MDD-Based Propagation of Among Constraints

Samid Hoda Willem-Jan van Hoeve J.N. Hooker

Tepper School of Business
Carnegie Mellon University

EURO 2010

Outline

- Motivation
  - constraint programming
  - propagation based on MDDs
- MDD-propagation of *among* constraints
  - edge filtering
  - node refinement
- Experimental results
- Conclusion

Motivation

Constraint Programming applies

- systematic search and
- inference techniques

to solve combinatorial problems

Inference mainly takes place through:

- Filtering provably inconsistent values from variable domains
- Propagating the updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

alldifferent(x_1, x_2, x_3, x_4)

$x_1 \in \{2\}$, $x_2 \in \{1\}$, $x_3 \in \{3\}$, $x_4 \in \{0\}$ }

Drawback of domain propagation

Observations:

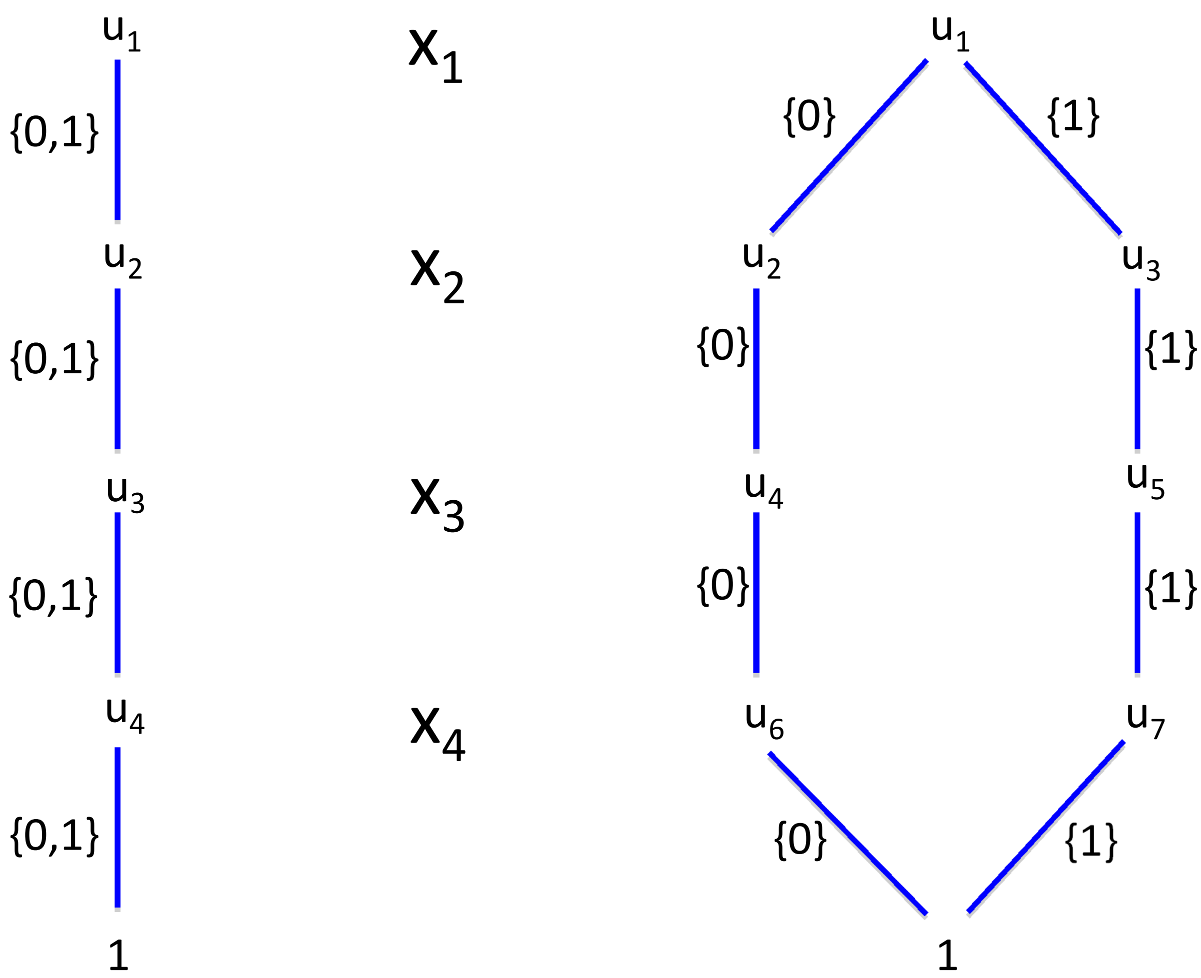
- Communication between constraints only via variable domains
- Information can only be expressed as a domain change
- Other (structural) information that may be learned by a constraint is lost: it must be projected onto variable domains
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

This drawback can be addressed by communicating more expressive information

- Using *multi-valued decision diagrams* (MDDs) [Andersen et al. 2007]
- Explicit representation of more refined potential solution space

