Implementation and Evaluation of a Compact-Table Propagator in Gecode

Linnea Ingmar
<linnea.ingmar.3244@student.uu.se>

The ASTRA Group
on Combinatorial Optimisation
Uppsala University

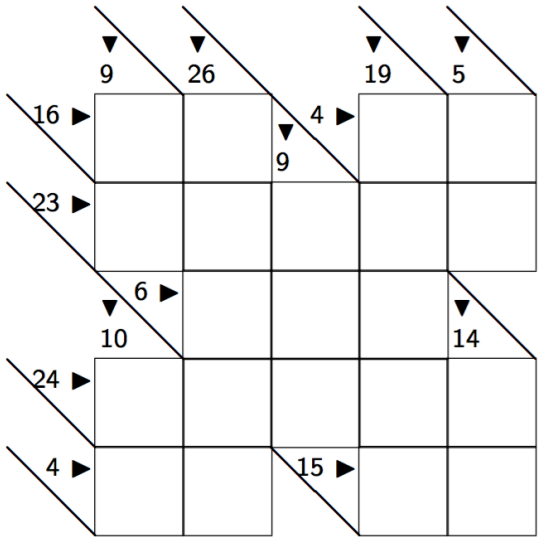
12th June 2017

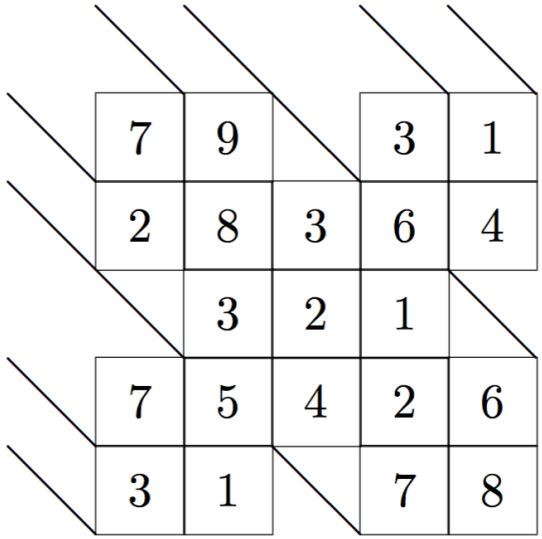
Supervisor: Mats Carlsson
Reviewer: Pierre Flener

Outline

Outline

Kakuro puzzle





Assign the cells digits from 1 to 9 such that for each row and column:

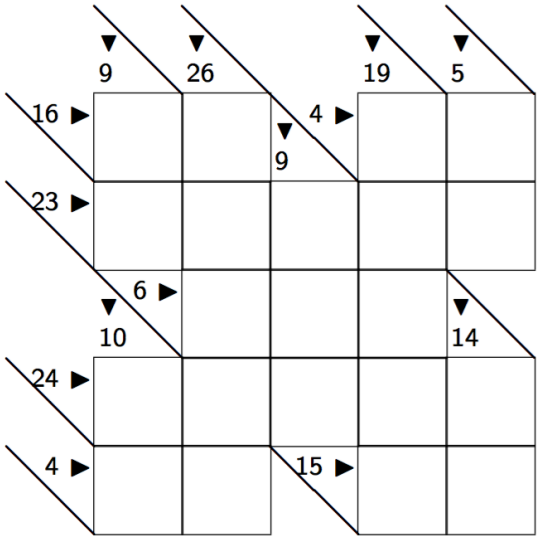
- digits are distinct, and
- the sum of the digits is equal to the *clue*

Kakuro puzzle as a constraint problem (1)

Variables One per cell.

Domains $\{1...9\}$ for all variables.

Constraints For each row and column: *distinct* digits, and the *sum* of the digits is equal to the clue.

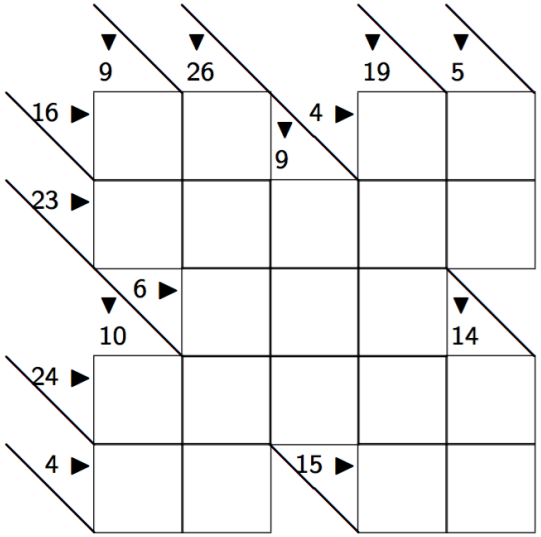


Kakuro puzzle as a constraint problem (2)

Variables One per cell.

Domains $\{1...9\}$ for all variables.

Constraints For each row and column: state the *possible combinations of values* that the variables can take.



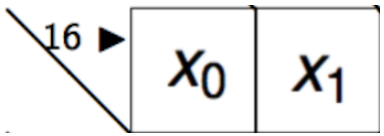
Row/column of size 2 and clue 16: $\langle 7,9 \rangle$ and $\langle 9,7 \rangle$ are the only combinations.

TABLE constraints

Definition (TABLE constraint)

A TABLE constraint lists the possible combinations of values that the variables can take as a sequence of *n*-tuples.

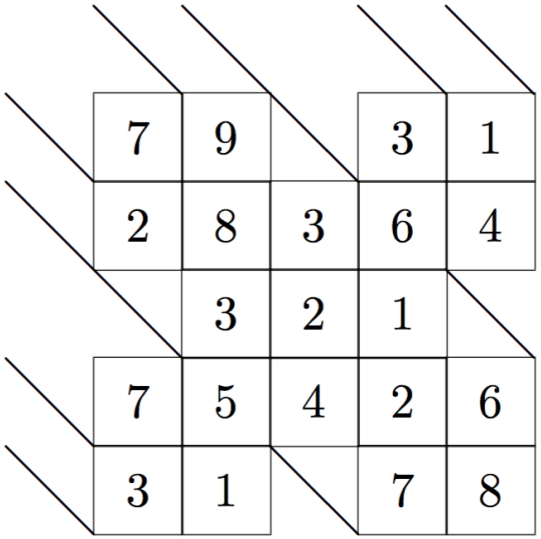
x_0	x_1
7	9
9	7



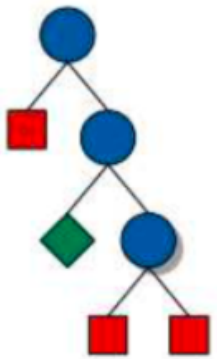
Solving Constraint Problems

Solution A complete variable-value assignment satisfying the constraints.

