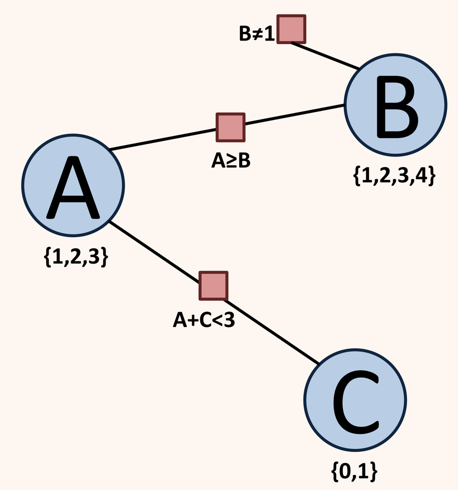
Selecting the Appropriate Consistency Algorithm for CSPs Using Machine Learning Classifiers

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.



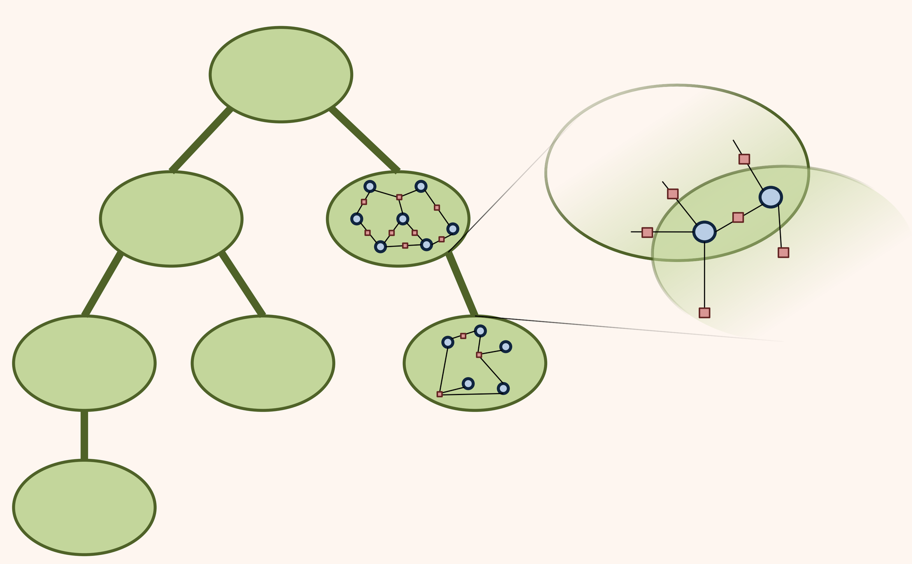
A CSP is defined as follows:

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≥B,B≠2,A+C<3}

Question
- Find a solution — NP-complete
- Count number of solutions — #P
- Find minimal network — NP-complete
- Minimize number of broken constraints — NP-hard

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:

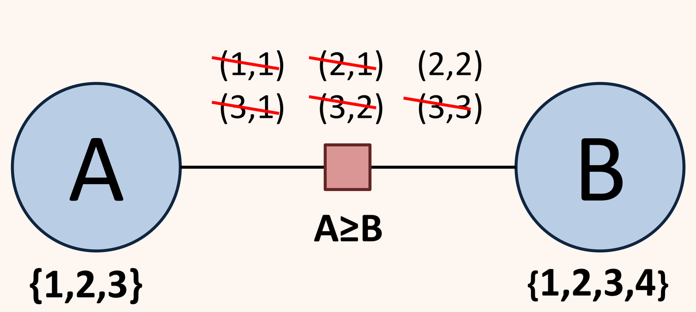
- Used to break up a CSP into clusters arranged in a tree structure
- Each cluster is a subproblem that can be independently handled
- Filtering performed on a cluster propagates to neighboring clusters

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 difficult of the CSP instance

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

Feature Set:

