A Portfolio Approach for Enforcing Minimality in a Tree Decomposition

Daniel J. Geschwender, Robert J. Woodward, Berthe Y. Choueiry, Stephen D. Scott

Constraint Systems Laboratory • Department of Computer Science & Engineering • University of Nebraska-Lincoln

We advocate the use of an algorithm portfolio for enforcing minimality on the clusters of a tree decomposition during lookahead in a backtrack search for solving CSPs.

Minimal network: A global consistency property
- Minimal domains: Every value in a domain appears in a solution
- Minimal relations: Every tuple in a relation appears in a solution (i.e., the constraints are as tight as possible)

∀ m-1 relations

Two algorithms for enforcing minimality [Karakashian, PhD 2013]

ALLSOL

PERTUPLE

- Better when there are many ‘almost’ solutions
- One search explores the entire search space
- Finds all solutions without storing them, keeps tuples that appear at least one solution

- Better when many solutions are available
- For each tuple, finds one solution where it appears
- Many searches that stop after the first solution

Classifiers

• Trained on 9362 individual clusters taken from 175 benchmarks
• Instances labeled: ‘ALLSOL’, ‘PERTUPLE’, or ‘Neither’ (more than 10 minutes)
• Used 73 separate features including: #tuples in relations, constraint tightness, relational linkage, features of incidence graph
• Computed descriptive statistics including: mean, min, max, coefficient of variation, entropy
• Weighted instances according to the function:
  weight(i) = \( \frac{w(\text{allSol}(i), \text{perTuple}(i))}{20} \)

  \( w(a, p) = \left[ \log_{10} \left( \frac{a}{p} \right) \right] \cdot \left[ \log_{10} \left( |a - p| + 0.01 \right) \right] \)

- Used 10-fold cross validation
- The trained decision tree achieved 90.8% weighted accuracy

FILTERCLUS algorithm

Run GAC globally

Build LIST from clusters

While propagation

For cluster C in LIST

Classify C

Run consistency on C under time limit (1s)

Run GAC globally

Reverse LIST

Experiments

- Used 1055 instances from 42 benchmarks
- Intel Xeon E5-2650 v3 2.30GHz processors with 12 GB memory
- 2 hour timeout per instance, 1 second timeout per cluster
- Backtrack search, dynamic dom/deg ordering
- Compared seven strategies for real-full lookahead
  - GAC, ALLSOL, PERTUPLE: basic algorithms
  - ALLSOL, PERTUPLE: ALLSOL/PERTUPLE with timeout and GAC Interleave
  - RANDOM: timeout, GAC interleave, and random classifier
  - DEC TREE: timeout, GAC interleave, and trained decision tree classifier

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of Completed Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAC</td>
<td>550</td>
</tr>
<tr>
<td>ALLSOL</td>
<td>472</td>
</tr>
<tr>
<td>PERTUPLE</td>
<td>567</td>
</tr>
<tr>
<td>ALLSOL</td>
<td>514</td>
</tr>
<tr>
<td>PERTUPLE</td>
<td>633</td>
</tr>
<tr>
<td>RANDOM</td>
<td>643</td>
</tr>
<tr>
<td>DEC TREE</td>
<td>685</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average Runtime (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.471</td>
</tr>
<tr>
<td>3.075</td>
</tr>
<tr>
<td>2.081</td>
</tr>
<tr>
<td>2.789</td>
</tr>
<tr>
<td>1.622</td>
</tr>
<tr>
<td>1.427</td>
</tr>
<tr>
<td>1.121</td>
</tr>
</tbody>
</table>

Conclusions

- A portfolio at the cluster level and during search is not only feasible but also a winner
- Enforcing a timeout on cluster consistencies prevents getting stuck on one part of the problem

Future work

- Use the classifier to dynamically set the timeout based on the anticipated filtering