(Sometimes: maximises/minimises given function.)



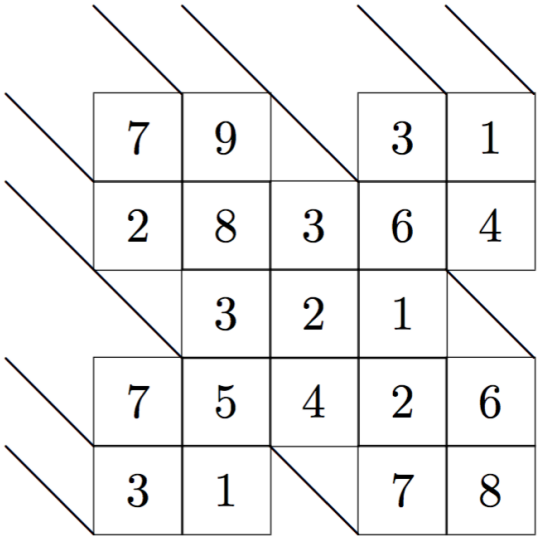
- Solutions are found by **search**
  - Propagation
  - Branching



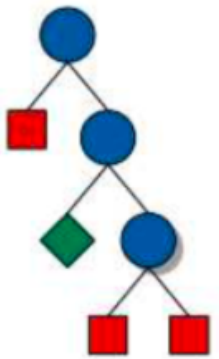
Solving Constraint Problems

Solution A complete variable-value assignment satisfying the constraints.

(Sometimes: maximises/minimises given function.)



- Solutions are found by **search**
  - **Propagation**
  - Branching



UPPSALA UNIVERSITET

Constraint Propagation

Important concepts:

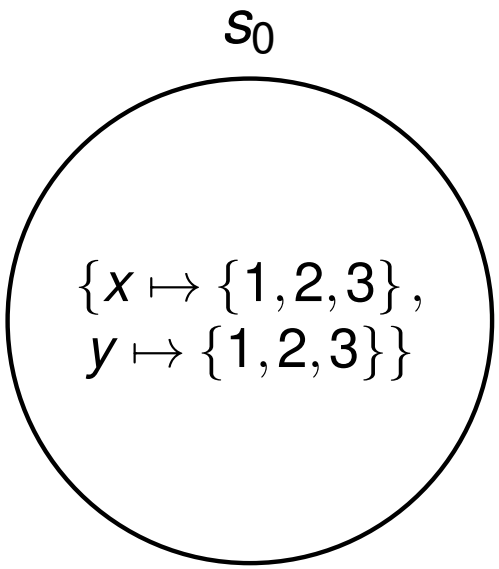
- Constraint store
- Propagator

Constraint Stores

Definition (Constraint store)

A **constraint store** s is a function mapping variables to domains:

$$s : \mathit{variables} \mapsto \mathit{domains}$$



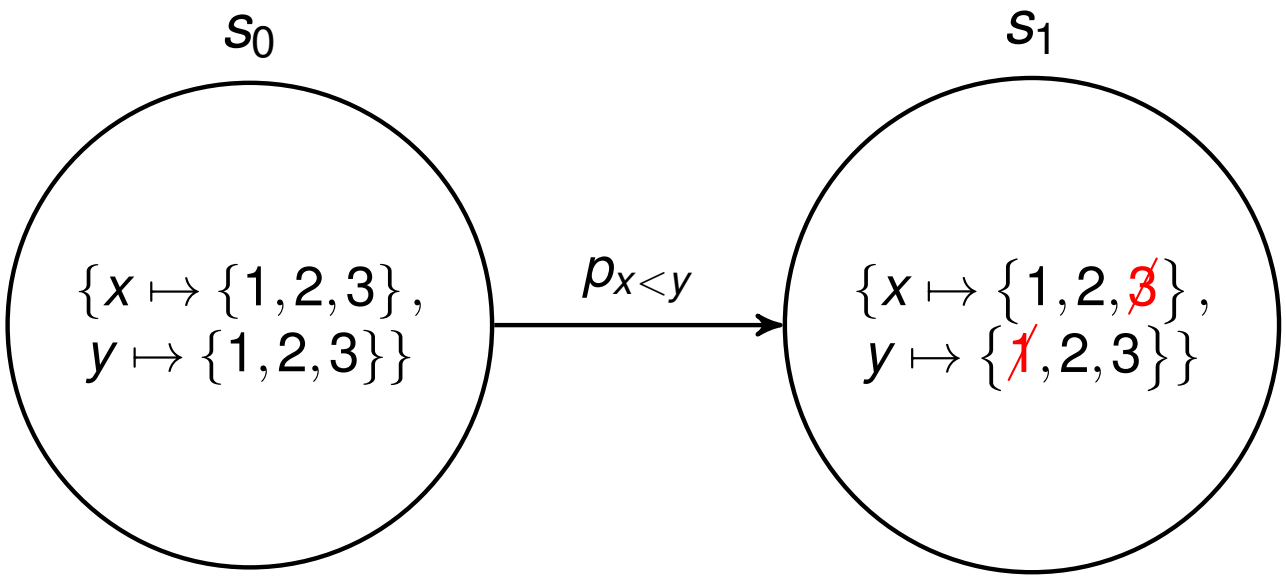
Propagators

Definition (Propagator)

A **propagator** p is a function mapping stores to stores:

$$p : store \mapsto store$$

- Implement constraints



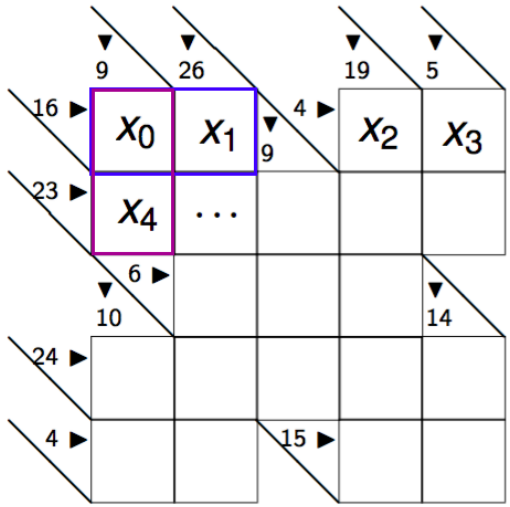
Constraint Propagation

$x_0 \in \{1,2,3,4,5,6,7,8,9\}$
$x_1 \in \{1,2,3,4,5,6,7,8,9\}$
...
$x_4 \in \{1,2,3,4,5,6,7,8,9\}$
...

x_0	x_1
7	9
9	7

x_0	x_4
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1

...