Illustrative Example

AllEqual(x_1, x_2, x_3, x_4), all x_i binary



domain store, size 2^4

MDD store, size 2

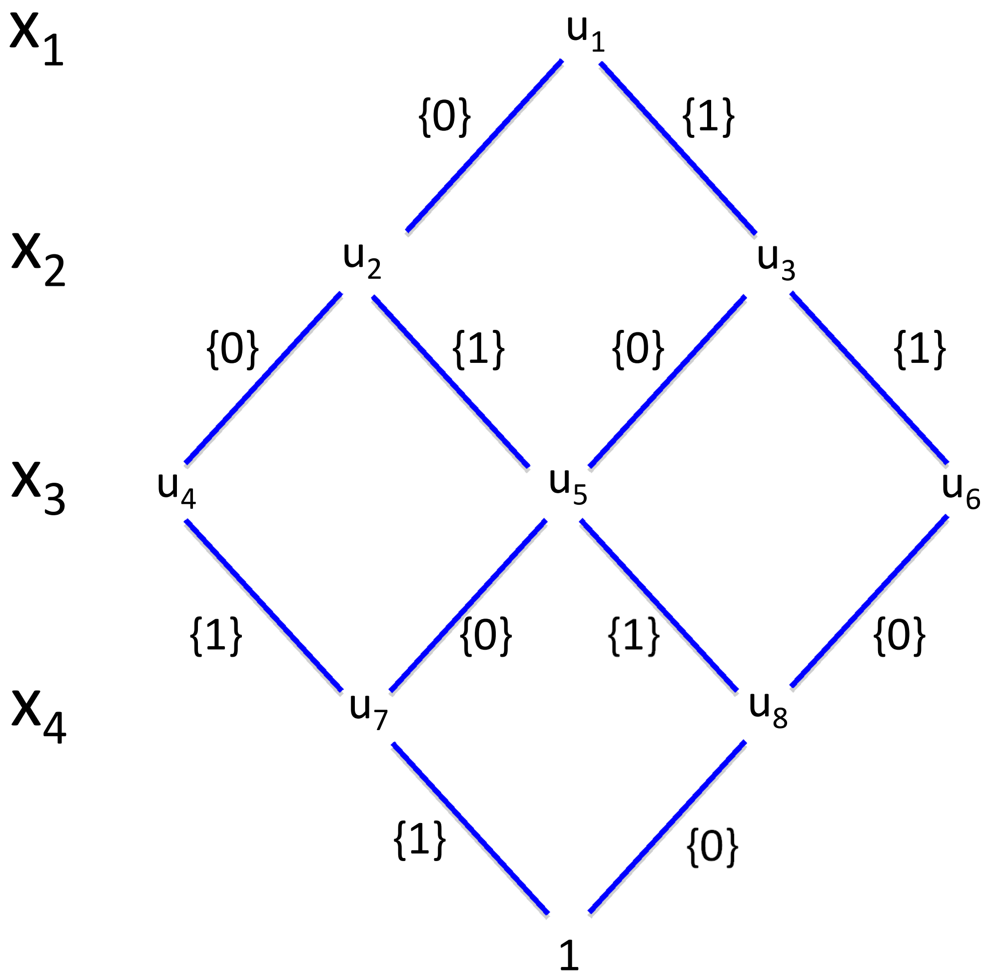
- Given a set of variables X, and a set of values S, a lower bound L and upper bound U,

$$\texttt{among}(X, S, L, U) := \quad L \leq \sum_{x \in X} (x \in S) \leq U$$

"among the variables in X, at least L and at most U take a value from the set S"

- Applications in, e.g., sequencing and scheduling
- WLOG assume that X are binary and S = {1}

Example: MDD for Among



Exact MDD for `among`$(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

MDD-based constraint programming

- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (domain store)
  - Dynamically adjust MDD based on constraints

- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges
  - Node refinement: Split nodes to separate edge information

- Search
  - As in classical CP, but may now be guided by MDD

MDD Filtering for Among

Goal: Given an MDD and an `among` constraint, remove *all* inconsistent edges from the MDD
(establish MDD-consistency)

Approach:

- Compute path lengths from the top node and from the bottom node
- Remove edges that are not on a path with lengths between lower and upper bound

- Complete (MDD-consistent) version
  - Maintain all path lengths; quadratic time
- Partial version (does not remove all inconsistent edges)
  - Maintain and check bounds (longest and shortest paths); linear time

Node refinement for Among

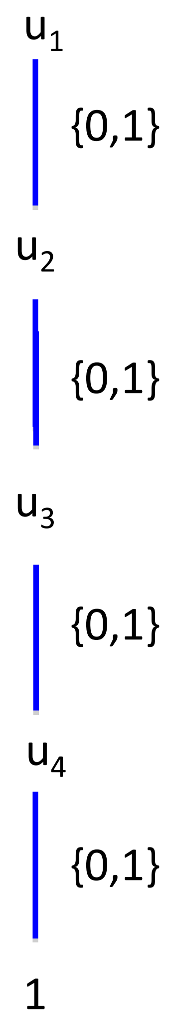
For each layer in MDD, we first apply edge filter, and then try to refine

- consider incoming edges for each node
- split the node if there exist incoming edges that are not equivalent (w.r.t. path length)

Example:

- We will propagate `among`($\{x_1,x_2,x_3,x_4\},\{1\},2,2$) through a BDD of maximum width 3

Example



Try to filter edge domain (u_1,u_2)

SP using (u_1,u_2,{0,1}) has length < U
LP using (u_1,u_2,{0,1}) has length > L

Can't filter

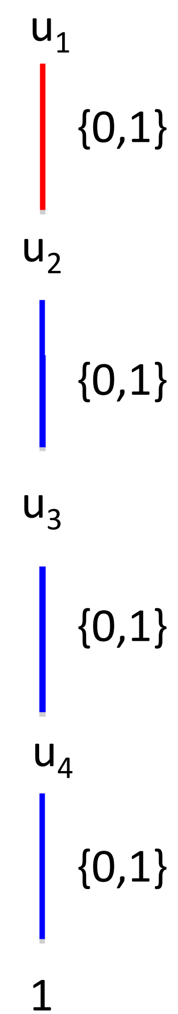
among($\{x_1,x_2,x_3,x_4\},\{1\},2,2$)

Example

Split u_2?

