Propagating Separable Equalities in an MDD Store

Tarik Hadzic
Cork Constraint Computation Centre
John Hooker
Carnegie Mellon University
Peter Tiedemann
Configit Software

INFORMS 2008

Constraint Programming

- **Strength** of CP – processes individual constraints.
  - Exploits problem substructure.

- **Weakness** of CP – processes individual constraints.
  - No global point of view.
  - Overlooks implications of combined constraints.

- Two compensating strategies:
  - **Global constraints**
  - **Domain store relaxation**

The domain store

- Stores the current variable domains
  - Values that occur in some feasible solution
- Provides a **global** point of view.
  - …to some extent.
  - **Pools** results of individual constraint processing.
  - Basis for **constraint propagation**.

Advantages of domain store

- Provides **natural input** into filtering algorithms.
  - Filters start with current domains when processing a constraint.
- Guides **branching** (on variables) in a natural way.
  - Simply split the domain in the current domain store.

Disadvantages of domain store

- Transmits relatively **little information**.
- A **weak relaxation** of the problem.
  - Ignores interactions between variables.
  - Feasible set is simply a Cartesian product of domains.
- **Unbalanced tradeoff.**
  - Search trees **too large**.
  - **Too little processing** at nodes.

A stronger discrete relaxation

- Relaxed **multivalued decision diagram** can serve as a constraint store.
  - Generalization of **binary decision diagram**, used in circuit verification, configuration problems, etc.

- An MDD is a **compact representation** of a search tree for the problem.
  - **Isomorphic subtrees** are merged.
  - MDD is **relaxed** by limiting its width.

Advantages of MDD store

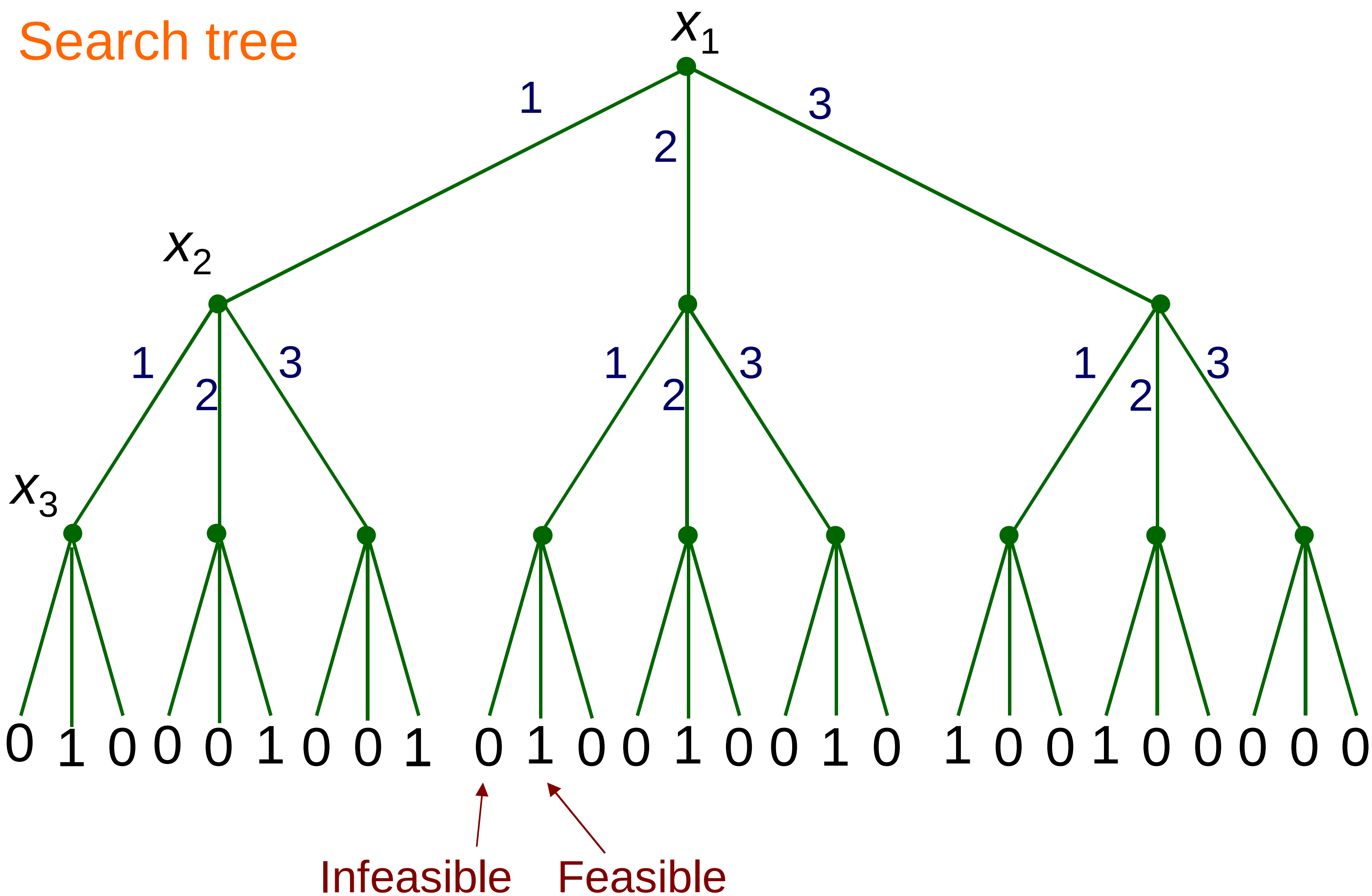
- A much **stronger relaxation** than domain store.
  - Strength of relaxation is **adjustable**, ranging from domain store to original problem.
- **Guides branching** in a natural way.
  - More guidance than domain store.
- Justifies **more processing** at search tree nodes.
  - Better integration of CP/IP.

Example problem

$$\begin{pmatrix} x_1 + x_3 = 4 \\ 3 \le x_1 + x_2 \le 5 \end{pmatrix} \vee \ (x_1, x_2, x_3) = (1,1,2)$$

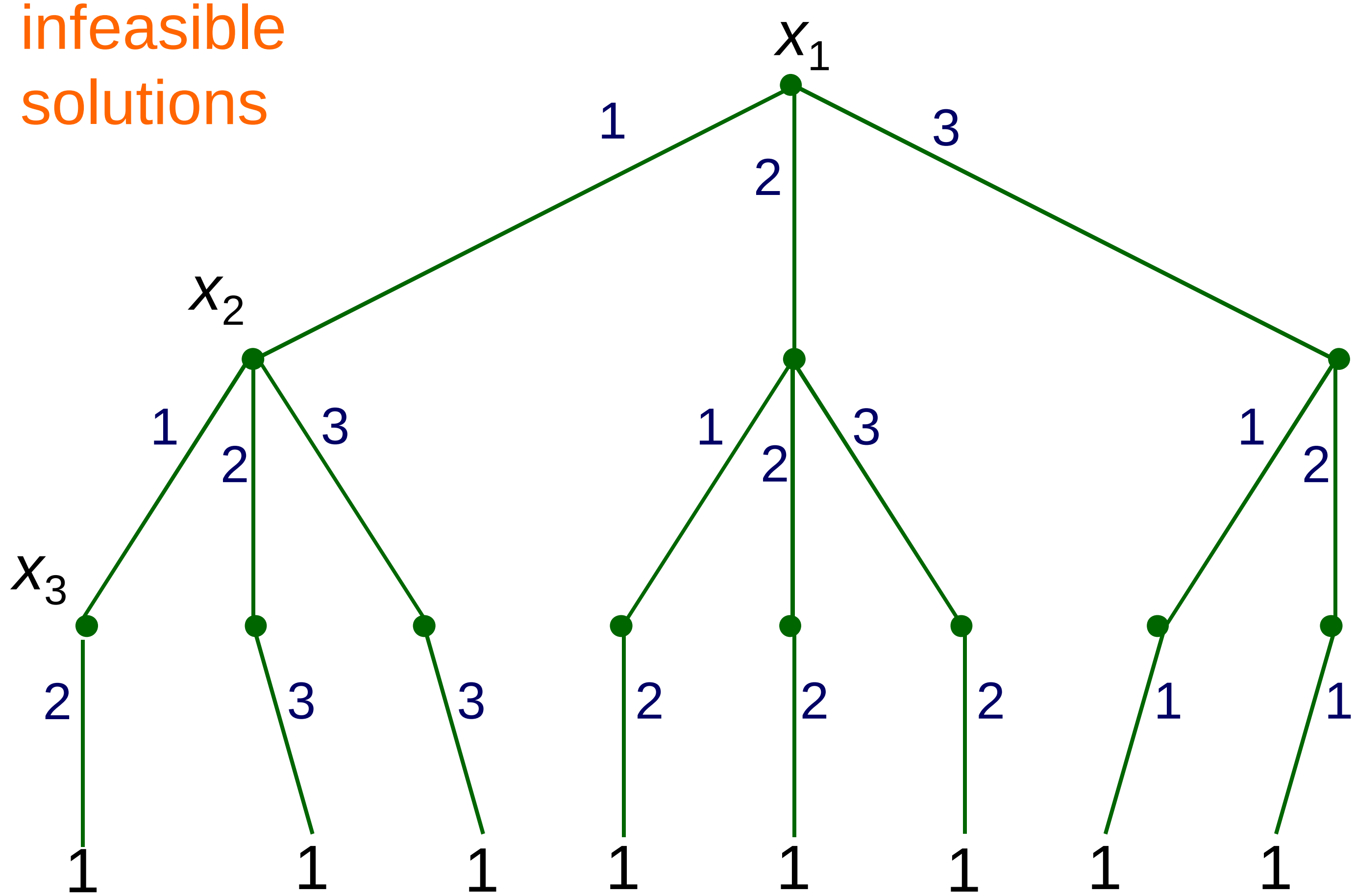
$$x_j \in \{1,2,3\}$$

Search tree

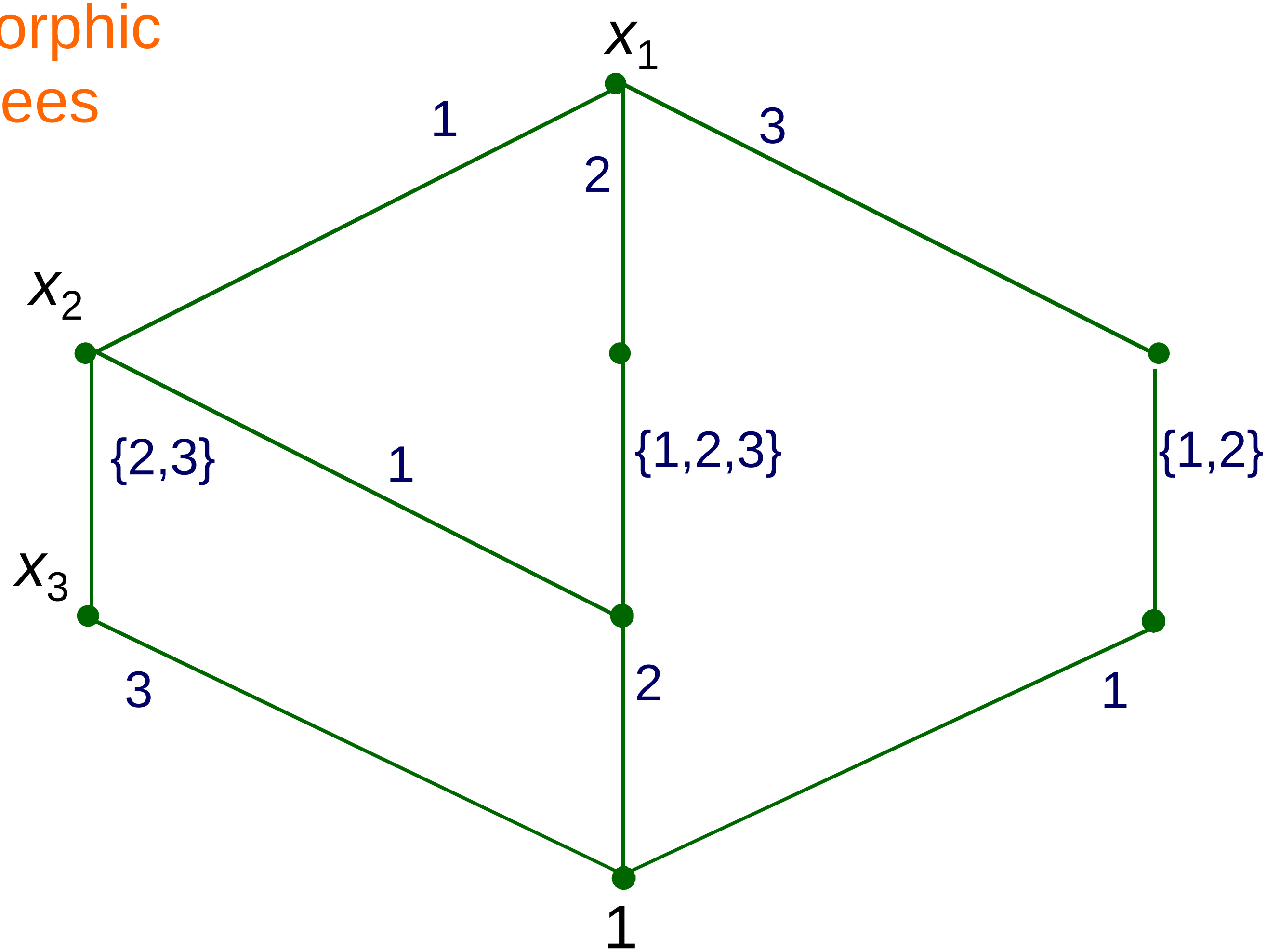


x_1
x_2
x_3
1
2
3
1
2
3
1
2
3
1
2
3
0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0
Infeasible
Feasible