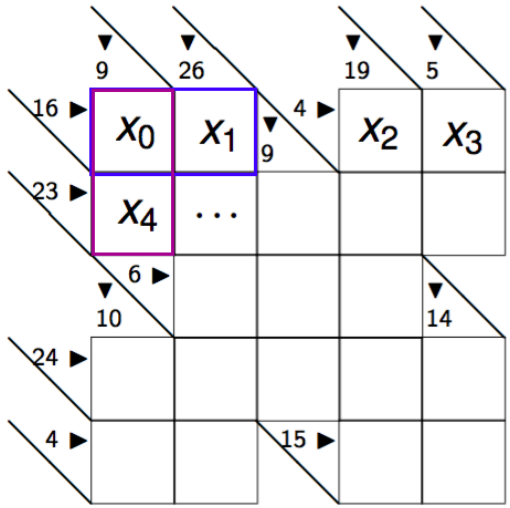
Constraint Propagation

$x_0 \in \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, 7, \cancel{8}, 9\}$
$x_1 \in \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, 7, \cancel{8}, 9\}$
...
$x_4 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$
...

x_0	x_1
7	9
9	7

x_0	x_4
~~1~~	~~8~~
~~2~~	~~7~~
~~3~~	~~6~~
~~4~~	~~5~~
~~5~~	~~4~~
~~6~~	~~3~~
7	2
~~8~~	~~1~~

...



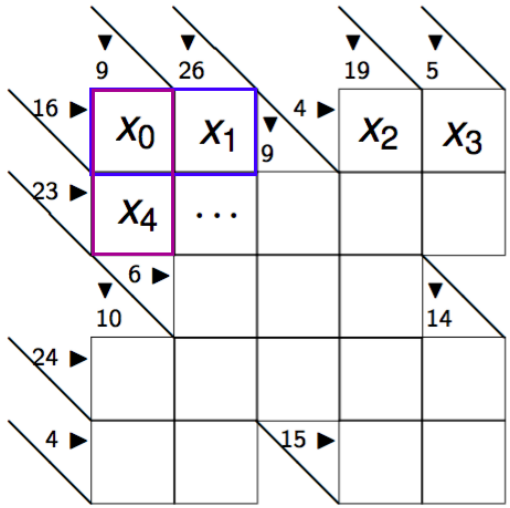
Constraint Propagation

$x_0 \in \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, \mathbf{7}, \cancel{8}, \cancel{9}\}$
$x_1 \in \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, 7, \cancel{8}, 9\}$
...
$x_4 \in \{\cancel{1}, \mathbf{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, \cancel{7}, \cancel{8}, \cancel{9}\}$
...

x_0	x_1
7	9
9	7

x_0	x_4
~~1~~	~~8~~
~~2~~	~~7~~
~~3~~	~~6~~
~~4~~	~~5~~
~~5~~	~~4~~
~~6~~	~~3~~
7	2
~~8~~	~~1~~

...



Constraint Propagation

$x_0 \in \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, \mathbf{7}, \cancel{8}, \cancel{9}\}$

$x_1 \in \{\cancel{1}, \cancel{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, \cancel{7}, \cancel{8}, \mathbf{9}\}$

...

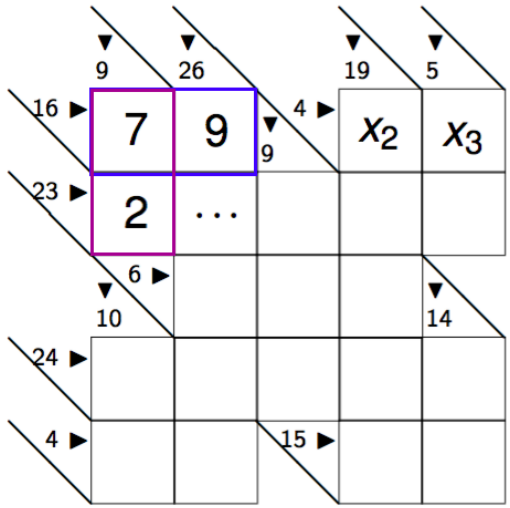
$x_4 \in \{\cancel{1}, \mathbf{2}, \cancel{3}, \cancel{4}, \cancel{5}, \cancel{6}, \cancel{7}, \cancel{8}, \cancel{9}\}$

...

x_0	x_1
7	9
~~9~~	~~7~~

x_0	x_4
~~1~~	~~8~~
~~2~~	~~7~~
~~3~~	~~6~~
~~4~~	~~5~~
~~5~~	~~4~~
~~6~~	~~3~~
7	2
~~8~~	~~1~~

...



Gecode

Gecode (Generic Constraint Development Environment)

- A constraint programming solver (a software that solves constraint problems).
- Written in C++, modular, extensible, and has state-of-the-art performance.
- Supports the programming of new propagators.
- Uses copying during branching, not trailing
- Two existing propagators for the TABLE constraint

generic
constraint
development
environment

Compact Table

Compact-Table

- A new propagation algorithm for the TABLE constraint
- First implemented in OR-tools (uses trailing during branching)
- Published in a 2016 paper **DBLP:conf/cp/DemeulenaereHLP16** – the starting point of this project
- Promising results – outperforms previously known algorithms in the benchmarks in **DBLP:conf/cp/DemeulenaereHLP16**
- Previously no attempt to implement it in Gecode (to the best of my knowledge)

Outline

Initialisation

$$\text{dom}(x_0) = \text{dom}(x_1) = \text{dom}(x_2) = \{1,2,3,4\}$$

x_0	1	2	1	2	6	7	4	1	7	8	2	0	2	5	4
x_1	5	1	3	4	5	7	2	1	8	9	2	0	3	8	3
x_2	8	4	2	2	9	8	1	1	9	6	3	0	1	5	1

Initialisation

$$\text{dom}(x_0) = \text{dom}(x_1) = \text{dom}(x_2) = \{1,2,3,4\}$$

x_0	1	**2**	**1**	**2**	6	7	**4**	**1**	7	8	**2**	0	**2**	5	**4**
x_1	5	**1**	**3**	**4**	5	7	**2**	**1**	8	9	**2**	0	**3**	8	**3**
x_2	8	**4**	**2**	**2**	9	8	**1**	**1**	9	6	**3**	0	**1**	5	**1**

Initialisation

$$\text{dom}(x_0) = \text{dom}(x_1) = \text{dom}(x_2) = \{1,2,3,4\}$$

x_0	**2**	**1**	**2**	**4**	**1**	**2**	**2**	**4**
x_1	**1**	**3**	**4**	**2**	**1**	**2**	**3**	**3**
x_2	**4**	**2**	**2**	**1**	**1**	**3**	**1**	**1**
	0	1	2	3	4	5	6	7