SP using $(u_1,0) = 0$
SP using $(u_1,1) = 1$

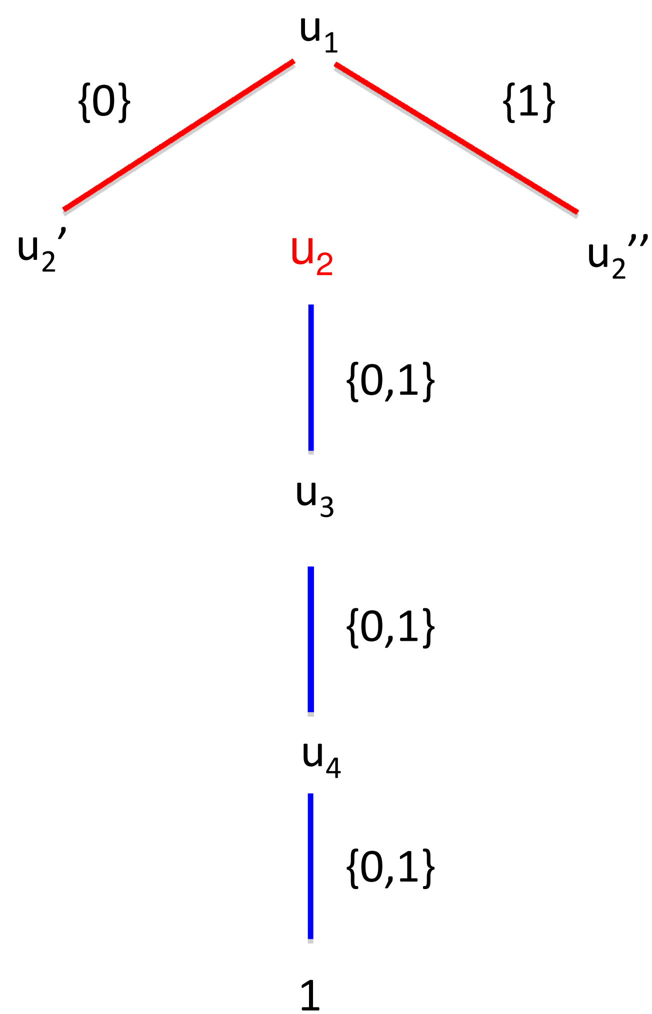
Incoming edge-value pairs are not equivalent: so split u_2



$\texttt{among}(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

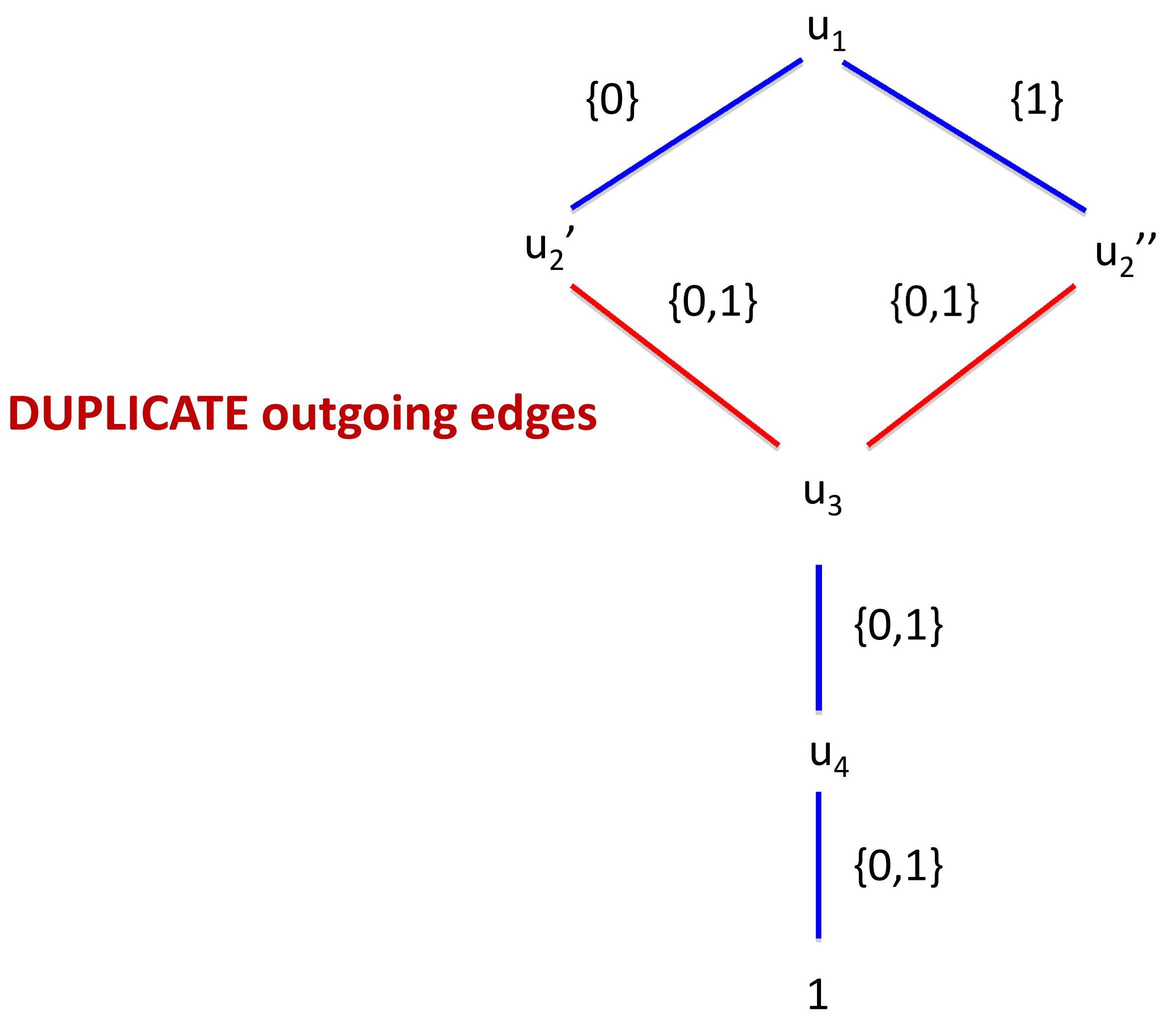
Example

SPLIT u_2 into two classes (less than maximum width)