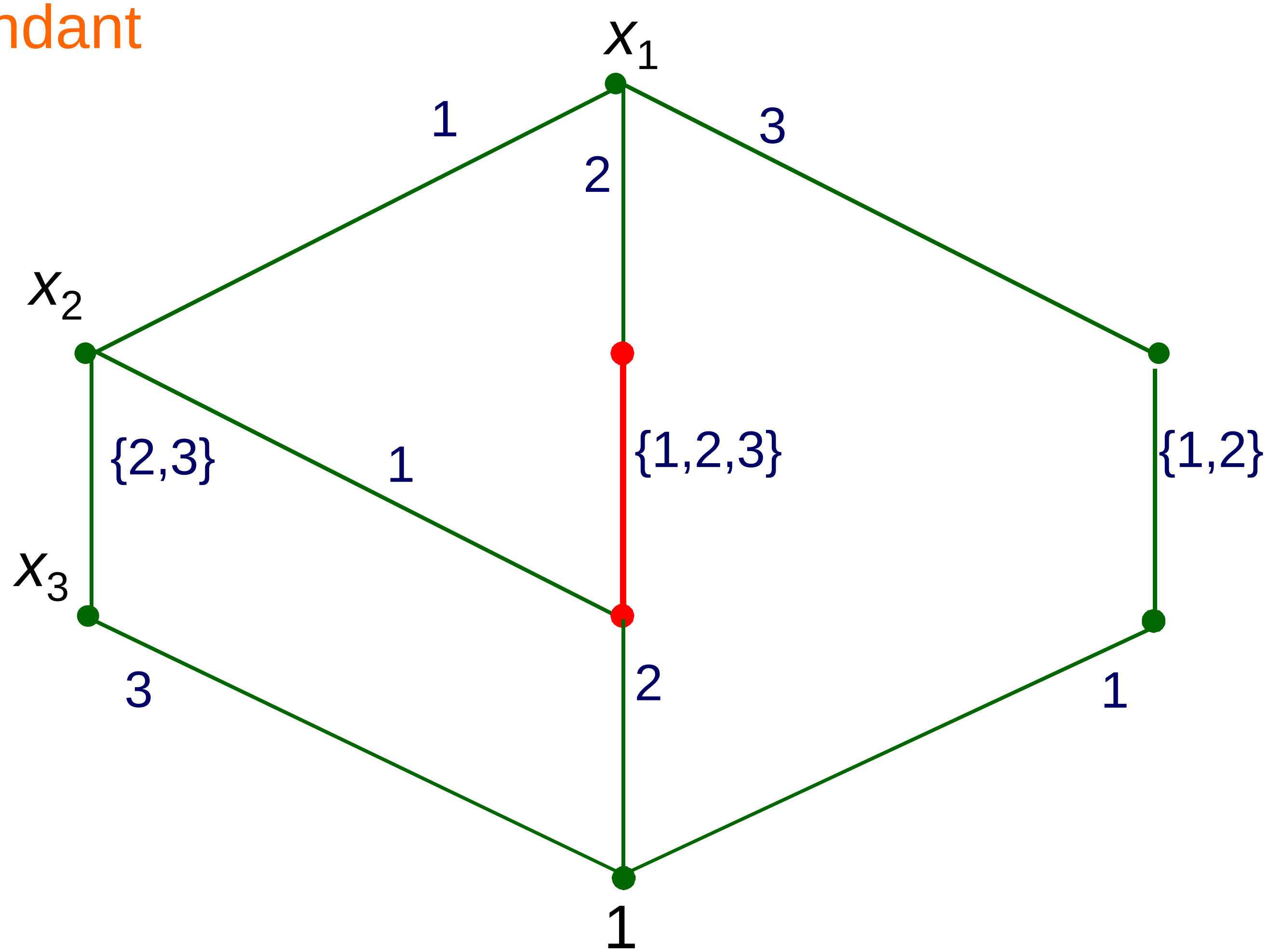
Remove infeasible solutions



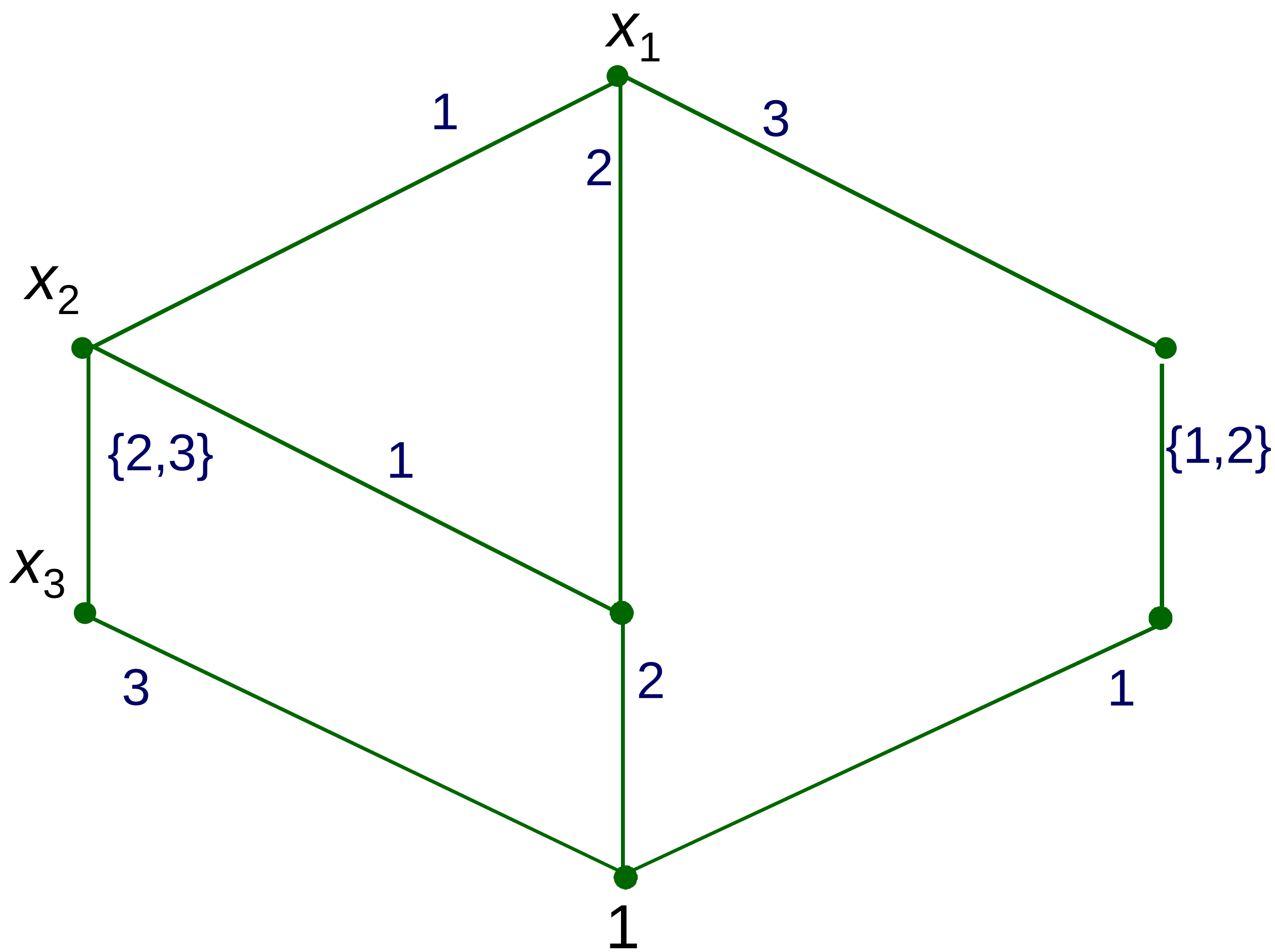
Merge isomorphic subtrees



Remove redundant edge

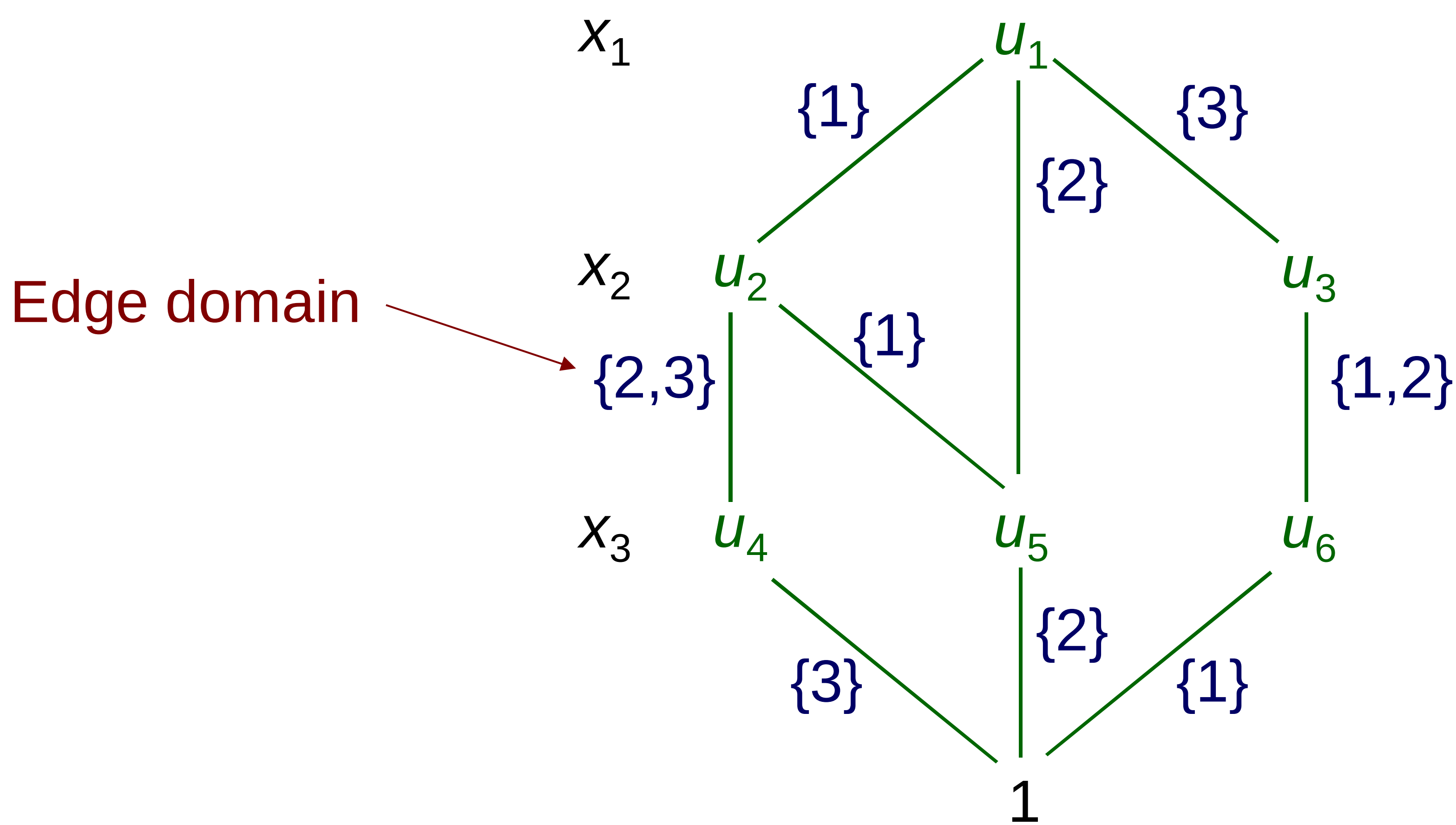


Reduced MDD

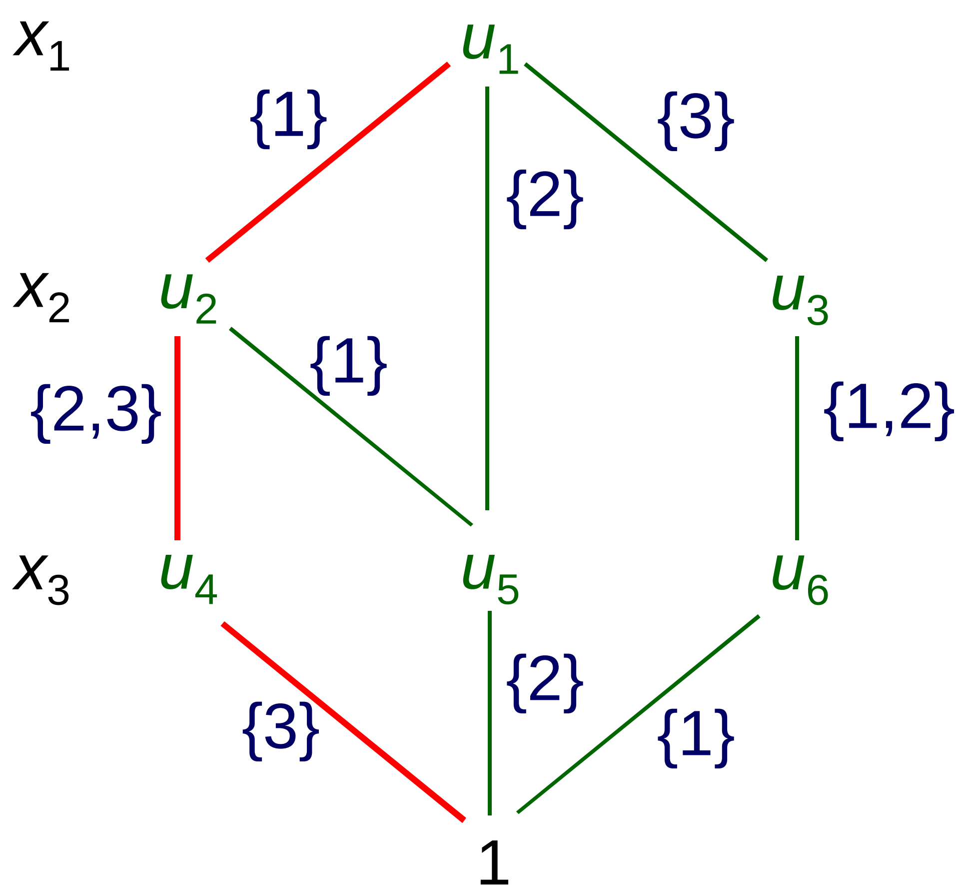


$$\begin{pmatrix} x_1 + x_3 = 4 \\ 3 \le x_1 + x_2 \le 5 \end{pmatrix} \vee \ (x_1, x_2, x_3) = (1,1,2)$$

$$x_j \in \{1,2,3\}$$



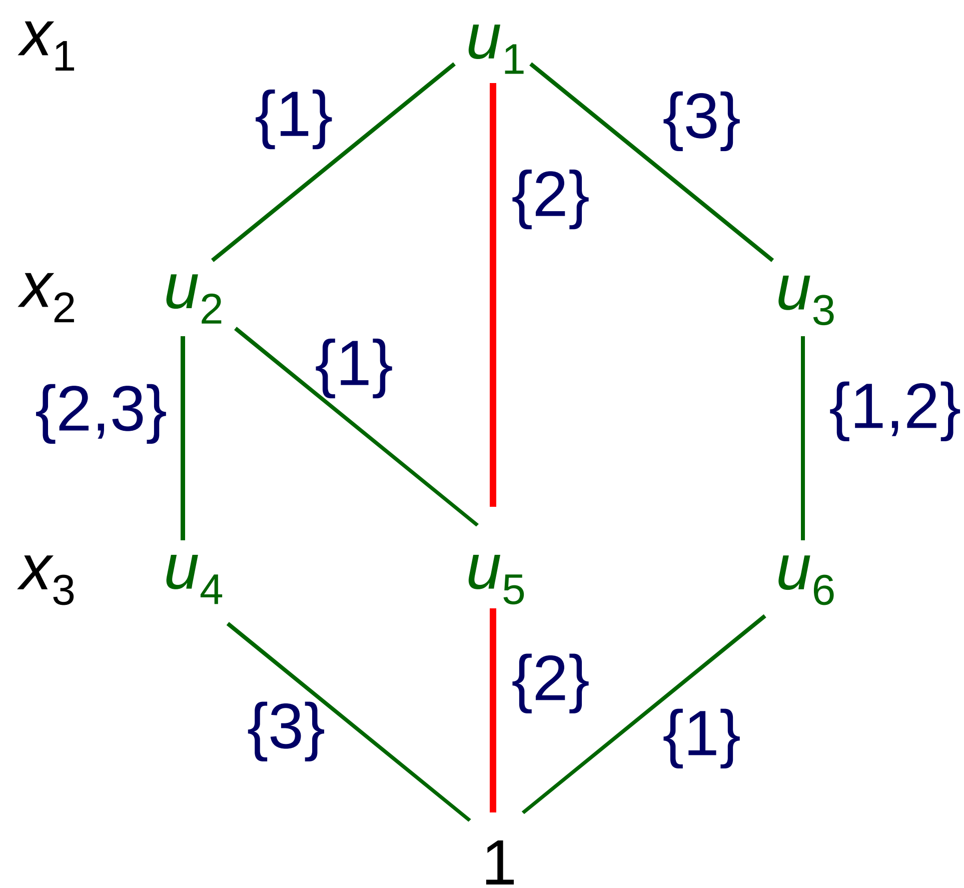