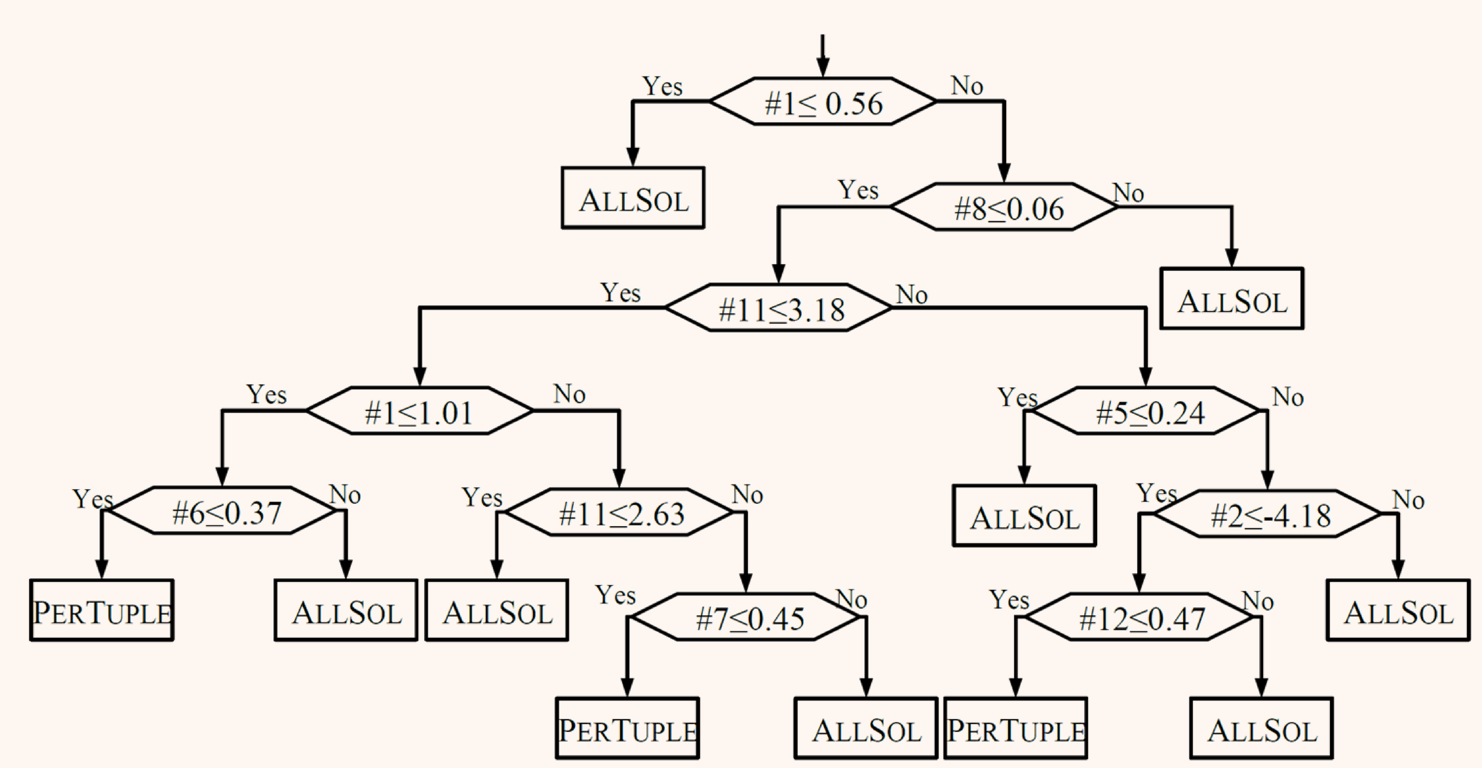
Characteristics of the problem selected to differentiate the classes

- *Kappa* – predicts if instance is near phase transition
- *relLinkage* – likelihood of a tuple at the overlap of relations to be in solution
- *tupPerVvp* – count of tuples containing a given variable value pair
- *relPerVar* – number of relations on a given variable

1. κ
2. $\log_2(avg(\texttt{relLinkage}))$
3. $\log_2(stDev(\texttt{relLinkage}))$
4. $stDev(\texttt{relLinkage})/avg(\texttt{relLinkage})$
5. $stDev(\texttt{tupPerVvp})/avg(\texttt{tupPerVvp})$
6. $avg(\texttt{tupPerVvpNorm})$
7. $stDev(\texttt{tupPerVvpNorm})$
8. $stDev(\texttt{tupPerVvpNormProd})$
9. $stDev(\texttt{tupPerVvpNormProd})/avg(\texttt{tupPerVvpNormProd})$
10. $avg(\texttt{relPerVar})$
11. $stDev(\texttt{relPerVar})$
12. $stDev(\texttt{relPerVar})/avg(\texttt{relPerVar})$

