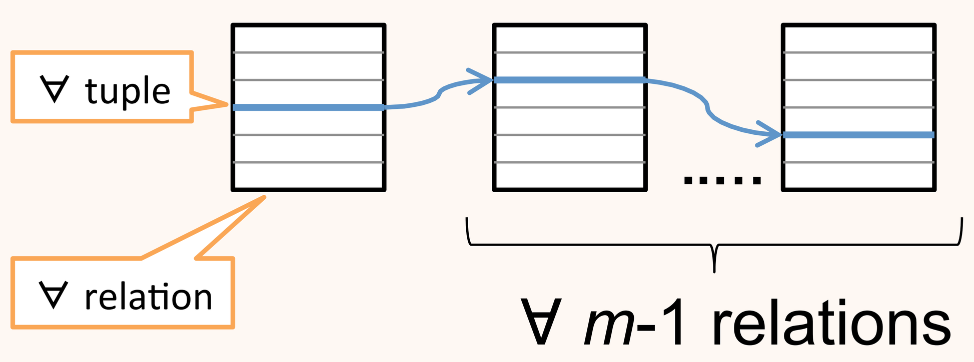
A Portfolio Approach for Enforcing Minimality in a Tree Decomposition

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

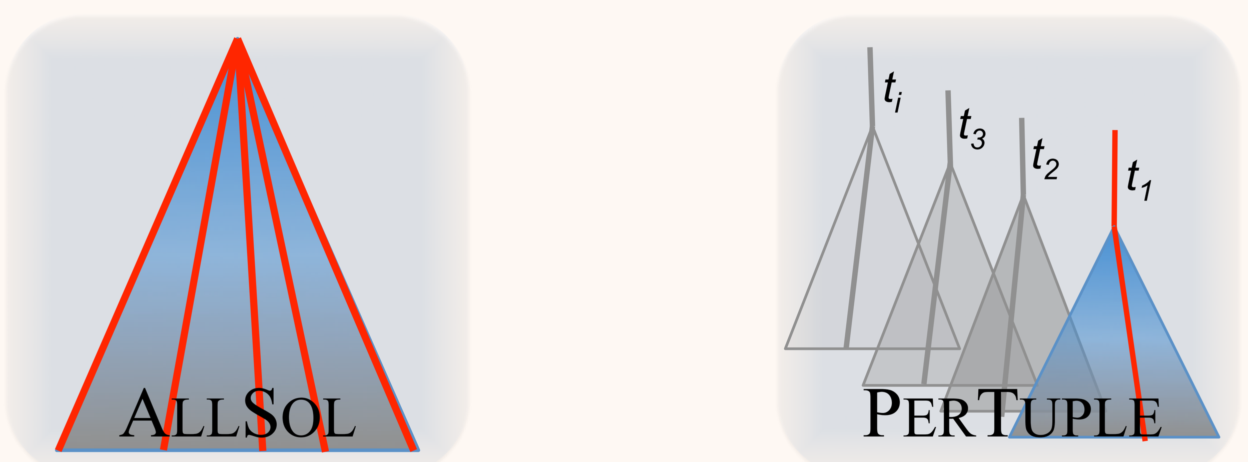
We advocate the use of an algorithm **portfolio** for enforcing minimality on the **clusters** of a tree decomposition during **lookahead** in a backtrack search for solving CSPs.

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)