Each path represents a cartesian product of solutions



$\{1\}\times\{2,3\}\times\{3\}$

Each path represents a cartesian product of solutions

$\{2\}\times\{1,2,3\}\times\{2\}$

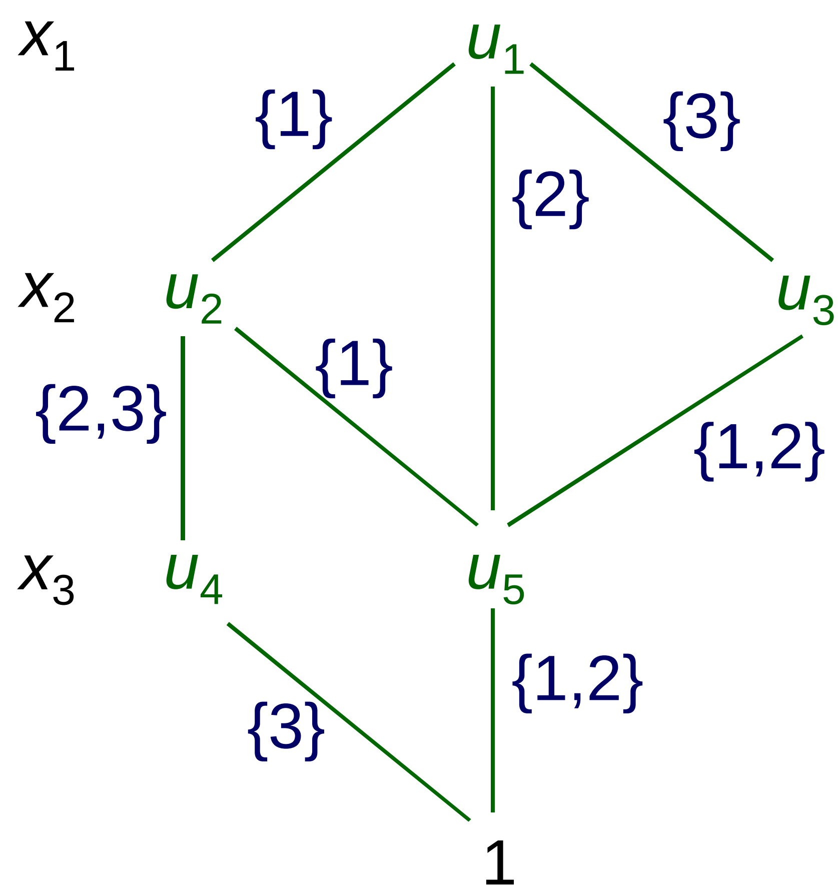


MDD Relaxation

Let's use relaxed MDD with max width of 2

Full MDD:
8 solutions

Relaxed MDD:
14 solutions



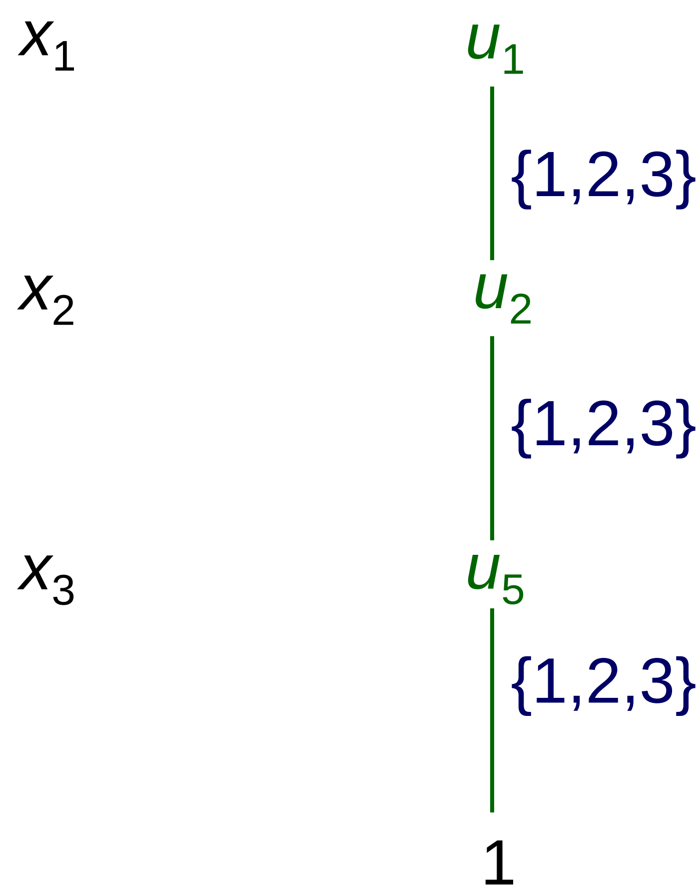
MDD Relaxation

MDD relaxation
with **width 1** is just
the **domain store**.

