders” for some are not available; others require significant familiarity with packages such as MATLAB
  - could allow showcasing the benefit of heuristics that are great on some instances but not necessarily in overall average performance!

Goal: Create a Portfolio Solver that is:

- Easy to Use by non-portfolio-creators
  - Must run “out-of-the box” without the need for an offline “training phase”
  - Can possibly improve its performance through Online Learning
- Easy to Understand, Parse, and Extend
- Has Competitive Performance across a Variety of Domains
Snappy: A Simple Portfolio in Python

- **Readable**: one single Python script, using scipy and other standard libraries
- Based on k-Nearest Neighbor performance information
  - Various metrics: Euclidean, Canberra, Mahalanobis, Minkowski, ...
- Needs Prior Performance data, as usual
- **BUT No “Training” Phase**: solver selection based on an “easy” function of the performance of all algorithm on k neighbors of test instance:
  - E.g., “min min” aggregation function:
    \[
    \arg \min_{A \in \text{Algorithms}} \min_{k \in [k_{\text{min}}, k_{\text{max}}]} \text{PAR}(A, k \text{ nearest neighbors of the test instance})
    \]
- “execution” mode: run on a new test instance
- “analysis” mode: run on several train-test splits, produce insights through summaries, charts, etc.
  - Can be tuned and improved, if desired
Snappy: Out-of-the-Box Performance

- Competitive simultaneously on several benchmarks with a single default setting! (3S and SATzilla trained for each benchmark row)

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>#Alg</th>
<th>#Feat.</th>
<th>Timeout</th>
<th>3S</th>
<th>%</th>
<th>PAR-1</th>
<th>snappy</th>
<th>%</th>
<th>PAR-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAT-2011-splits</td>
<td>37</td>
<td>48</td>
<td>5000</td>
<td>91.23</td>
<td>772.8</td>
<td>94.52</td>
<td>512.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAT-2012-10fold-f1</td>
<td>72</td>
<td>48</td>
<td>2000</td>
<td>96.59</td>
<td>174.3</td>
<td>96.48</td>
<td>161.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAT-2012-comp-f1</td>
<td>72</td>
<td>48</td>
<td>2000</td>
<td>83.05</td>
<td>556.4</td>
<td>83.77</td>
<td>560.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAT-2012-10fold-f2</td>
<td>72</td>
<td>32</td>
<td>2000</td>
<td>97.23</td>
<td>146.1</td>
<td>96.17</td>
<td>167.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAT-2012-comp-f2</td>
<td>72</td>
<td>32</td>
<td>2000</td>
<td>85.42</td>
<td>499.1</td>
<td>85.42</td>
<td>526.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>#Alg</th>
<th>#Feat.</th>
<th>Timeout</th>
<th>SATzilla</th>
<th>%</th>
<th>PAR-1</th>
<th>snappy</th>
<th>%</th>
<th>PAR-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industrial</td>
<td>18</td>
<td>125</td>
<td>5000</td>
<td>75.3</td>
<td>1685</td>
<td>72.6</td>
<td>1789</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Crafted</td>
<td>15</td>
<td>125</td>
<td>5000</td>
<td>66.0</td>
<td>2096</td>
<td>63.3</td>
<td>2198</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>9</td>
<td>125</td>
<td>5000</td>
<td>80.8</td>
<td>1172</td>
<td>80.3</td>
<td>1221</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Based on results reported by Xu et al [SAT-2012]
Last years tool

- **SatX10**: Plug&Play Parallel SAT Solver
  - Supports information sharing, configurable communication patterns
  - Framework available since last year
  - With Competition: Version available that has several solvers integrated already (e.g., Glucose, CircMinisat, MiniSat, etc.)
  - Should run on 1 to 10,000 cores