Selecting the Appropriate Consistency Algorithm for CSPs Using Machine Learning Classifiers

Daniel J. Geschwender, Shant Karakashian, Robert J. Woodward, Berthe Y. Choueiry, Stephen D. Scott
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:
  1. $A\geq B, B\geq A$, $A+C<3$
  2. $\sum(D_A^2)= (2k, 0)$
  3. $\log(D_A^2)= (0, 0)$

Question:
- Find a solution
- Count number of solutions
- Find minimal network
- Minimize number of broken constraints

Practical Tractability:
- [Karakashian+ AAAI 2013]
  - Enforces minimality on each cluster of a tree decomposition
  - Bolsters propagation between clusters by adding constraints to separators
  - Solves many instances in a backtrack-free manner

Tree Decomposition:
- Used to break up a CSP into clusters arranged in a tree structure
- Each cluster is a subproblem that can be independently handled
- Filtering performed on a cluster propagates to neighboring clusters

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)

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

The Problem of AllSol vs. PerTuple:
- The performance of the two algorithms varies widely
- One algorithm may complete quickly while the other may not terminate
- The performance depends on size and difficulty of the CSP instance

Question: Can we use Machine Learning to classify the instance & predict the best algorithm?

Feature Set:
- Characteristics of the problem selected to differentiate the classes
  - $\log(\text{size(allLinks)})$
  - $\log(\text{size(permLinks)})$
  - $\log(\text{size(allPermLinks)})$
  - $\log(\text{size(allPermLinks)})$
  - $\log(\text{size(allPermLinks)})$
  - $\log(\text{size(allPermLinks)})$
  - $\log(\text{size(allPermLinks)})$
  - $\log(\text{size(allPermLinks)})$

Using Machine Learning:
- We used a decision tree classifier (J48 from Weka Machine Learning suite) to make our predictions
- Each instance is a single cluster from a tree decomposition
- The 12 features of the CSP are observed
- CPU time for AllSol & PerTuple is recorded for each instance
- Experimented with four sets of training conditions

Experiments:
1. All instances – Trained using all data collected
2. $\delta \geq 100ms$ – Removed all instances where the difference in time was less than 100ms
3. Weighted – All instances are given a weight equal to the difference in execution time of the two algorithms
4. Cost – A cost matrix is used in the training, which provides average misclassification costs for each class

Experiment Results:

<table>
<thead>
<tr>
<th>Strategy</th>
<th>F-measure</th>
<th>Time saved</th>
<th>Time lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>All instances</td>
<td>.727</td>
<td>99.87%</td>
<td>15,301,950</td>
</tr>
<tr>
<td>$\delta \geq 100ms$</td>
<td>.729</td>
<td>99.90%</td>
<td>15,306,510</td>
</tr>
<tr>
<td>Weighted</td>
<td>.743</td>
<td>99.96%</td>
<td>15,314,980</td>
</tr>
<tr>
<td>Cost</td>
<td>.557</td>
<td>99.57%</td>
<td>15,255,190</td>
</tr>
</tbody>
</table>

Our Classification:
3592 instances from 5 benchmarks

Larger Instance Space:
31878 instances from 119 benchmarks

Future Work:
- Use a larger & more diverse set of benchmarks
- Explore additional features and classifiers
- Consider additional consistency properties & propagation algorithms

Date July 7, 2013