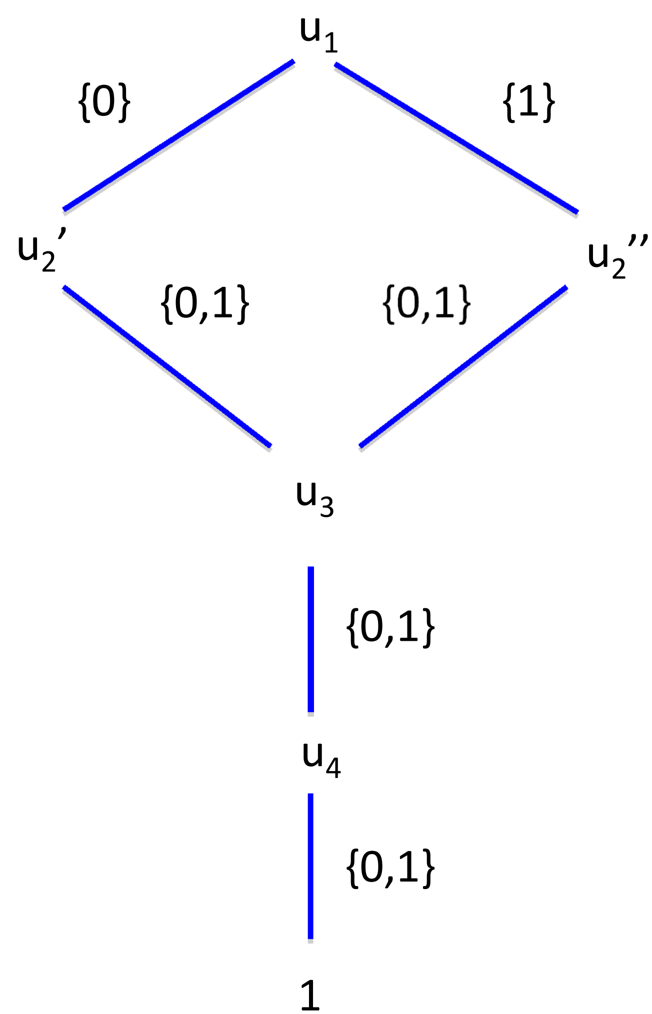
among({x_1,x_2,x_3,x_4},{1},2,2)

Example



$$\texttt{among}(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$$

Example



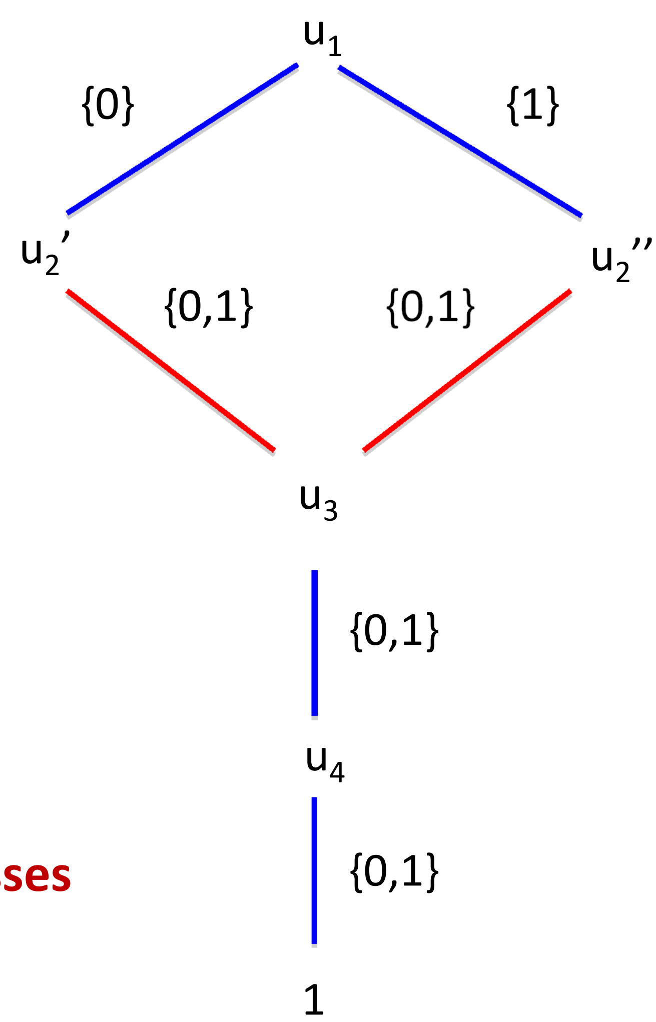
Filter edge domains (u_2',u_3) and (u_2'',u_3)

(No filtering possible)

