Configuring Random CSP Generators to Favor a Particular Consistency Algorithm

Daniel J. Geschwender, Robert J. Woodward, Berthe Y. Choueiry
Department of Computer Science & Engineering • University of Nebraska-Lincoln

Constraint Satisfaction Problem:
- Used to model constrained combinatorial problems
- Important real-world applications: hardware & software verification, scheduling, resource allocation, etc.

A CSP is defined as follows:
- A set of variables \((A, B, C)\)
- Their domains \(D_A = \{1,2,3\}, D_B = \{1,2,3,4\}, D_C = \{0,1\}\)
- A set of constraints:
  - \((A \geq B, B \neq 1, A+C < 3)\)

RBGenerator:
- Generates hard satisfiable CSP instances at the phase transition
- \(k\): arity of the constraints
- \(n\): number of variables
- \(a\): domain size \(d^n\)
- \(r\): number of constraints \(m = m(n)\)
- \(\delta\): distance from phase transition, \(\delta = \delta + 8/1000\)
- \(\text{forced}\): forced satisfiable?
- \(\text{merged}\): merge similar scopes?

SMAC: Sequential Model-based Algorithm Configuration
- SMAC tunes the parameter configuration of RBGenerator
- RBGenerator creates CSP to run on PerTuple and AllSol
- Move toward parameters which favor one algorithm over the other
- Algorithm time limit of 20 minutes
- Is a consistency property
- Compare other consistency algorithms
- Guarantees that every tuple allowed by a constraint must participate in some solution

Experiment Results:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>(k)</th>
<th>(n)</th>
<th>(a)</th>
<th>(r)</th>
<th>(\delta)</th>
<th>(\text{forced})</th>
<th>(\text{merged})</th>
<th>Speedup</th>
<th>Cov.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllSol</td>
<td>1</td>
<td>4</td>
<td>19</td>
<td>0.75</td>
<td>9.91</td>
<td>578</td>
<td>y</td>
<td>y</td>
<td>101.36</td>
</tr>
<tr>
<td>2</td>
<td>18</td>
<td>0.20</td>
<td>5.88</td>
<td>92.6</td>
<td>4.96</td>
<td>y</td>
<td>4</td>
<td>6.0%</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>17</td>
<td>0.66</td>
<td>18.9</td>
<td>759</td>
<td>75.73</td>
<td>8.10%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>1.74</td>
<td>8.28</td>
<td>309</td>
<td>348.43</td>
<td>4.88%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>1.45</td>
<td>4.94</td>
<td>54</td>
<td>11.22</td>
<td>32.69%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>0.78</td>
<td>1.52</td>
<td>100</td>
<td>9.63</td>
<td>27.00%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>1.75</td>
<td>8.12</td>
<td>302</td>
<td>289.63</td>
<td>27.61%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>1.70</td>
<td>3.01</td>
<td>175</td>
<td>13.72</td>
<td>77.29%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>19</td>
<td>0.72</td>
<td>7.70</td>
<td>825</td>
<td>80.70</td>
<td>35.68%</td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>14</td>
<td>0.86</td>
<td>9.95</td>
<td>446</td>
<td>115.01</td>
<td>17.31%</td>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>

Sequential Model-based Algorithm Configuration:
- SMAC tunes the parameter configuration of RBGenerator
- RBGenerator creates CSP to run on PerTuple and AllSol
- Compare runtimes and update SMAC response model
- Move toward parameters which favor one algorithm over the other

Experiments:
- 4 tests run, testing two factors:
  - \(\text{forced}\) : Configuring to favor PerTuple and AllSol
  - With adjustable and fixed problem size parameters
- Each test run over 10 configuration seeds
- Configuration run for 4 days
- Algorithm time limit of 20 minutes

Conclusion:
- Configured PerTuple 100x faster, AllSol 100x faster
- PerTuple configuration: less constraints, lower constraint tightness
- AllSol configuration: more constraints, higher constraint tightness
- Adjustable problem size only offers marginally better configuration

Future Work:
- Compare other consistency algorithms
- Use more parameterized CSP generator
- Apply results found to algorithm selection

Date April 13, 2014