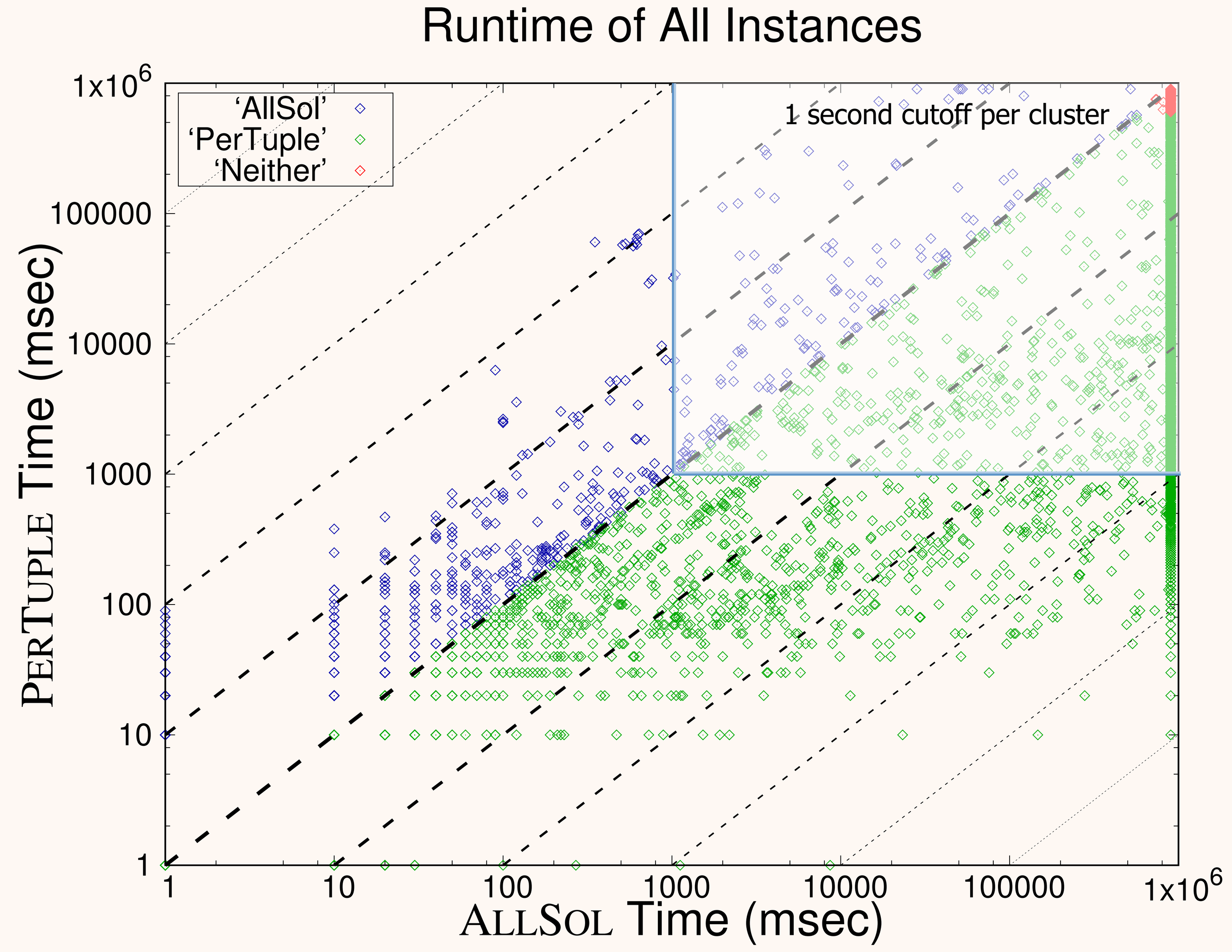
Two algorithms for enforcing minimality [Karakashian, PhD 2013]