$\texttt{among}(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example

Carnegie Mellon Tepper School of Business



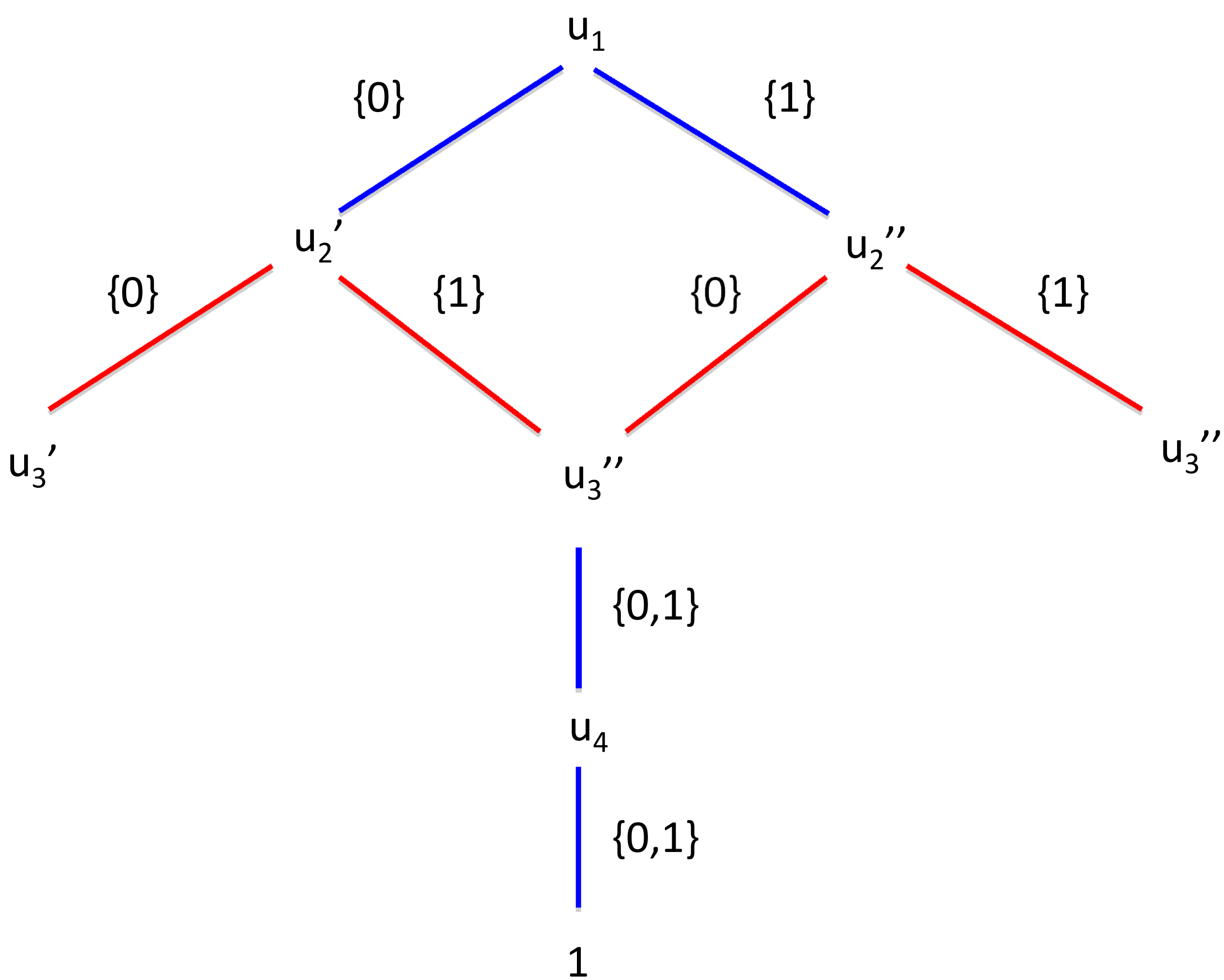
Split u_3?

SP $(u_2',u_3,0) = 0$
SP $(u_2',u_3,1) = 1$
SP $(u_2'',u_3,0)= 1$
SP $(u_2'',u_3,1)= 2$

Split u_3 into 3 equivalence classes

among($\{x_1,x_2,x_3,x_4\},\{1\},2,2$)