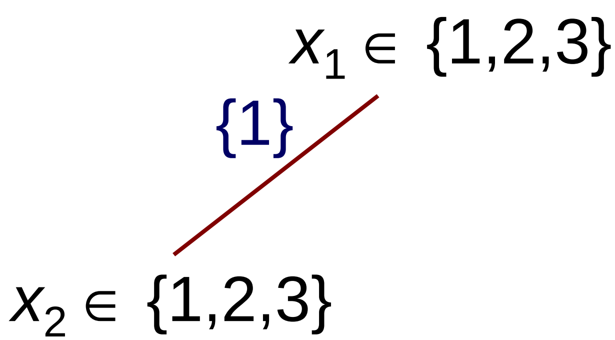
Full MDD:
8 solutions

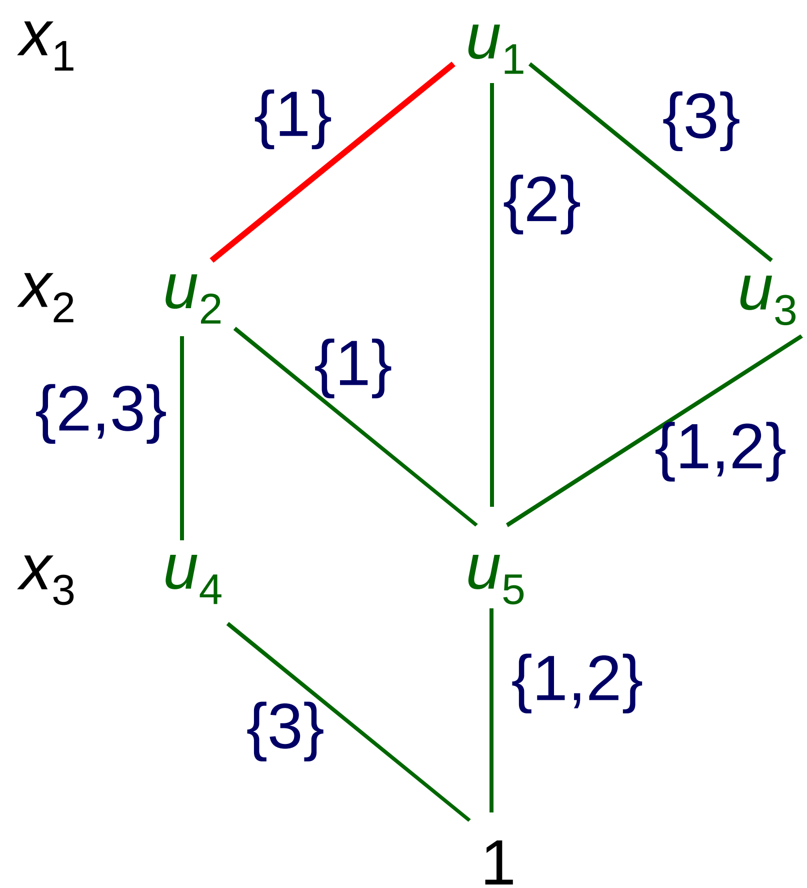
Relaxed MDD:
14 solutions

Domain store:
3×3×3 = 27 solutions

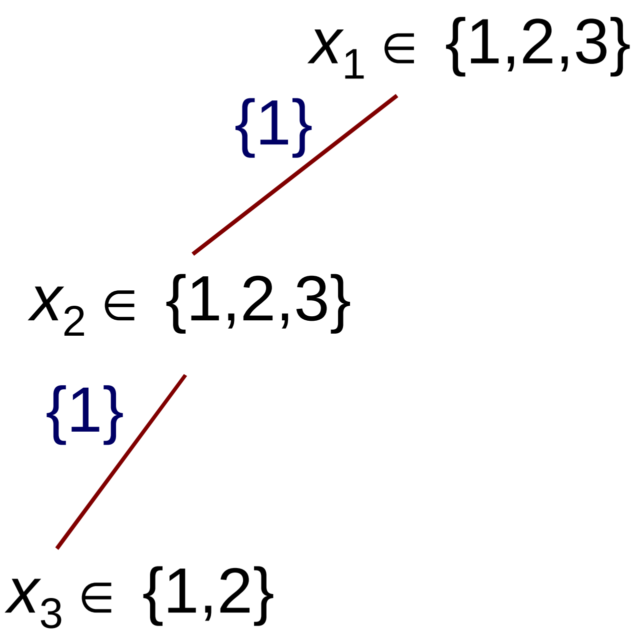


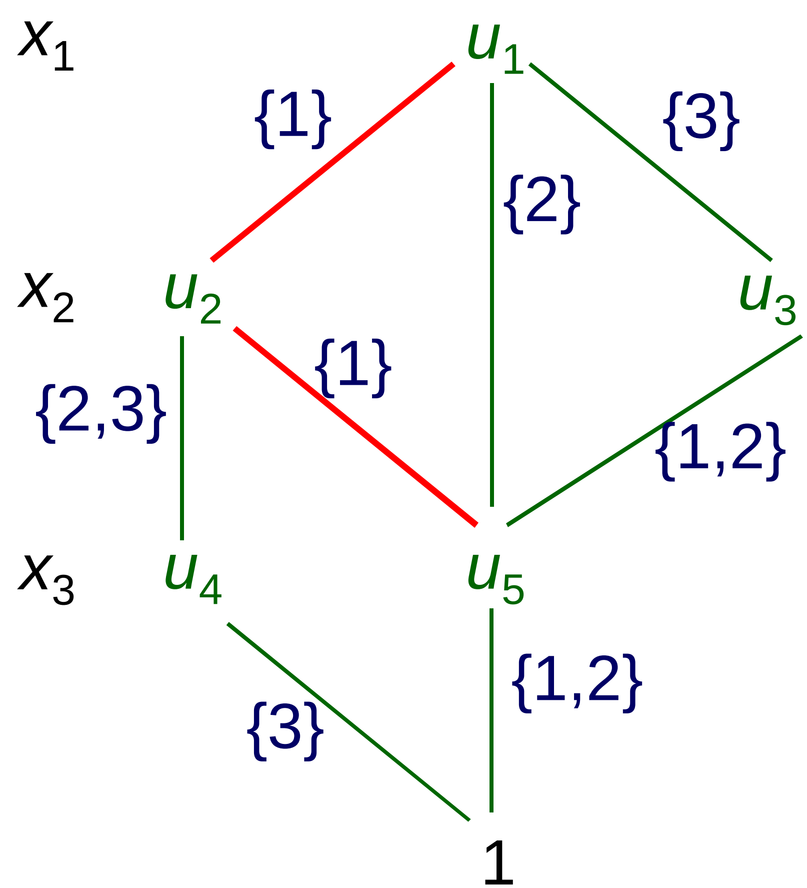
Branching search using relaxed MDD



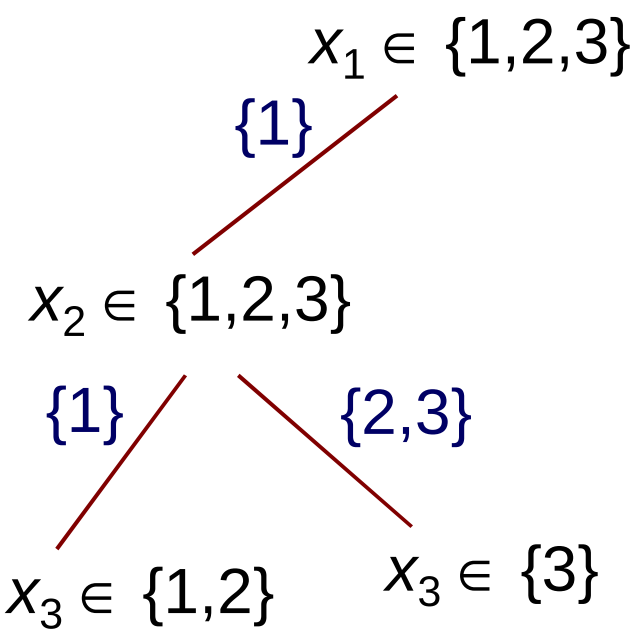


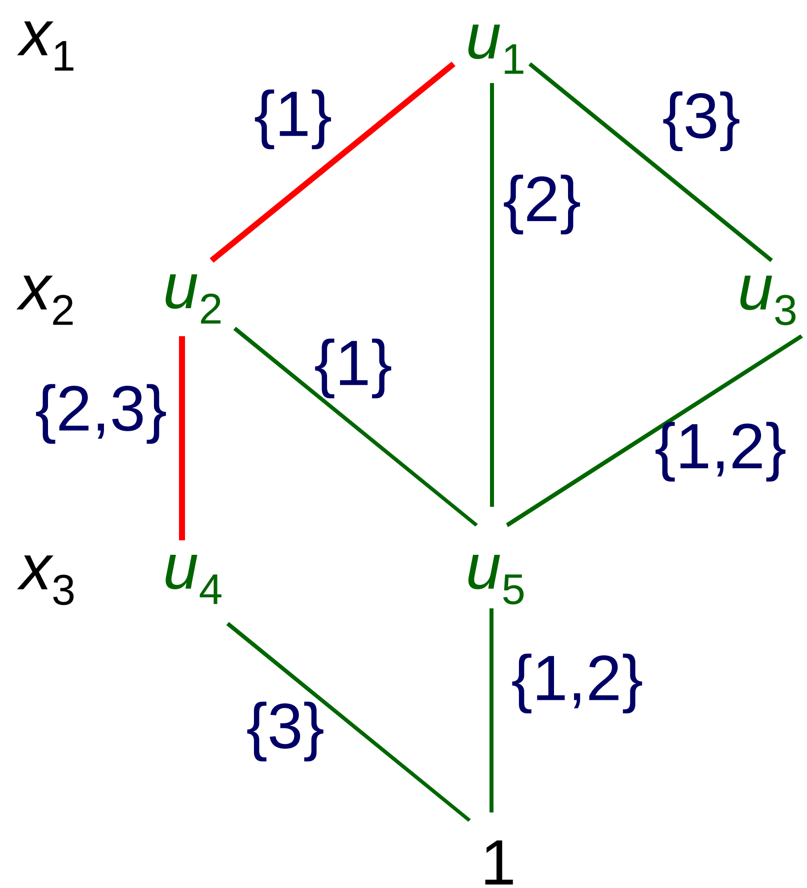
Branching search using relaxed MDD



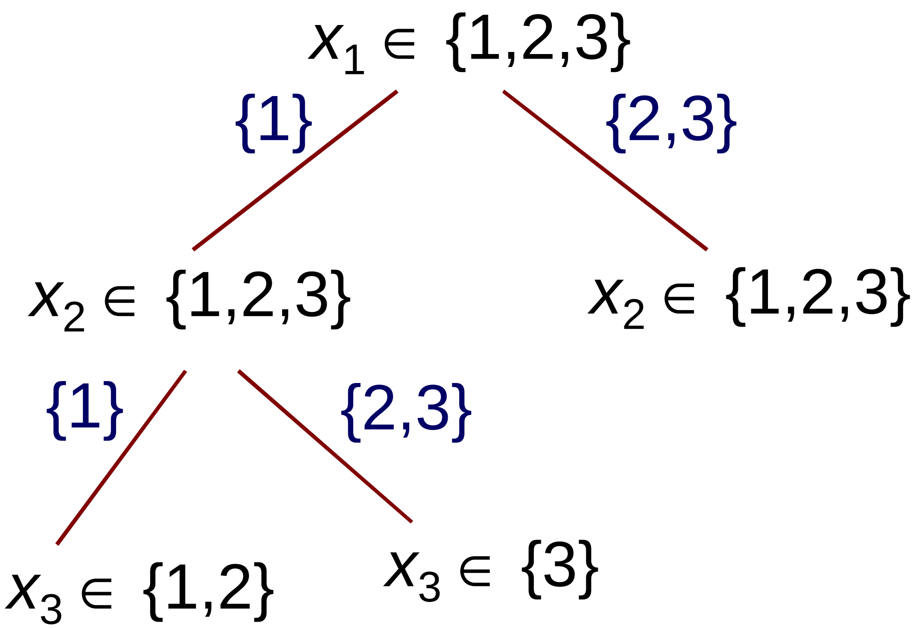


Branching search using relaxed MDD



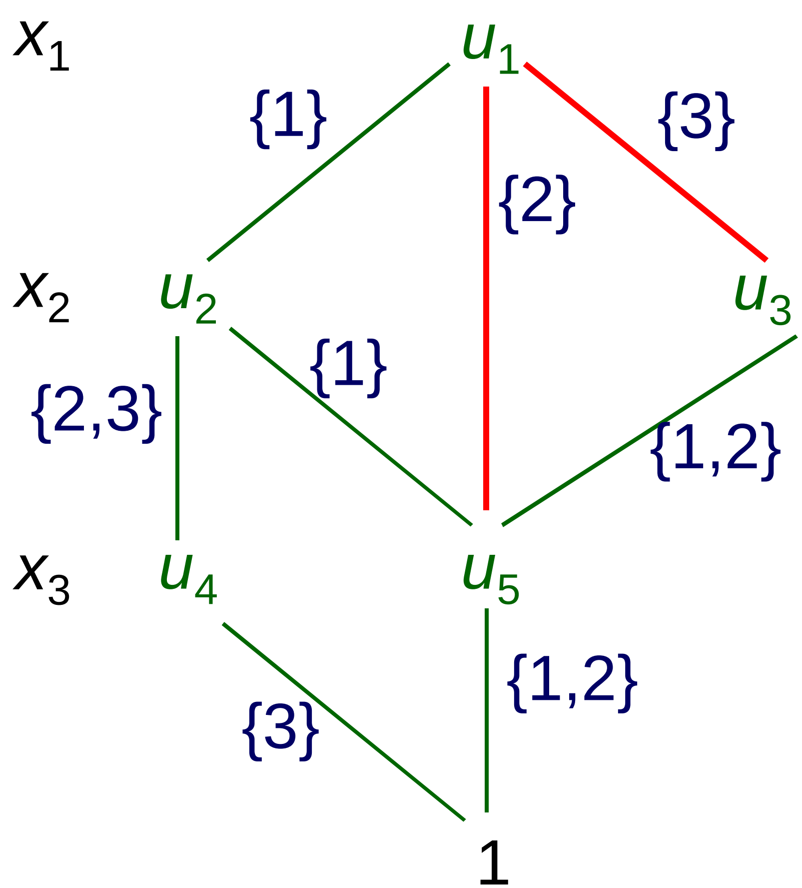


Branching search using relaxed MDD



And so forth.

Less branching than with domain store.



Propagation in MDDs

We can **propagate** a constraint in an MDD relaxation by **edge domain filtering** and **refinement** (node splitting).