Initialisation

$$\text{dom}(x_0) = \text{dom}(x_1) = \text{dom}(x_2) = \{1,2,3,4\}$$

1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	} validTuples

x_0	**2**	**1**	**2**	**4**	**1**	**2**	**2**	**4**
x_1	**1**	**3**	**4**	**2**	**1**	**2**	**3**	**3**
x_2	**4**	**2**	**2**	**1**	**1**	**3**	**1**	**1**
	0	1	2	3	4	5	6	7

Initialisation

$$\mathrm{dom}(x_0) = \mathrm{dom}(x_1) = \mathrm{dom}(x_2) = \{1,2,3,4\}$$

	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	} validTuples
$\langle x_0, 1\rangle$	0	**1**	0	0	**1**	0	0	0	} supports
$\langle x_0, 2\rangle$	**1**	0	**1**	0	0	**1**	**1**	0	
$\langle x_0, 3\rangle$	0	0	0	0	0	0	0	0	
...									
$\langle x_2, 4\rangle$	**1**	0	0	0	0	0	0	0	

x_0	**2**	**1**	**2**	**4**	**1**	**2**	**2**	**4**
x_1	**1**	**3**	**4**	**2**	**1**	**2**	**3**	**3**
x_2	**4**	**2**	**2**	**1**	**1**	**3**	**1**	**1**
	0	1	2	3	4	5	6	7

Initialisation

$\text{dom}(x_0) = \{1, 2, \cancel{3}, 4\}$
$\text{dom}(x_1) = \text{dom}(x_2) = \{1, 2, 3, 4\}$

	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	} validTuples
$\langle x_0, 1 \rangle$	0	**1**	0	0	**1**	0	0	0	} supports
$\langle x_0, 2 \rangle$	**1**	0	**1**	0	0	**1**	**1**	0	
$\langle x_0, 3 \rangle$	0	0	0	0	0	0	0	0	
...									
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0	

x_0	**2**	**1**	**2**	**4**	**1**	**2**	**2**	**4**
x_1	**1**	**3**	**4**	**2**	**1**	**2**	**3**	**3**
x_2	**4**	**2**	**2**	**1**	**1**	**3**	**1**	**1**
	0	1	2	3	4	5	6	7

Variable Modifications

When a variable x is modified, there are two ways to update `validTuples`:

$|\Delta_x| < |\mathrm{dom}(x)|$ Incremental update

$|\Delta_x| \geq |\mathrm{dom}(x)|$ Reset-based update

Δ_x: set of removed values

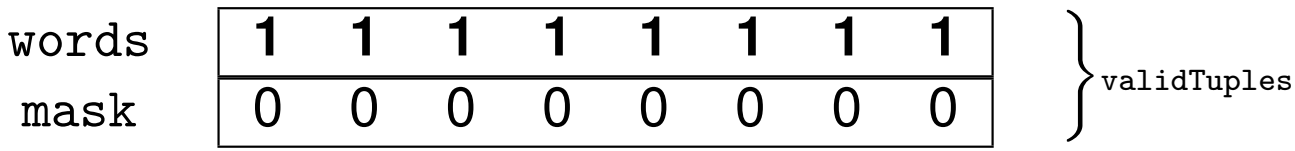
Variable Modifications

Reset-based update

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update



$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	} validTuples
mask	0	0	0	0	0	0	0	0	

supports[x_1,3]	0	**1**	0	0	0	0	**1**	**1**
supports[x_1,4]	0	0	**1**	0	0	0	0	0

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	validTuples
mask	0	0	0	0	0	0	0	0	

supports[x_1,3]	0	**1**	0	0	0	0	**1**	**1**
supports[x_1,4]	0	0	**1**	0	0	0	0	0

$$\texttt{mask} = \texttt{supports}[x_1,3] \mid \texttt{supports}[x_1,4]$$

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	validTuples
mask	0	**1**	**1**	0	0	0	**1**	**1**	

supports[x_1,3]	0	**1**	0	0	0	0	**1**	**1**
supports[x_1,4]	0	0	**1**	0	0	0	0	0

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	validTuples
mask	0	**1**	**1**	0	0	0	**1**	**1**	

supports[x_1,3]	0	**1**	0	0	0	0	**1**	**1**
supports[x_1,4]	0	0	**1**	0	0	0	0	0

words = words & mask

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update

words	0	1	1	0	0	0	1	1
mask	0	1	1	0	0	0	1	1

} validTuples

supports[x_1,3]	0	1	0	0	0	0	1	1
supports[x_1,4]	0	0	1	0	0	0	0	0

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Reset-based update

words	0	**1**	**1**	0	0	0	**1**	**1**
mask	0	**1**	**1**	0	0	0	**1**	**1**

} validTuples

supports[x_1,3]	0	**1**	0	0	0	0	**1**	**1**
supports[x_1,4]	0	0	**1**	0	0	0	0	0

x_0	2	**1**	**2**	4	1	2	**2**	**4**
x_1	1	**3**	**4**	2	1	2	**3**	**3**
x_2	4	**2**	**2**	1	1	3	**1**	**1**

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

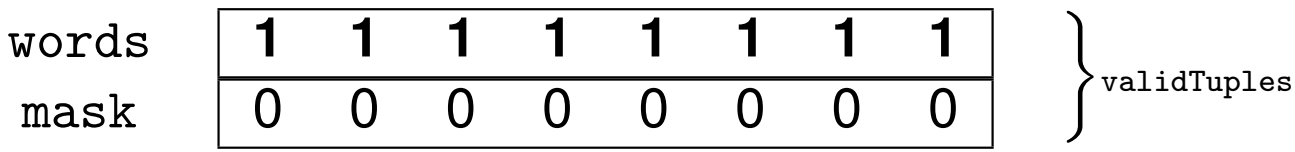
Variable Modifications

Incremental update

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update



$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	} validTuples
mask	0	0	0	0	0	0	0	0	

supports[x_1,1]	**1**	0	0	0	**1**	0	0	0
supports[x_1,2]	0	0	0	**1**	0	**1**	0	0

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	validTuples
mask	0	0	0	0	0	0	0	0	

supports[x_1,1]	**1**	0	0	0	**1**	0	0	0
supports[x_1,2]	0	0	0	**1**	0	**1**	0	0

$$\texttt{mask} = \texttt{supports}[x_1,1] \mid \texttt{supports}[x_1,2]$$

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	validTuples
mask	**1**	0	0	**1**	**1**	**1**	0	0	