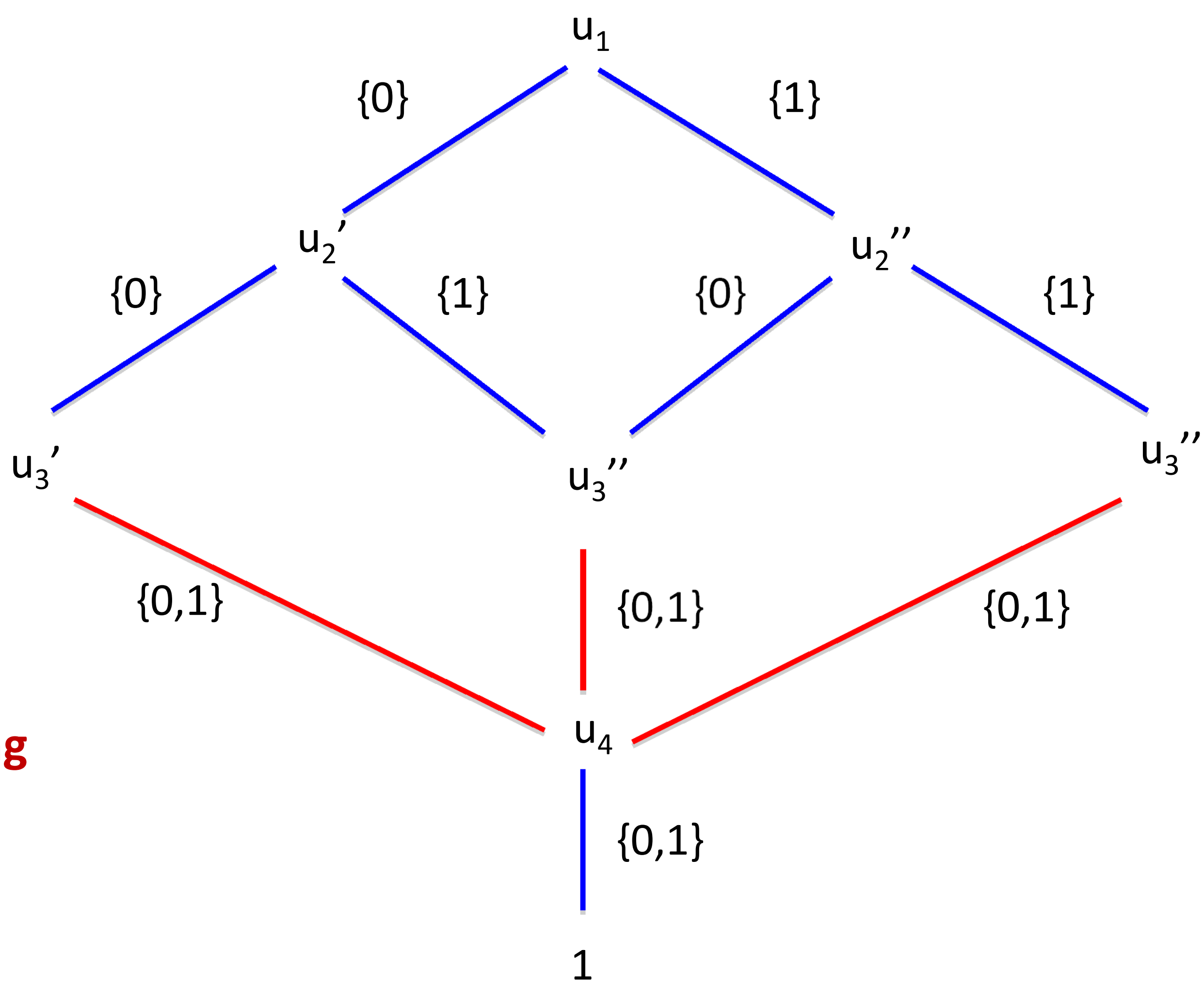
Example



Split u_3 into 3 equivalence classes

among($\{x_1,x_2,x_3,x_4\},\{1\},2,2$)