We take care not to exceed max width.

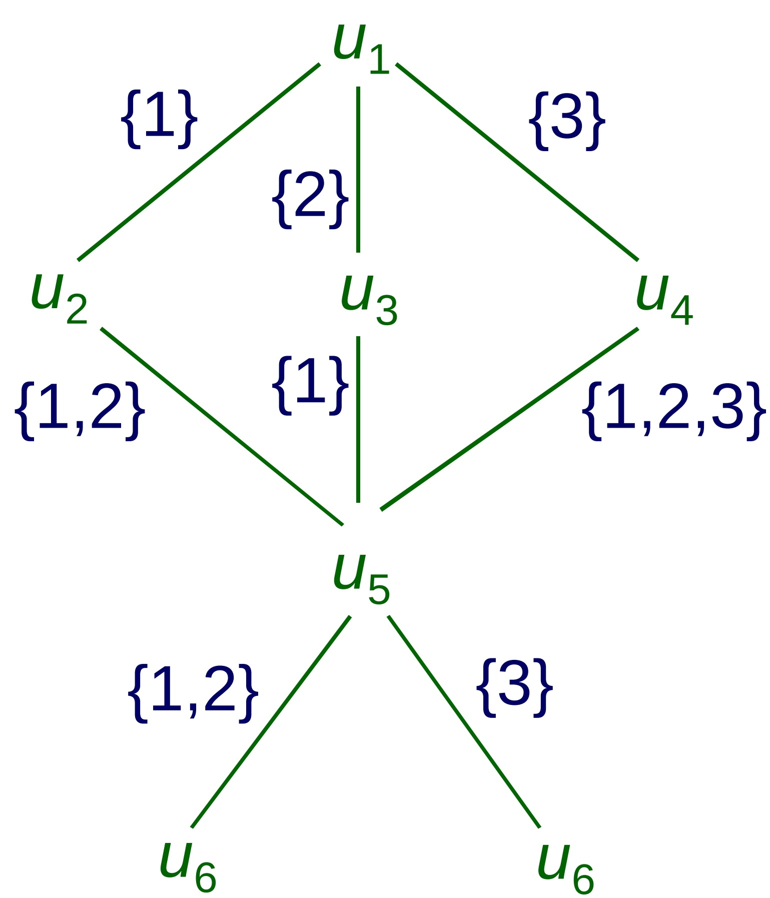
Example:

We will propagate **alldiff** in an MDD relaxation of width 3.

Andersen, Hadzic, Hooker & Tiedemann, A constraint store based on multivalued decision diagrams, 2007

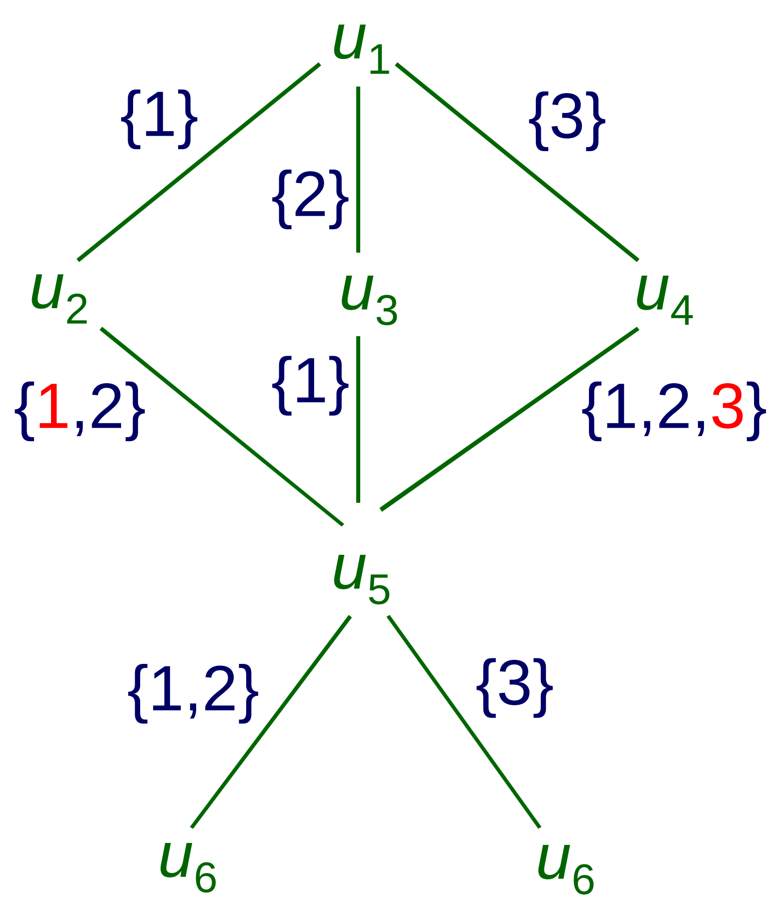
Propagation in MDDs

Current MDD relaxation



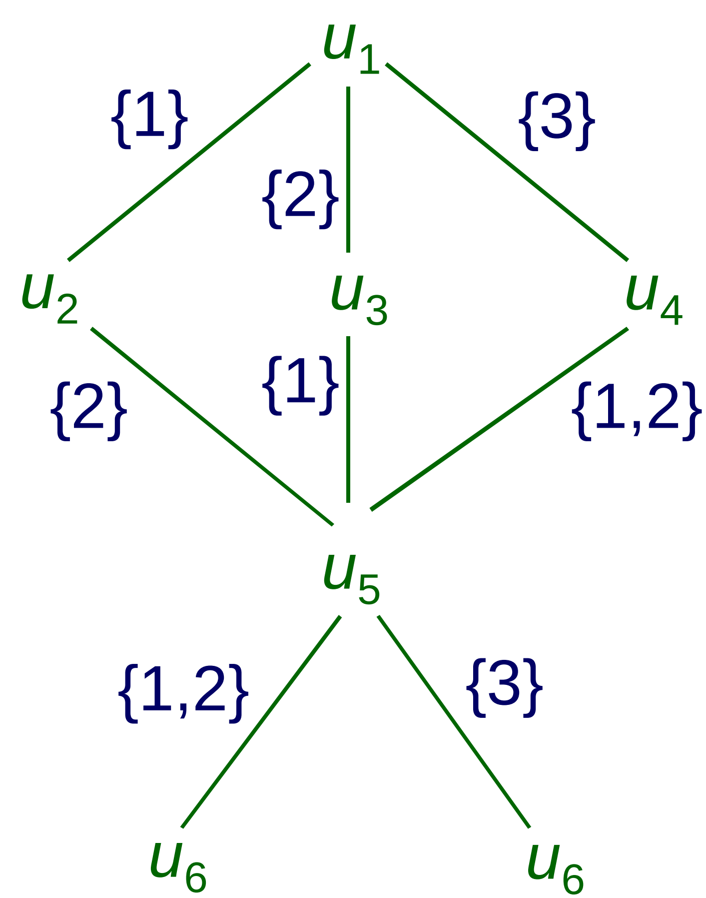
Propagation in MDDs

First filter edge domains using alldiff



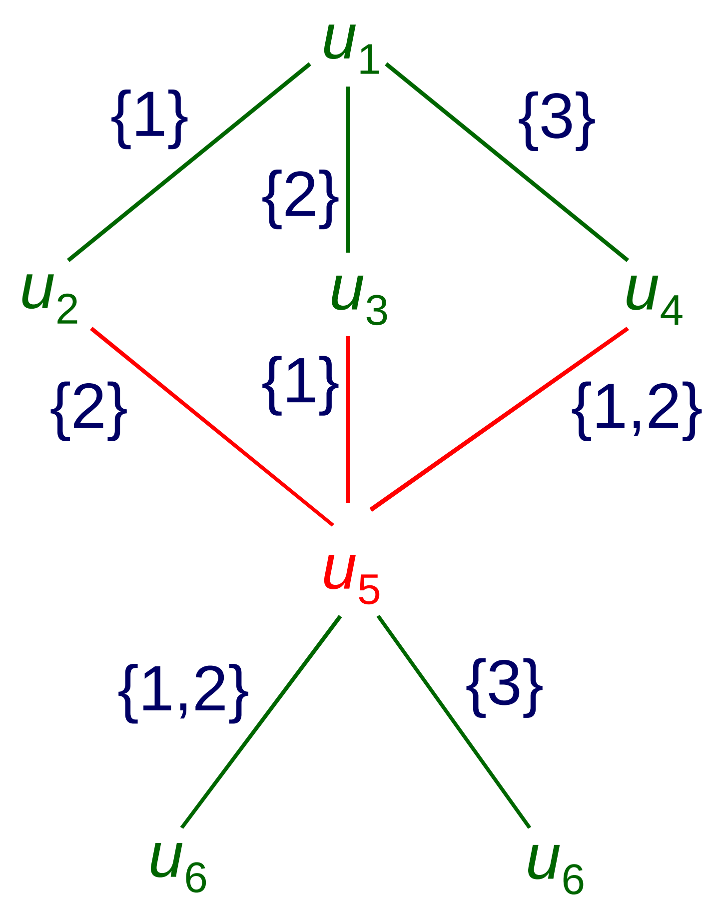
Propagation in MDDs

First filter edge domains using alldiff



Propagation in MDDs

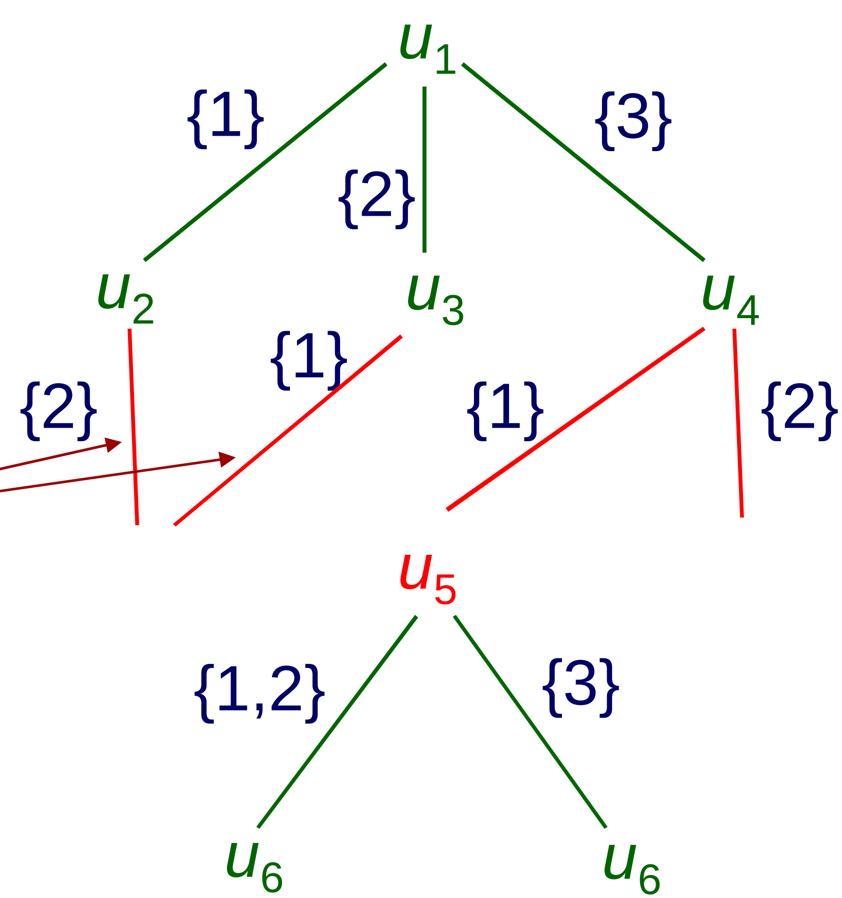
To split u_5:
Identify equivalence classes of incoming edges