supports[x_1,1]	**1**	0	0	0	**1**	0	0	0
supports[x_1,2]	0	0	0	**1**	0	**1**	0	0

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update

words	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	} validTuples
mask	**1**	0	0	**1**	**1**	**1**	0	0	

supports[x_1,1]	**1**	0	0	0	**1**	0	0	0
supports[x_1,2]	0	0	0	**1**	0	**1**	0	0

words = words & !mask

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update

words	0	**1**	**1**	0	0	0	**1**	**1**	} validTuples
mask	**1**	0	0	**1**	**1**	**1**	0	0	

supports[x_1,1]	**1**	0	0	0	**1**	0	0	0
supports[x_1,2]	0	0	0	**1**	0	**1**	0	0

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Variable Modifications

Incremental update

words	0	**1**	**1**	0	0	0	**1**	**1**
mask	**1**	0	0	**1**	**1**	**1**	0	0

} validTuples

supports[x_1,1]	**1**	0	0	0	**1**	0	0	0
supports[x_1,2]	0	0	0	**1**	0	**1**	0	0

x_0	2	**1**	**2**	4	1	2	**2**	**4**
x_1	1	**3**	**4**	2	1	2	**3**	**3**
x_2	4	**2**	**2**	1	1	3	**1**	**1**

$$\text{dom}(x_0) = \{1,2,4\}$$
$$\text{dom}(x_1) = \{\cancel{1},\cancel{2},3,4\}$$
$$\text{dom}(x_2) = \{1,2,3,4\}$$

Filtering

- Intersect every support entry with `validTuples`
- Remove value if intersection is empty

`validTuples`:

`words`	0	**1**	**1**	0	0	0	**1**	**1**

&

$\langle x_2, 1 \rangle$	0	0	0	**1**	**1**	0	**1**	**1**
$\langle x_2, 2 \rangle$	0	**1**	**1**	0	0	0	0	0
$\langle x_2, 3 \rangle$	0	0	0	0	0	**1**	0	0
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0

$$\text{dom}(x_2) = \{1, 2, 3, 4\}$$

Filtering

- Intersect every support entry with `validTuples`
- Remove value if intersection is empty

`validTuples`:

`words`	0	**1**	**1**	0	0	0	**1**	**1**

&

$\langle x_2, 1 \rangle$	0	0	0	1	1	0	1	1
$\langle x_2, 2 \rangle$	0	**1**	**1**	0	0	0	0	0
$\langle x_2, 3 \rangle$	0	0	0	0	0	**1**	0	0
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0

$$\text{dom}(x_2) = \{1, 2, 3, 4\}$$

Filtering

- Intersect every support entry with `validTuples`
- Remove value if intersection is empty

`validTuples`:

`words`	0	**1**	**1**	0	0	0	**1**	**1**

&

$\langle x_2, 1 \rangle$	0	0	0	**1**	**1**	0	**1**	**1**
$\langle x_2, 2 \rangle$	0	**1**	**1**	0	0	0	0	0
$\langle x_2, 3 \rangle$	0	0	0	0	0	**1**	0	0
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0

$$\text{dom}(x_2) = \{1, 2, 3, 4\}$$

Filtering

- Intersect every support entry with `validTuples`
- Remove value if intersection is empty

`validTuples`:

`words`	0	**1**	**1**	0	0	0	**1**	**1**

&

$\langle x_2, 1 \rangle$	0	0	0	**1**	**1**	0	**1**	**1**
$\langle x_2, 2 \rangle$	0	**1**	**1**	0	0	0	0	0
$\langle x_2, 3 \rangle$	0	0	0	0	0	**1**	0	0
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0

$$\text{dom}(x_2) = \{1, 2, 3, 4\}$$

Filtering

- Intersect every support entry with `validTuples`
- Remove value if intersection is empty

`validTuples`:

`words`	0	**1**	**1**	0	0	0	**1**	**1**

&

$\langle x_2, 1 \rangle$	0	0	0	**1**	**1**	0	**1**	**1**
$\langle x_2, 2 \rangle$	0	**1**	**1**	0	0	0	0	0
$\langle x_2, 3 \rangle$	0	0	0	0	0	**1**	0	0
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0

$$\text{dom}(x_2) = \{1, 2, 3, 4\}$$

Filtering

- Intersect every support entry with `validTuples`
- Remove value if intersection is empty

`validTuples`:

`words`	0	**1**	**1**	0	0	0	**1**	**1**

&

$\langle x_2, 1 \rangle$	0	0	0	**1**	**1**	0	**1**	**1**
$\langle x_2, 2 \rangle$	0	**1**	**1**	0	0	0	0	0
$\langle x_2, 3 \rangle$	0	0	0	0	0	**1**	0	0
$\langle x_2, 4 \rangle$	**1**	0	0	0	0	0	0	0

$$\mathrm{dom}(x_2) = \{1, 2, \cancel{3}, \cancel{4}\}$$

Sparse bit-set

- Bit-wise operations are performed on non-zero words only

validTuples:

	0	1	2	3
words =	**0 1 0 0**	**0 0 1 0**	**1 0 0 0**	**1 0 0 1**
mask =	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
index =	[**0**, **1**, **2**, **3**]			
limit =	3			

Sparse bit-set

- Bit-wise operations are performed on non-zero words only

validTuples:

	0	1	2	3
words =	**0 1 0 0**	**0 0 1 0**	0 0 0 0	**1 0 0 1**
mask =	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
index =	[**0**, **1**, **3**, 2]			
limit =	2			

Outline

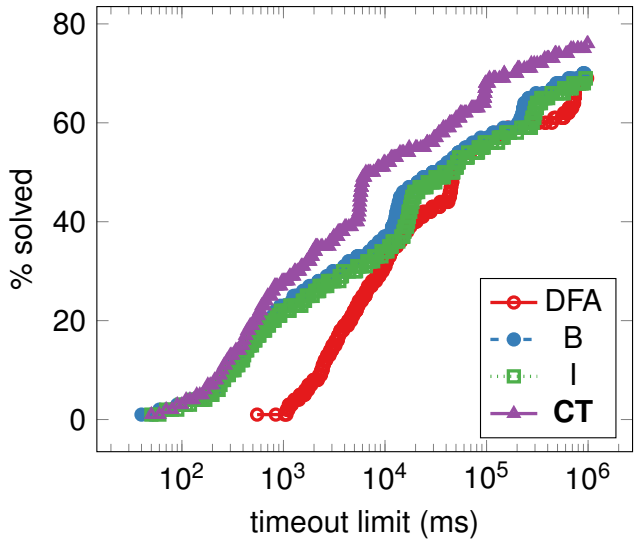
Experiments

- Compared CT against
  - 2 existing propagators for TABLE
  - 1 propagator for REGULAR
- Same benchmark set that was used in **DBLP:conf/cp/DemeulenaereHLP16** (translated to MiniZinc)
  - 1507 instances over 30 series
  - Table sizes: $1 - 1.6 \cdot 10^6$ tuples
  - Arities: $2 - 20$ variables
  - Domain sizes: $1 - 1000$
  - Available at `https://bitbucket.org/pschaus/xp-table` (15 GB of uncompressed files)