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:

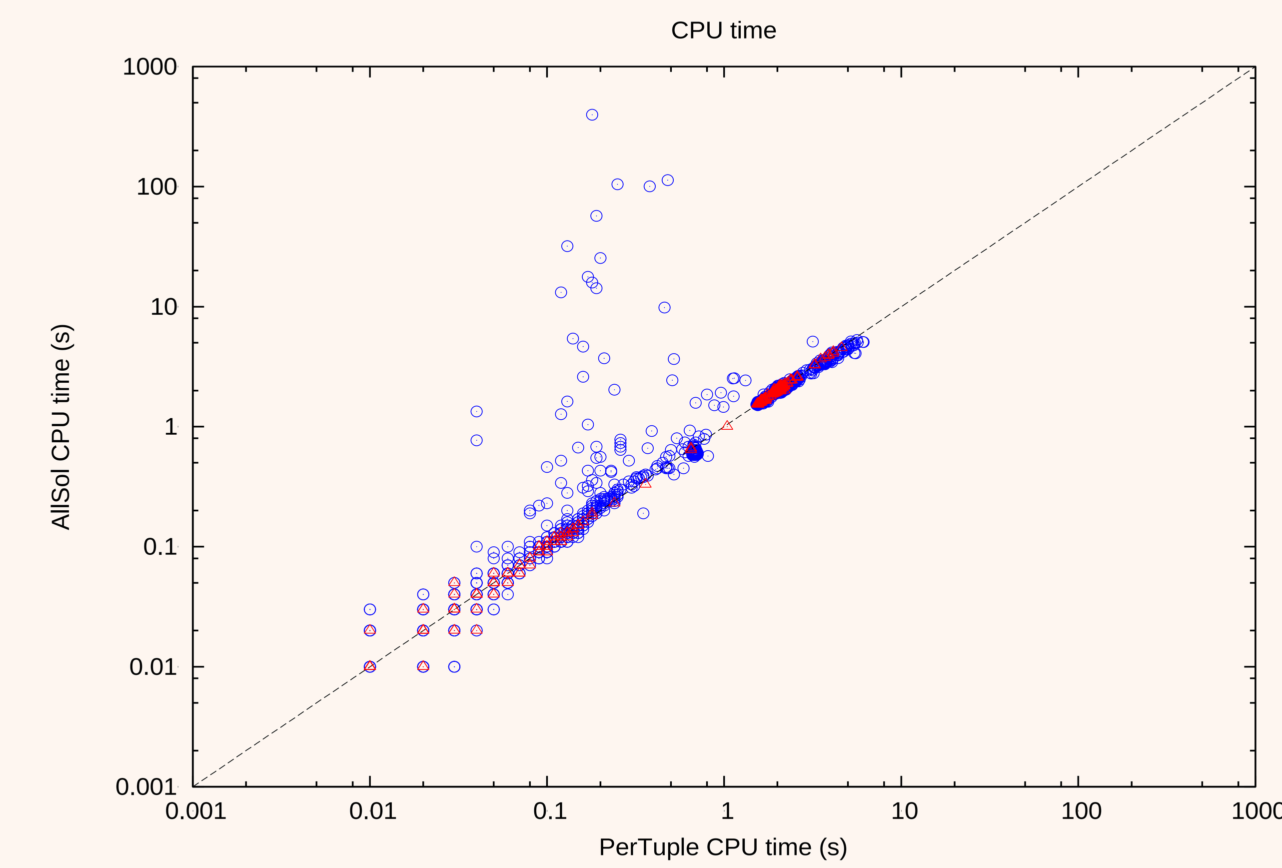
1. All instances – Trained using all data collected
2. δt ≥ 100ms – Removed all instances where the difference in time was less than 100ms
3. Weighted – All instances are given a weight equal to the difference in execution time of the two algorithms
4. Cost – A cost matrix is used in the training, which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

Experiment Results:

Strategy	F-measure	Time saved		Time lost
		%	ms	ms
All instances	.727	99.87%	15,301,950	19,350
$\delta_t \geq$ 100ms	.729	99.90%	15,306,510	14,790
Weighted	**.743**	**99.96%**	**15,314,980**	**6,320**
Cost	.557	99.57%	15,255,190	66,110