Propagation in MDDs

To split u_5:
Identify equivalence classes of incoming edges

These are equivalent for alldiff because they lead to the same set of feasible paths.

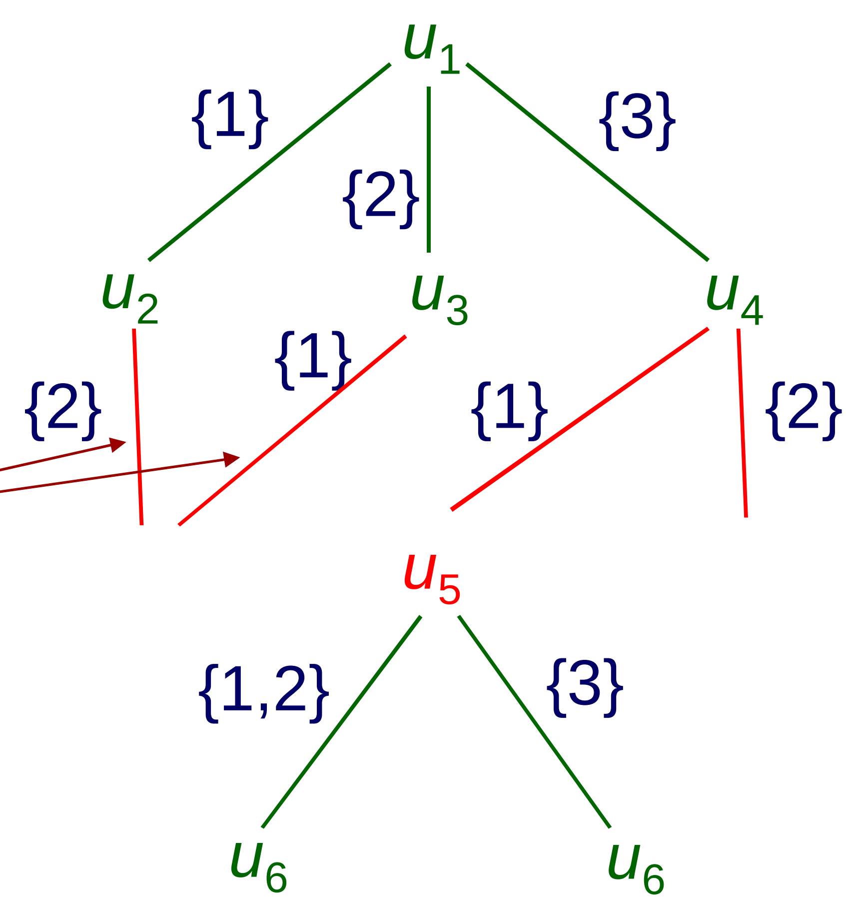


Propagation in MDDs

To split u_5:
Identify equivalence classes of incoming edges

These are equivalent for alldiff because they lead to the same set of feasible paths.

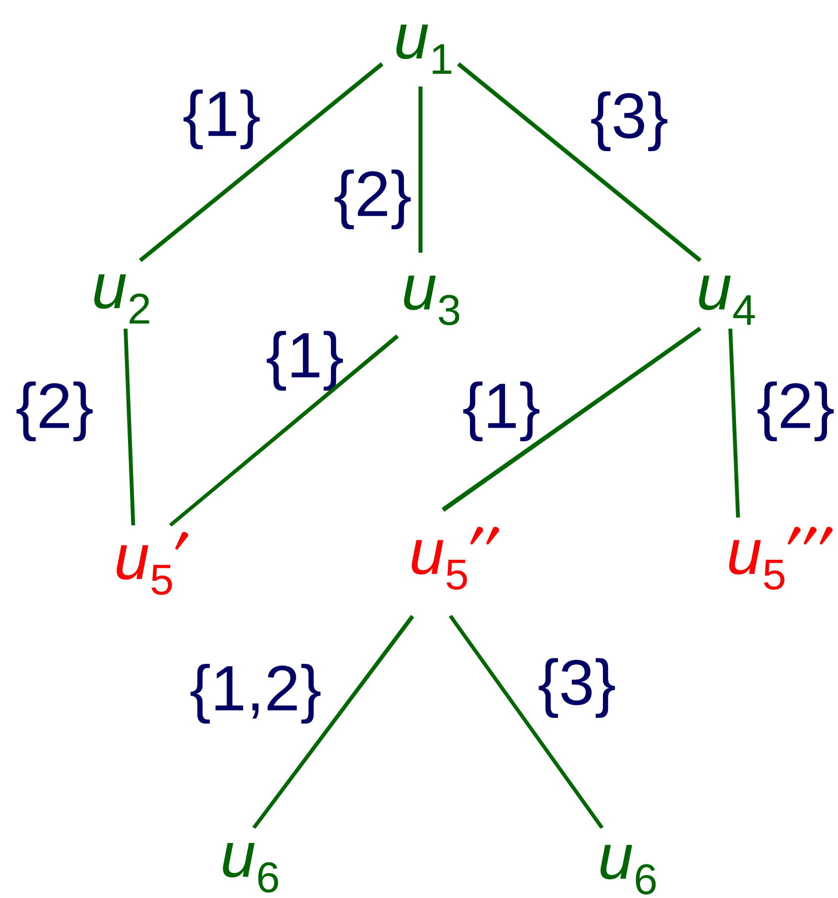
Easy to check equivalence for alldiff – all incoming paths use same values



Propagation in MDDs

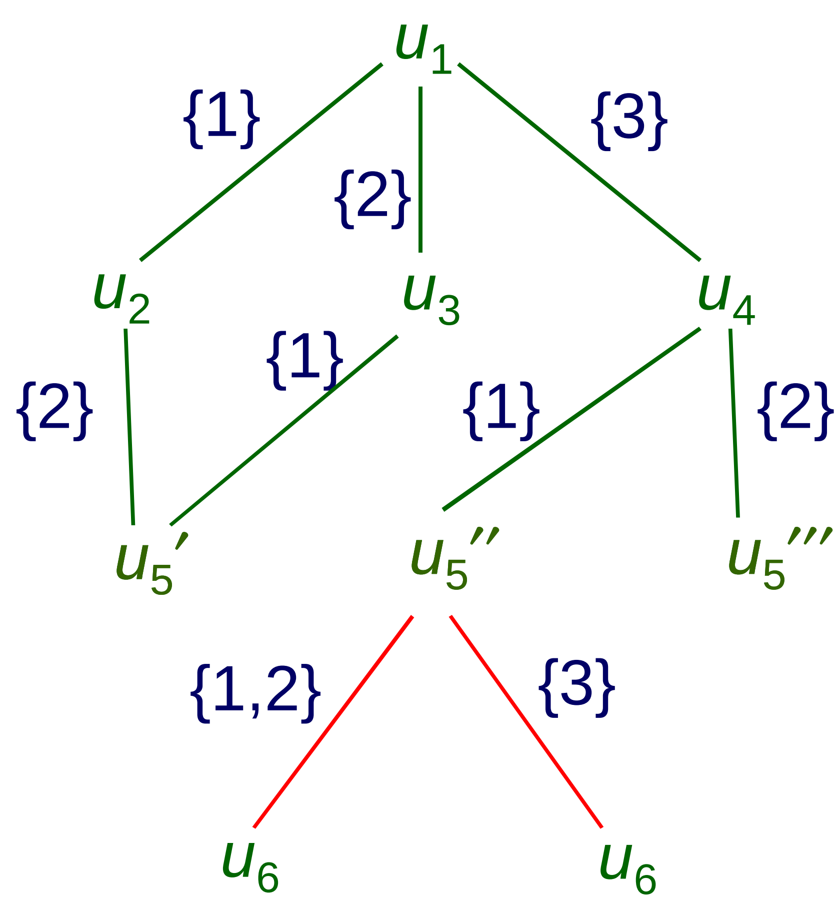
To split u_5:
Identify equivalence classes of incoming edges.

Split u_5 to receive ≤ 3 equivalence classes.



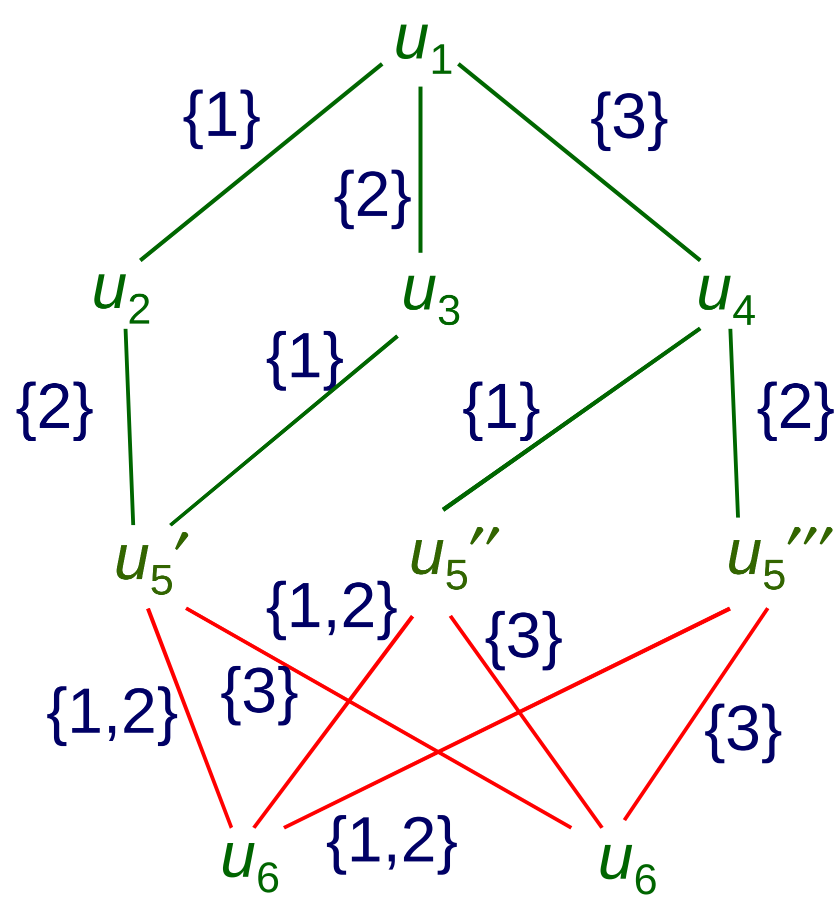
Propagation in MDDs

Duplicate outgoing edges.