Annotations

CT	Compact-Table propagator
DFA	Layered graph (DFA) propagator
B	Basic tuple set propagator
I	Incremental tuple set propagator

All instances

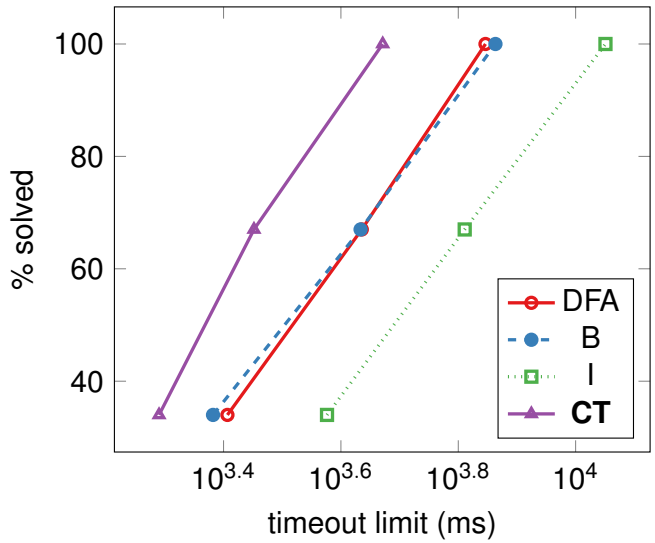


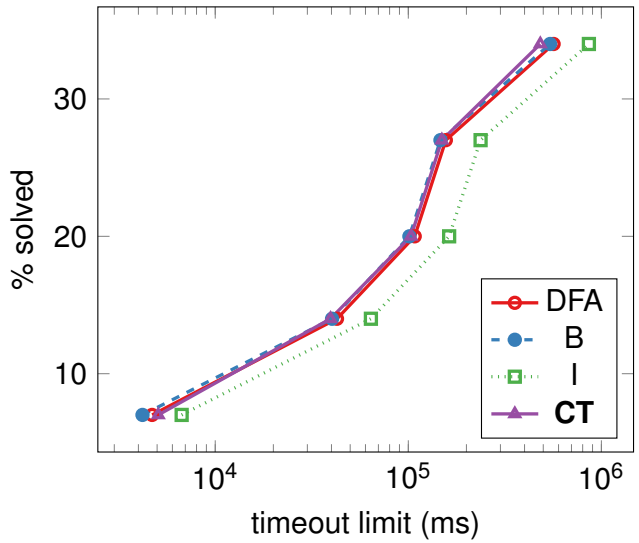
661 instances after filtering out those where

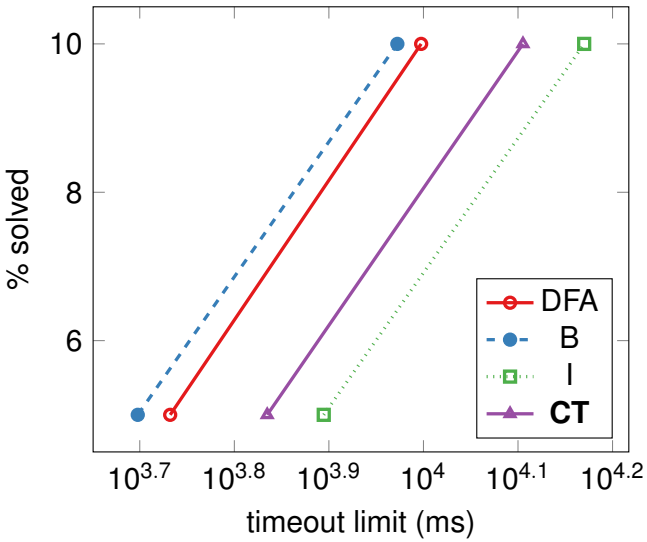
- the runtime was < 1 s for all algorithm (639 instances), or
- at least one algorithm ran out of memory (207 instances).

