Characterizing Performance of Consistency Algorithms by Algorithm Configuration of Random CSP Generators

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

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

A CSP is defined as follows:
- Given:
  - 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: \( \{A \geq B, B=2, A+C<3\} \)

- Question:
  - Determine consistency
  - Count number of solutions
  - Find minimal network
  - Minimize number of broken constraints

Minimal Network:
- Is a consistency property
- Guarantees that every tuple allowed by a constraint must participate in some solution to the CSP (i.e., the constraints are as minimal as possible)

\[ \forall \text{ tuple } \]
\[ \forall m-1 \text{ relations} \]

Two Algorithms for Enforcing Minimality:
- AllSol: better when there are many ‘almost’ solutions
  - One search explores the entire search space
  - Finds all solutions without storing them, keeps tuples that appear in at least one solution
- PERTUPLE: better when many solutions are available
  - For each tuple, finds one solution where it appears
  - Many searches that stop after the first solution

RBGenerator:
- Generates hard satisfiable CSP instances at the phase transition
- \( k \): arity of the constraints
- \( n \): number of variables
- \( \alpha \): domain size \( d^{=\alpha} \)
- \( r \): # constraints \( m=\tau(n) \)
- \( \delta \): distance from phase transition, \( \rho_r \approx 1/1000 \)
- forced: forced satisfiable?
- merged: merge similar scopes?

Sequential Model-based Algorithm Configuration:
- SMAC: Sequential Model-based Algorithm Configuration
- User-Defined Parameter Space
- Parameter Space Response Model
- Input parameters
- RBGenerator
- Random CSP instance
- AllSol
- AllSol time
- PerTuple time
- Break

Final Parameter Configurations:

<table>
<thead>
<tr>
<th>seed</th>
<th>( k )</th>
<th>( n )</th>
<th>( \alpha )</th>
<th>( r )</th>
<th>( \delta )</th>
<th>forced</th>
<th>speedup</th>
<th>Cov</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>4</td>
<td>2</td>
<td>1.00</td>
<td>0.74</td>
<td>0.481</td>
<td>n</td>
<td>n</td>
<td>1301</td>
</tr>
<tr>
<td>19</td>
<td>15</td>
<td>16</td>
<td>1.00</td>
<td>0.74</td>
<td>0.481</td>
<td>n</td>
<td>n</td>
<td>1301</td>
</tr>
</tbody>
</table>

Effect of Parameters \( r \) and \( \delta \):

Adjustable Problem Size:

Fixed Problem Size:

Conclusions:
- Configured PERTUPLE 1000x faster, AllSol 100x faster
- PERTUPLE configuration: fewer 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 a CSP generator with more parameters
- Apply results found to algorithm selection

Date January 29, 2015