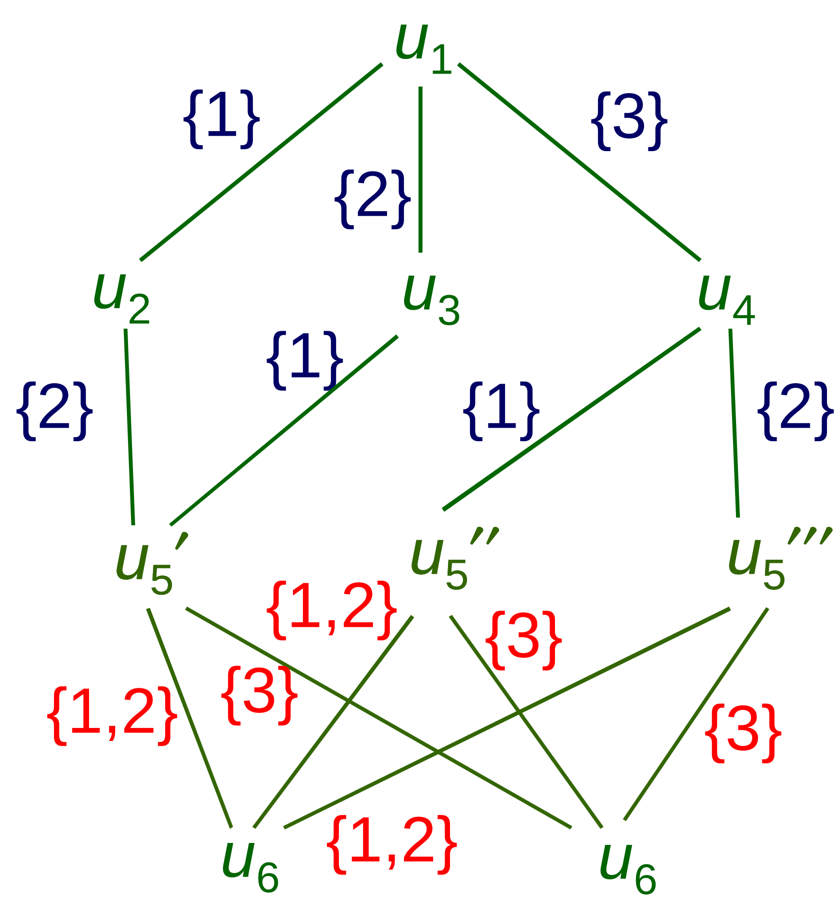
Propagation in MDDs

Duplicate outgoing edges.



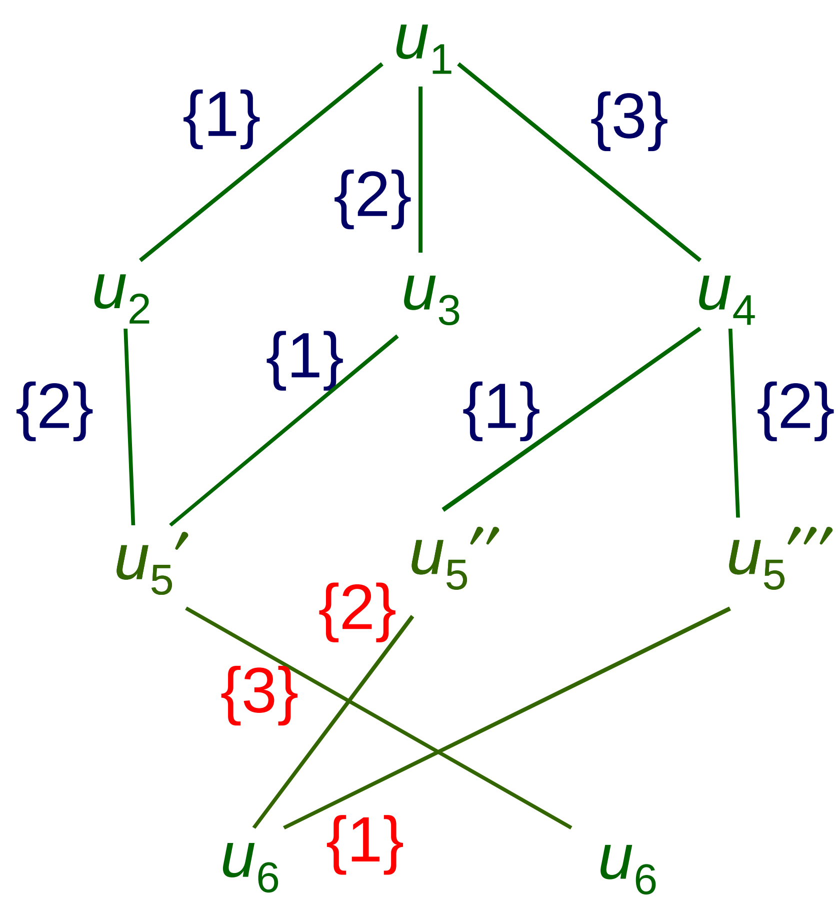
Propagation in MDDs

Filter domains.



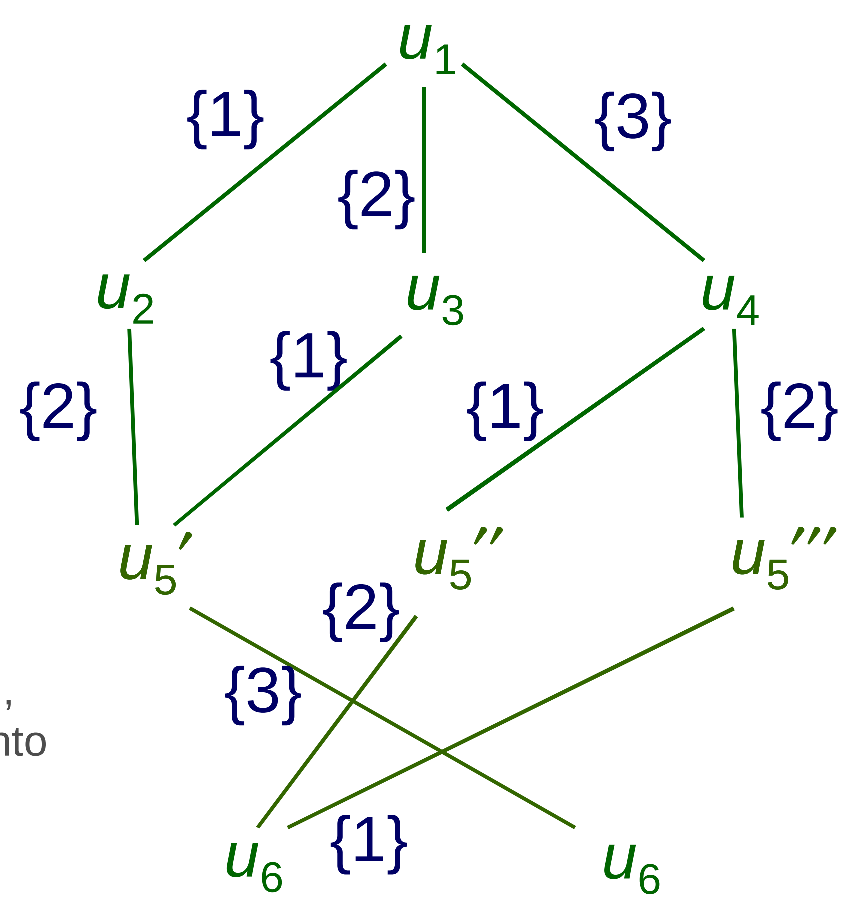
Propagation in MDDs

Filter domains.

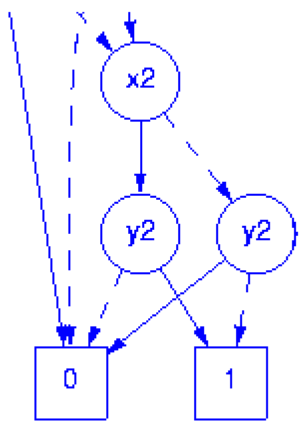


Propagation in MDDs

Alldiff has now
been propagated.



Hadzic, Hooker, O'Sullivan & Tiedemann,
Approximate compilation of constraints into
multivalued decision diagrams, 2008



Computational results

Problems

- Multiple alldiffs.
  - Three overlapping alldiffs.
  - 10-12 variables

$$\text{alldiff}(x_1,\ldots,x_n)$$

- Separable equalities.
  - Three nonlinear equalities in 15 variables.
  - Variable domains contain 3 values.

$$\sum_i g_i(x_i) = \beta$$

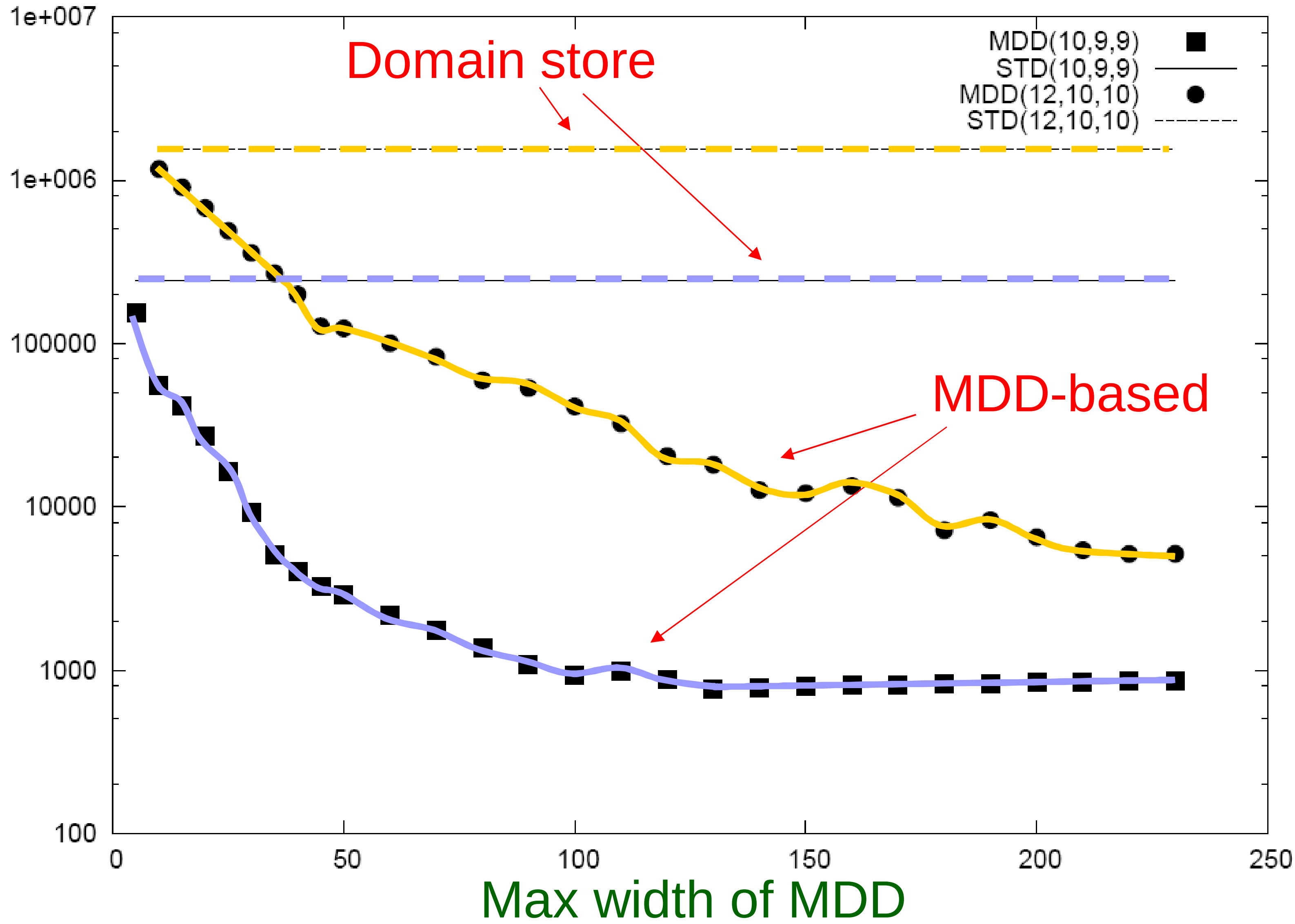
Results – Multiple alldiffs

- Conventional domain store – About a **million** search tree nodes.