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

Runtime of All Instances



Classifier training

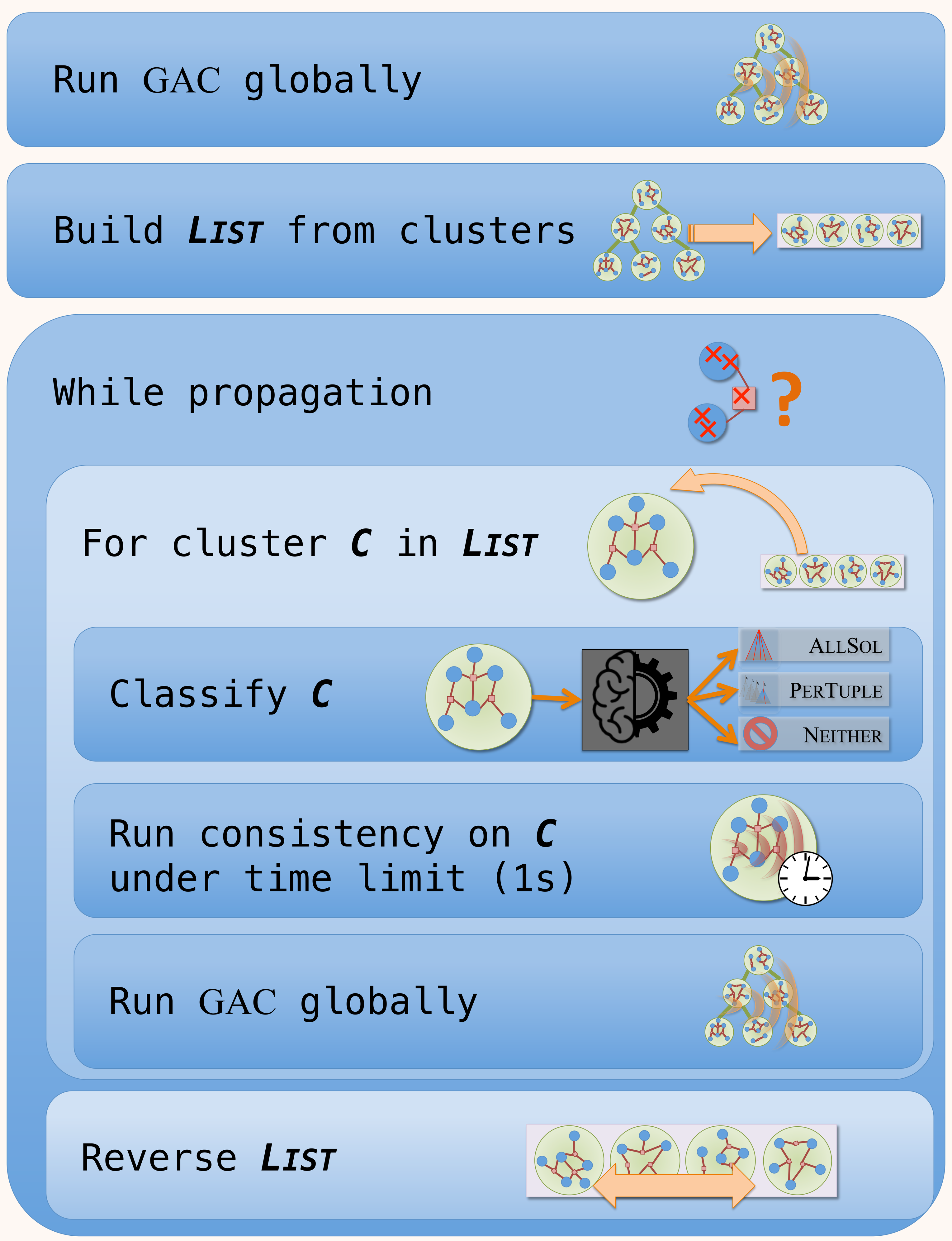
- Trained on 9362 individual clusters taken from 175 benchmarks
- Instances labeled: 'AllSol', 'PerTuple', or 'Neither' (more than 10 minutes)
- Used 73 separate features including: #tuples in relations, constraint tightness, relational linkage, features of incidence graph
- Computed descriptive statistics including: mean, min, max, coefficient of variation, entropy
- Weighted instances according to the function:

$$weight(i) = \begin{cases} w(allSol(i), perTuple(i)) & label(i) = \text{'}AllSol\text{'} \| \text{'}PerTuple\text{'} \\ 20 & label(i) = \text{'}Neither\text{'} \end{cases}$$