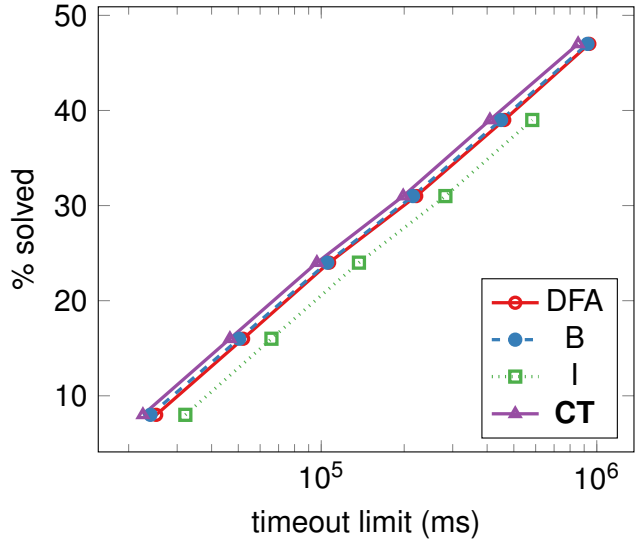
Small tables

3 – 7 tuples



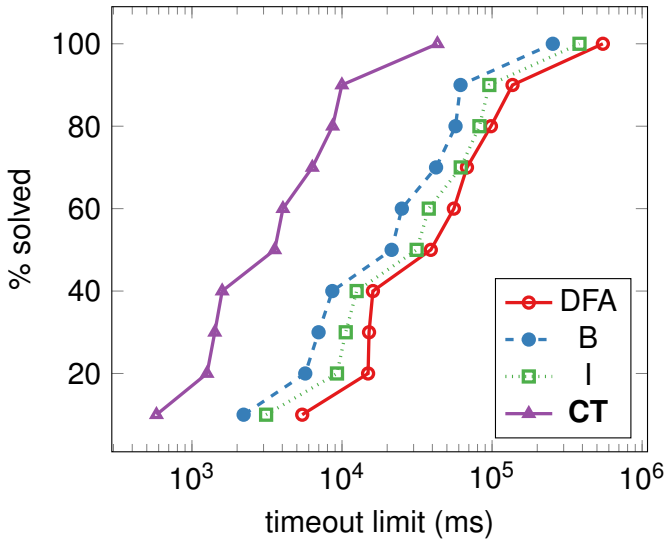


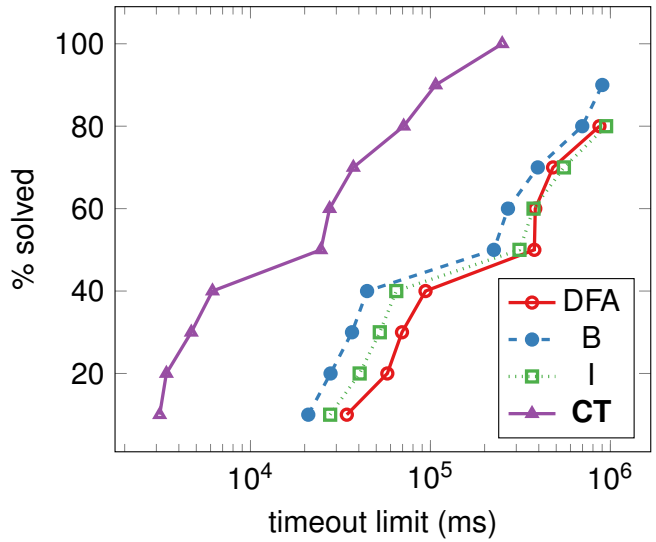


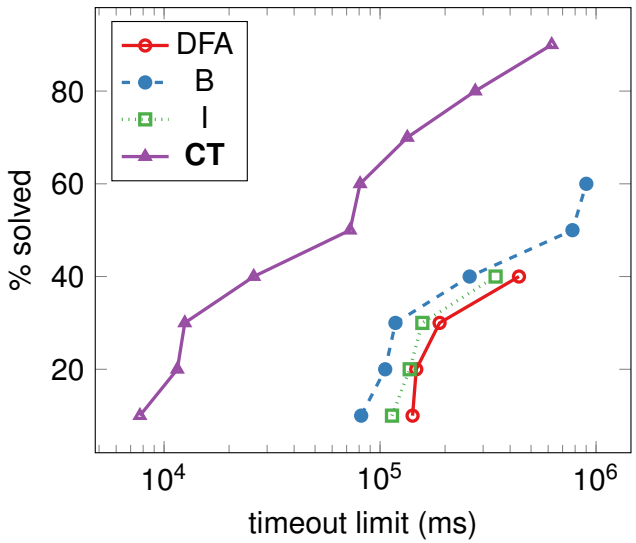


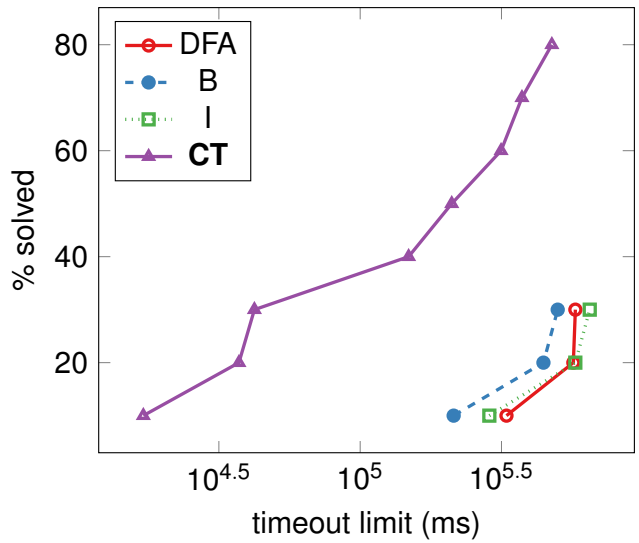
Large tables

2500 – 10000 **tuples**



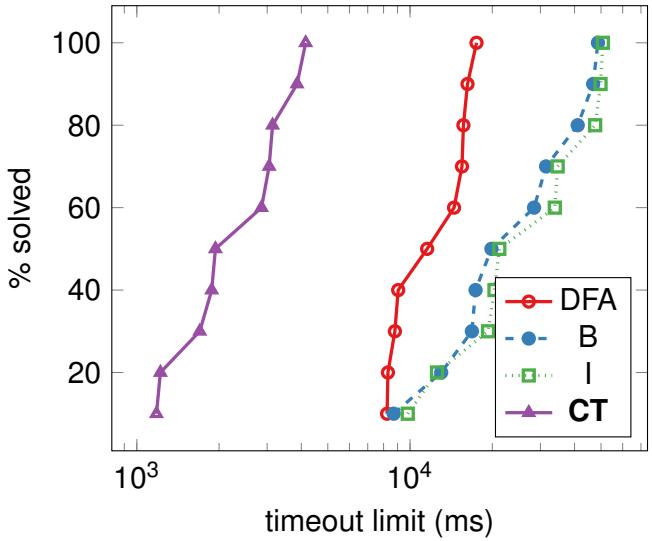