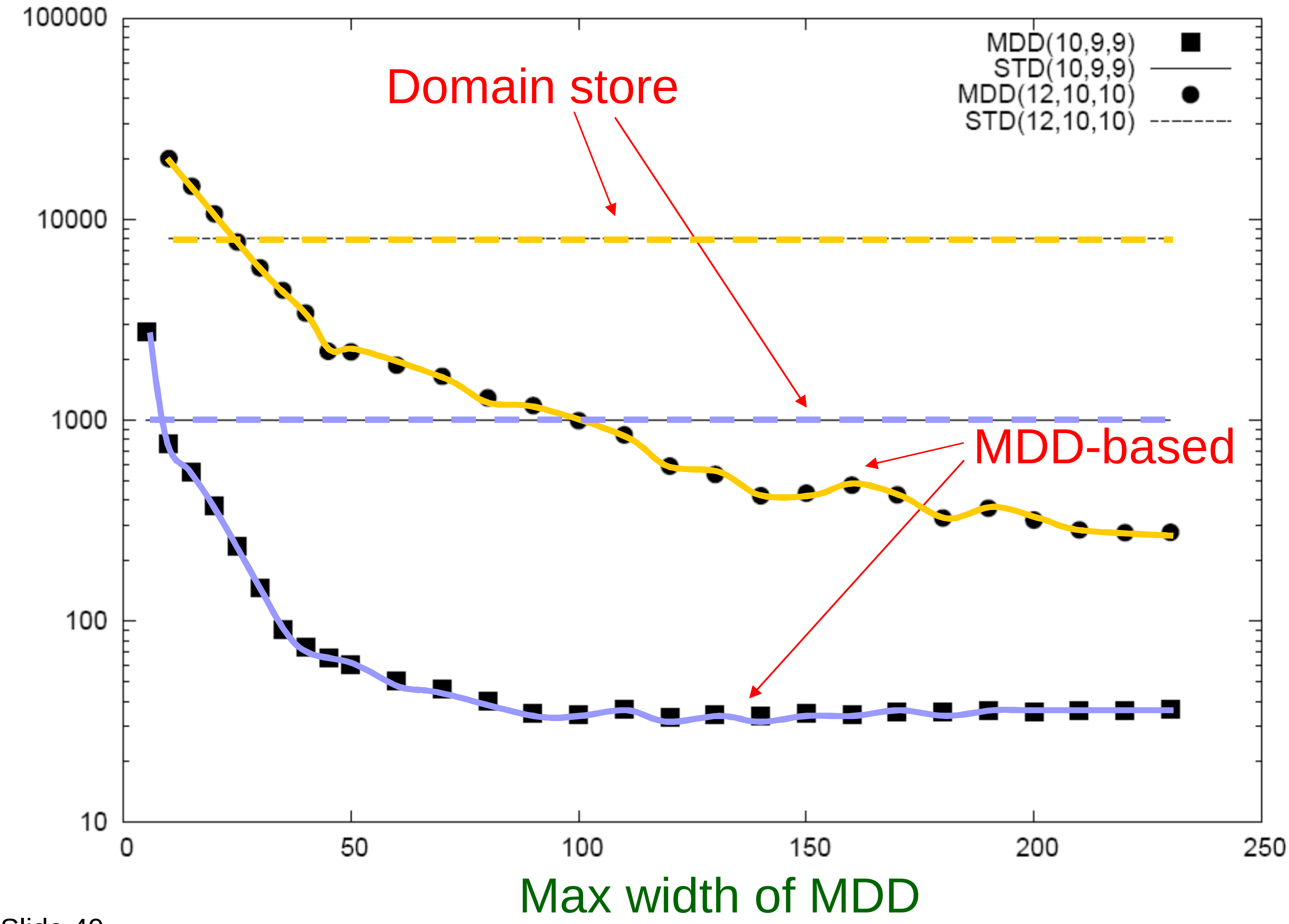
- MDD constraint store **– No backtracking**.
  - Even for width as small as 5.

- **Wider** MDDs require **fewer splits**.

Number of branches + number of splits



Computation time (milliseconds)



- **MDDs are faster** for all but smallest widths.

- Speed advantage of MDDs **increases** with maximum **width**.
  - Up to a factor of about **30**.
  - Within the range of widths studied.

Results – Separable Equalities

- MDD propagation cannot compete with MILP for solving separable equalities alone.

Results – Separable Equalities

- MDD propagation cannot compete with MILP for solving separable equalities alone.

- MDD propagation is designed for problems that are suitable for CP.
  - …but contain some separable equalities.
  - We want effective propagation of the equalities.

Results – Separable Equalities

- MDD propagation cannot compete with MILP for solving separable equalities alone.

- MDD propagation is designed for problems that are suitable for CP.
  - …but contain some separable equalities.
  - We want effective propagation of the equalities.

- We compare MDD propagation for equalities with MDD propagation for 2 inequalities
  - i.e., replacing each equality with 2 inequalities
  - And propagating each inequality separately.

- Filter edge domains with dynamic programming.
  - Applied separately for each equality.
  - Use shortest and longest path lengths to the terminus to prune the calculation.

- Checking for edge equivalence is hard.
  - Two partial solutions

$$(x_1, \ldots, x_k) = (a_1, \ldots, a_k)$$
$$(x_1, \ldots, x_k) = (b_1, \ldots, b_k)$$

are equivalent iff

$$\sum_{i=1}^{k} g_i(a_i) + \sum_{i=k+1}^{n} g_i(x_i) = \beta$$

has same solutions as

$$\sum_{i=1}^{k} g_i(b_i) + \sum_{i=k+1}^{n} g_i(x_i) = \beta$$

- Checking for edge equivalence is hard.
  - Checking whether

$$\sum_{i=1}^{k} g_i(a_i) = \sum_{i=1}^{k} g_i(b_i)$$

    is too strict

- Check for **approximate equivalence**.
  - This is constraint-specific.
  - Allows us to exploit special structure.
  - In the case of equalities, check whether

$$\left| \sum_{i=1}^{k} g_i(a_i) - \sum_{i=1}^{k} g_i(b_i) \right| \leq \Delta$$

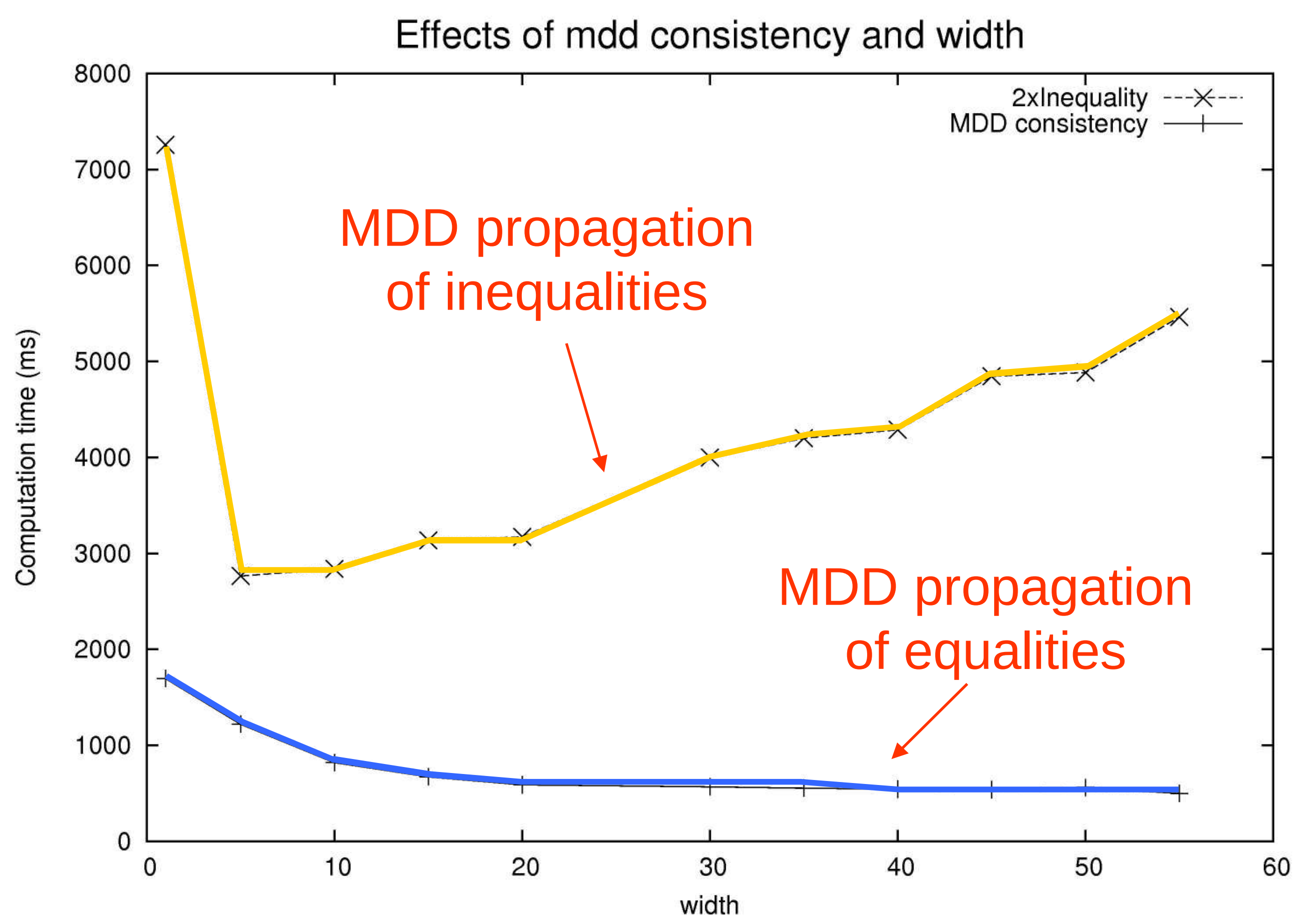
- Actually, we did something else.
  - Split on the nodes u_i for which

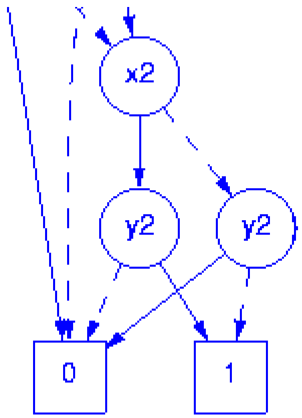
$$\left| L_{\max}(u_i) - L_{\min}(u_i) \right| \le \Delta$$

Longest path from u_i to terminus

Shortest path from u_i to terminus

- MDD propagation of an equality is much faster than propagation of 2 inequalities.
  - Both performance and advantage increase with MDD width.


Effects of mdd consistency and width
Computation time (ms)
0
1000
2000
3000
4000
5000
6000
7000
8000
0
10
20
30
40
50
60
width
2xInequality
MDD consistency
MDD propagation of inequalities
MDD propagation of equalities



Conclusions and research issues

Conclusions

- MDD store provides **substantial advantage** over constraint store in **multiple alldiff** problems.
  - **Wider** MDDs yield greater speedups (factor of 30).
  - **No backtracking**, even with narrow MDDs.

- MDD store is slower than MILP for **separable equalities,** but much faster than using inequality propagator**.**

Conclusions

- **Intensive processing** at search tree nodes can pay off when constraint store is richer.

- Ideally, use **integrated problem solving**
  - Continuous relaxation, cutting planes, etc.
  - MDD propagation, filtering, in parallel.

Research Issues

- Are MDDs superior for **optimization** problems?
  - Optimal solution = shortest path to terminus
- How to adjust **width** of relaxed MDD?
  - Can always set width = 1 if MDDs not useful.
- How to propagate **other global constraints** in an MDD-based constraint store?
  - **Regular** constraint (and special cases, **pattern** & **stretch**)
  - **Sequence** constraint (and special case, **among**)