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

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

FilterClusters algorithm

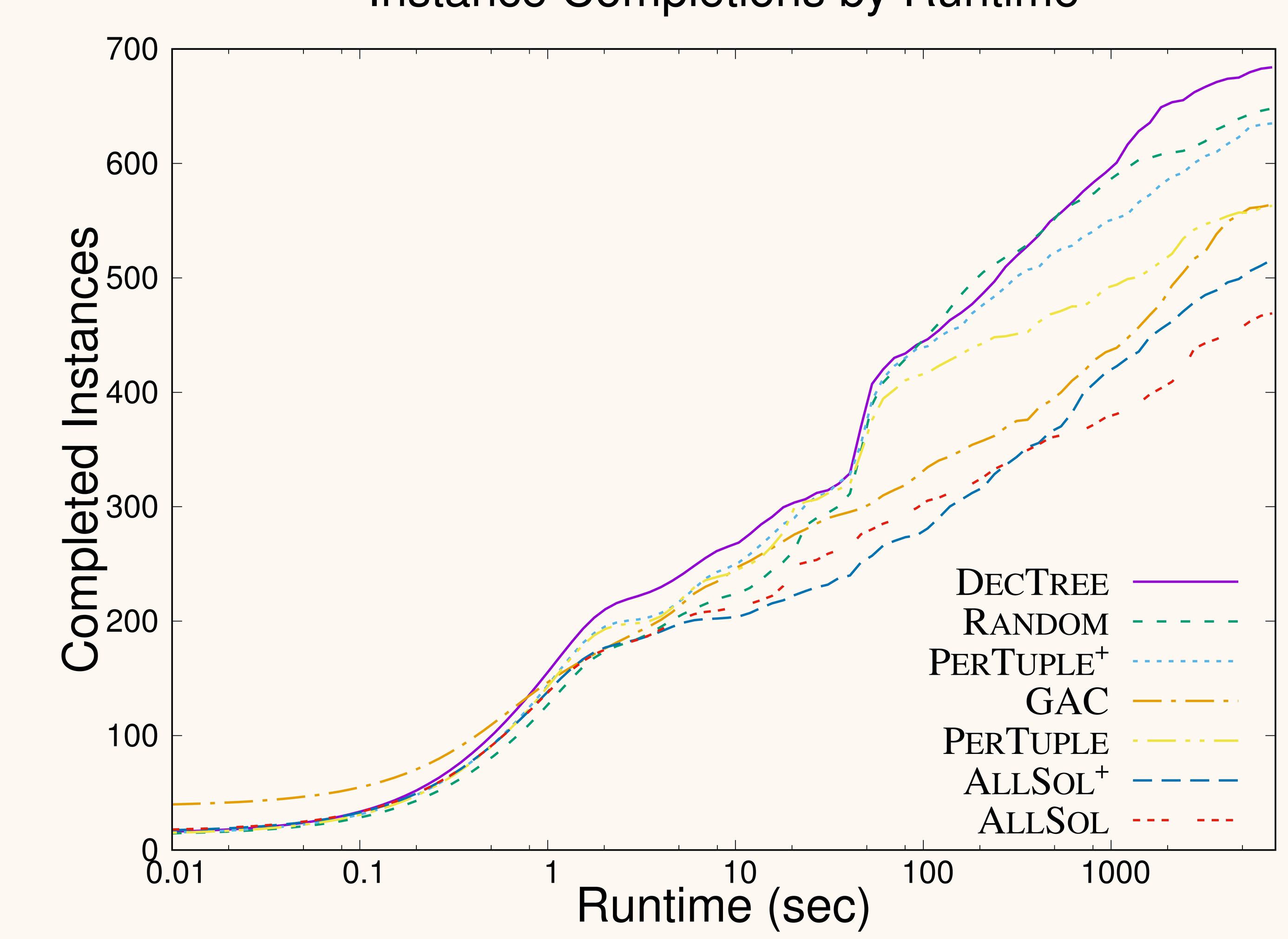


Experiments

- 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
  - DecTree: timeout, GAC interleave, and trained decision tree classifier

	GAC	AllSol	PerTuple	AllSol+	PerTuple+	Random	DecTree
Instances Completed	550	472	567	514	633	643	**685**
Average Time (s)	2,471	3,075	2,081	2,789	1,622	1,427	**1,121**

Instance Completions by Runtime



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

University of Nebraska Lincoln · Constraint Systems Laboratory · NSF

Experiments were conducted at the Holland Computing Center facility of the University of Nebraska. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 1041000 and NSF Grants No. RI-111795 and RI-1619344.

Date September 19, 2016