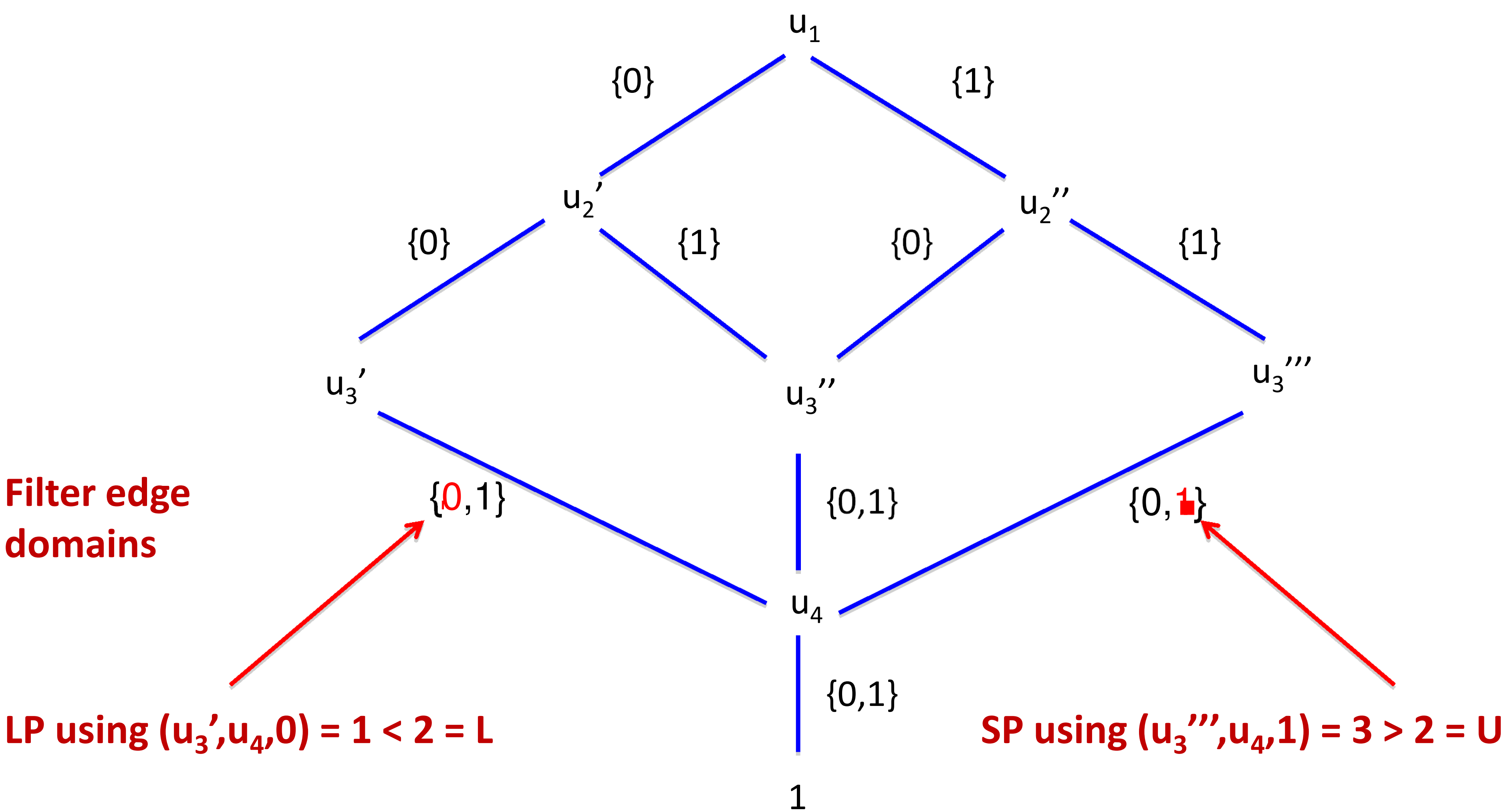
Example



Duplicate outgoing edges

among$(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

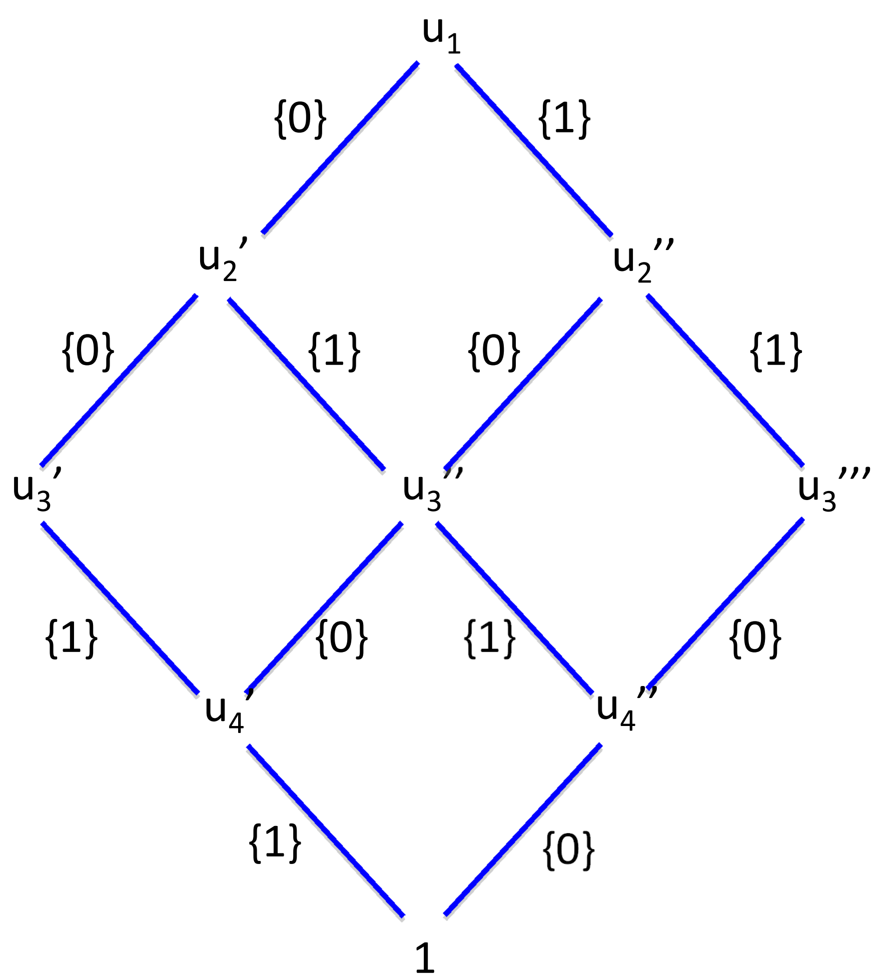
Example



$$\text{among}(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$$

Example

Continuing…



among($\{x_1,x_2,x_3,x_4\}$,{1},2,2)

Approximate equivalence

Example: edge-value equivalence was exact

- Problem: a few nodes “consume” BDD when processing a constraint

Remedy: approximate equivalence

- Edge-value pairs are equivalent if SPs/LPs differ by at most some threshold value

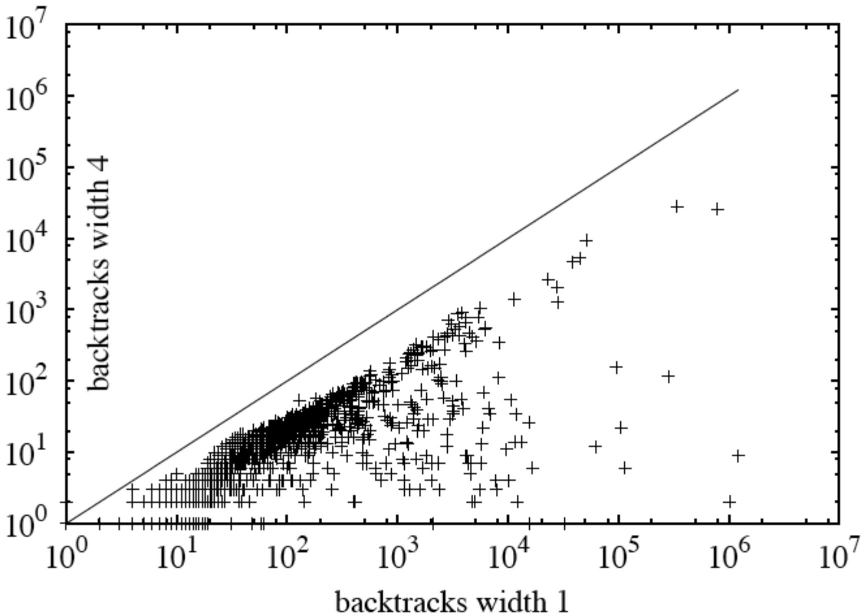
Experiments

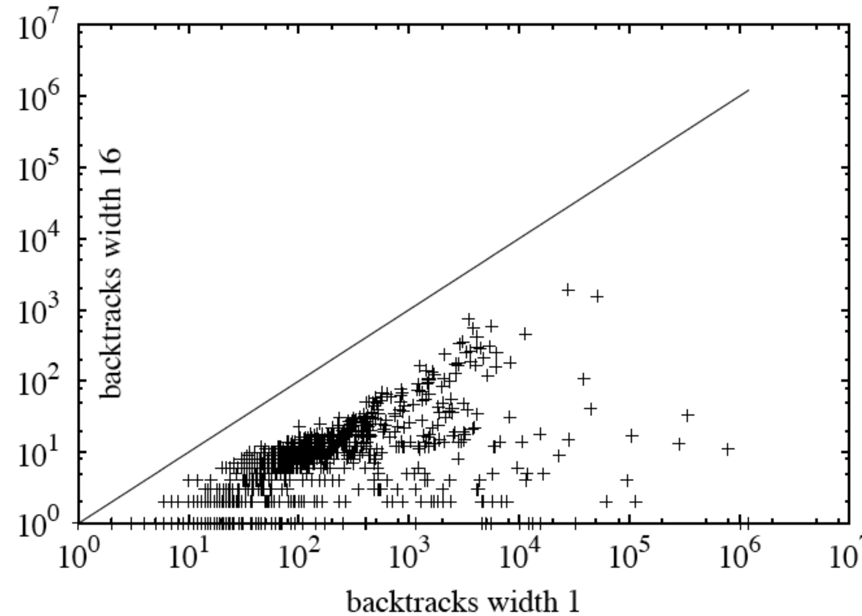
- Multiple among constraints
  - 50 binary variables total
  - 5 variables per among constraint, indices chosen from normal distribution with uniform-random mean in [1..50] and stdev 2.5, modulo 50
  - Classes: 5 to 200 among constraints (step 5), 100 instances per class