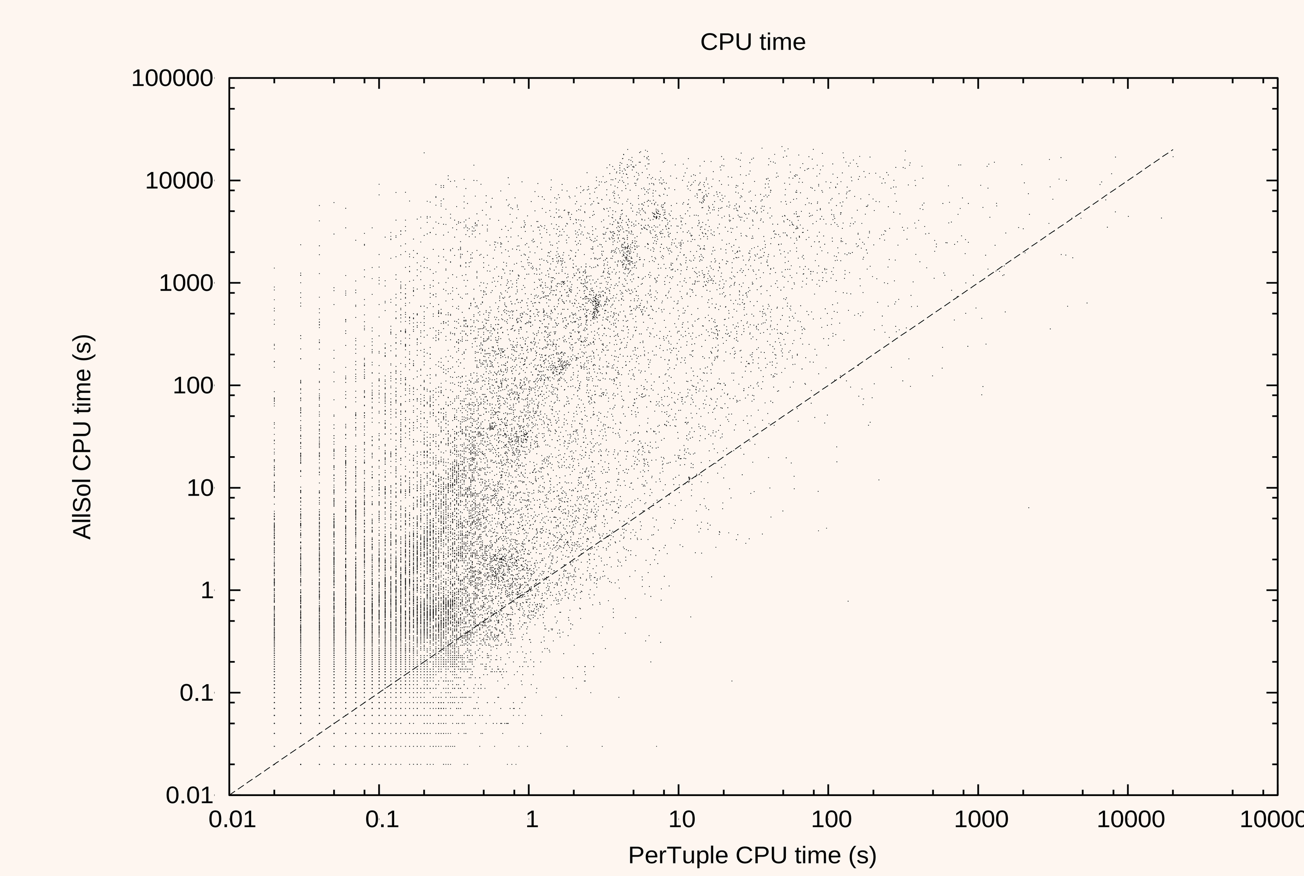
Our Classification:

3592 instances from 5 benchmarks



Larger Instance Space:

318158 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

University of Nebraska Lincoln — Constraint Systems Laboratory — NSF

Supported by NSF Grant No. RI-111795 and Undergraduate Creative Activities and Research Experiences Program (UCARE) of the University of Nebraska-Lincoln. Experiments conducted at UNL's Holland Computing Center.

Date July 7, 2013