- Nurse rostering instances (horizon *n* days)
  - Work 4-5 days per week
  - Max A days every B days (Max A/B)
  - Min C days every D days (Min A/B)
  - Three problem classes

- Compare width 1 (domain store) with increasing widths

Multiple Amongs: Backtracks

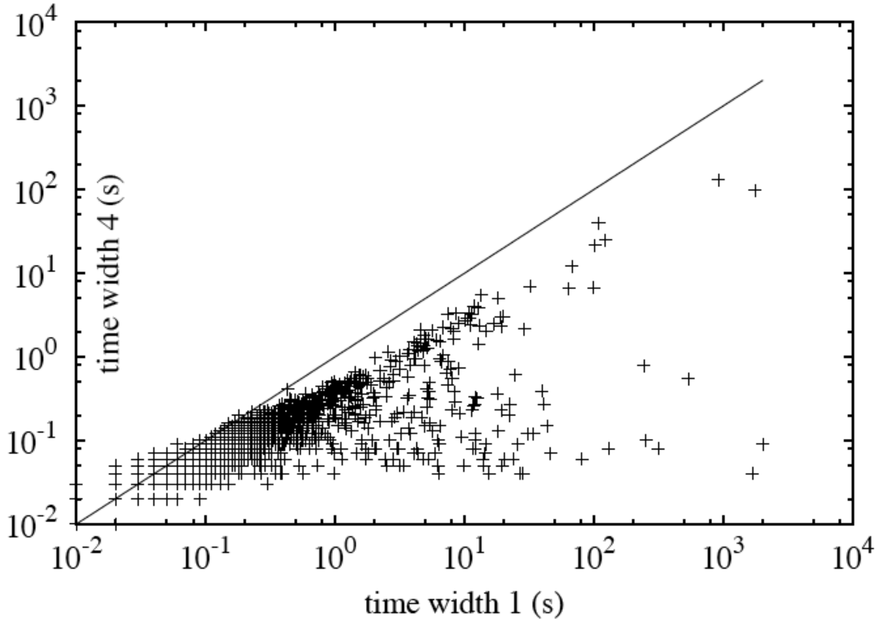




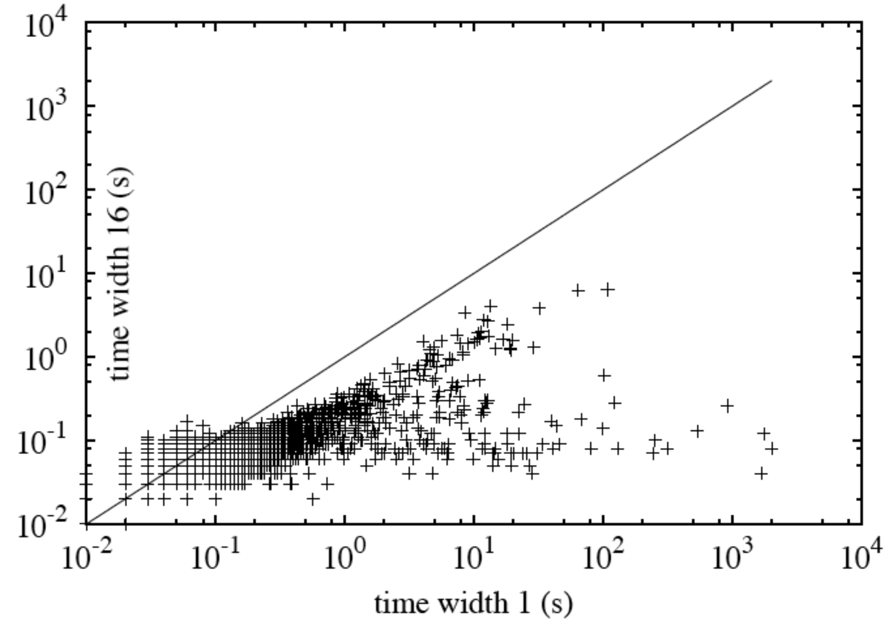
width 1 vs 4

width 1 vs 16

Multiple Amongs: Running Time



width 1 vs 4



width 1 vs 16

Nurse rostering problems

		Width 1		Width 4		Width 32	
	Size	BT	CPU	BT	CPU	BT	CPU
Class 1	40	61,225	55.63	8,138	12.64	3	0.09
	80	175,175	442.29	5,025	44.63	11	0.72
Class 2	40	179,743	173.45	17,923	32.59	4	0.07
	80	179,743	459.01	8,747	80.62	2	0.32
Class 3	40	91,141	84.43	5,148	9.11	7	0.18
	80	882,640	2,391.01	33,379	235.17	55	3.27

Conclusion

- MDD store provides substantial advantage over domain store for filtering multiple among constraints
  - Wider MDDs yield greater speedups
  - Huge reduction in the amount of backtracking and solution time

- Intensive processing at search nodes can pay off when more structural information is communicated between constraints