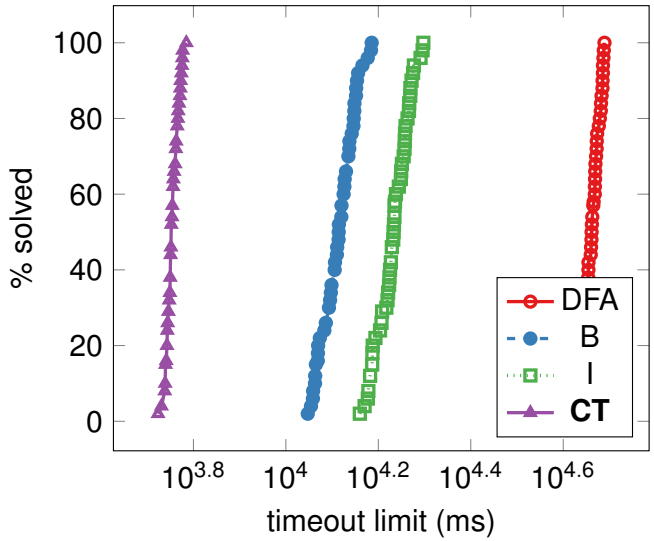


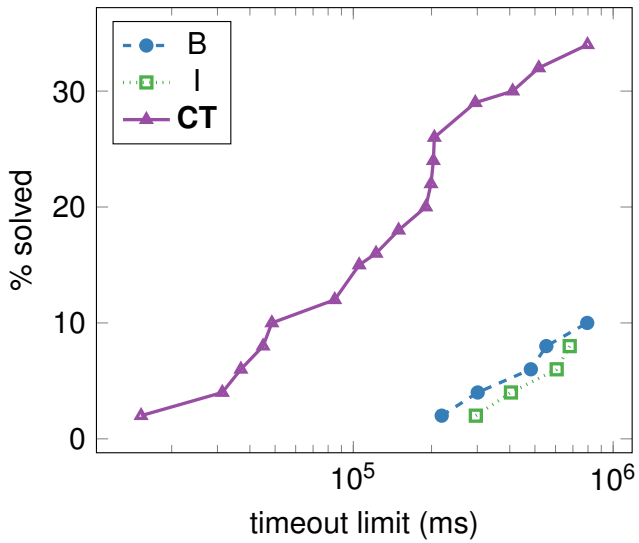


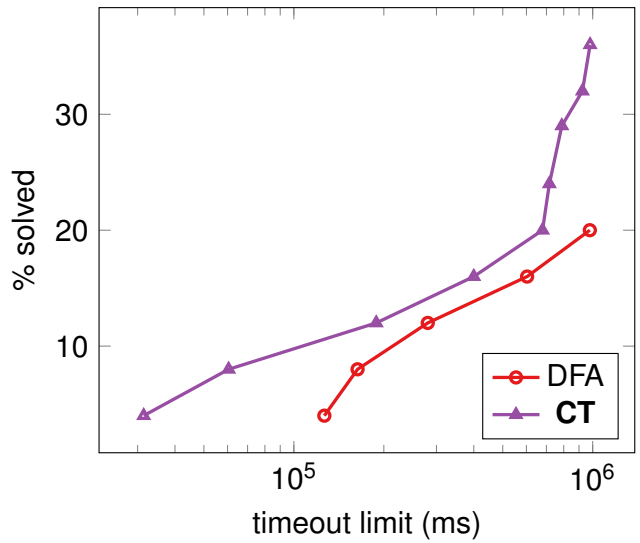
Large tables

> 12000 tuples



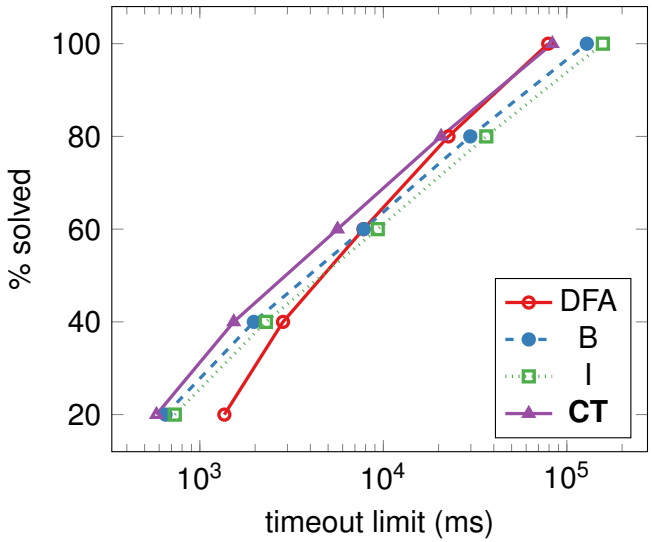


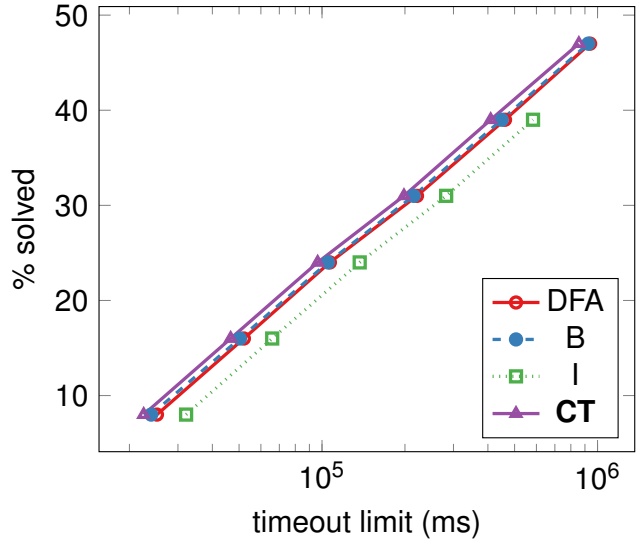


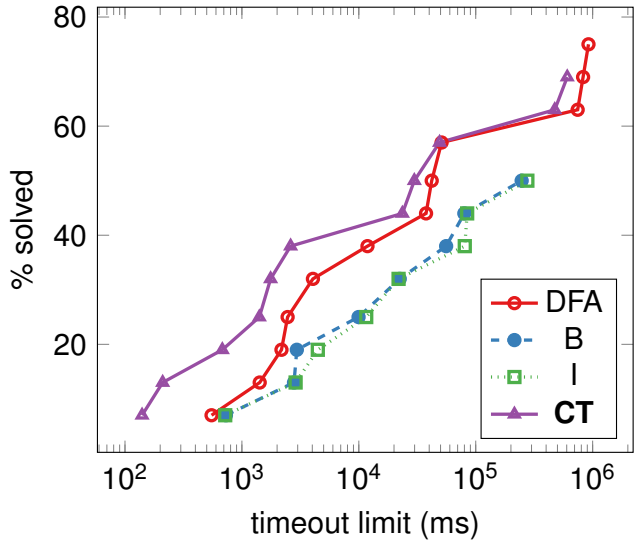


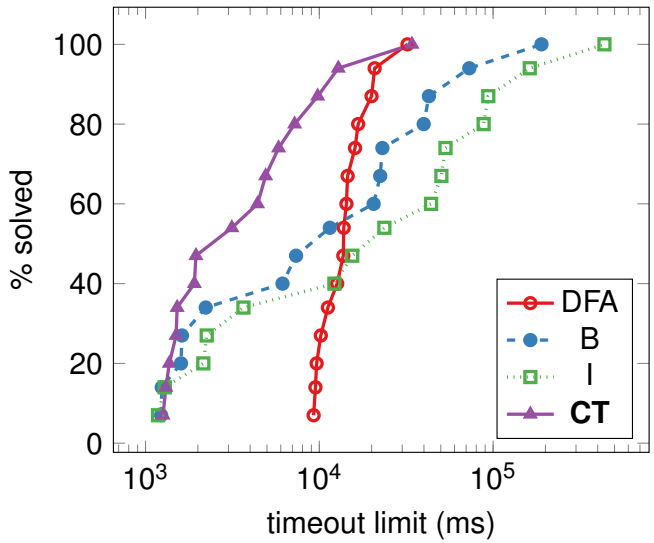
Low arities

2 – 3 variables



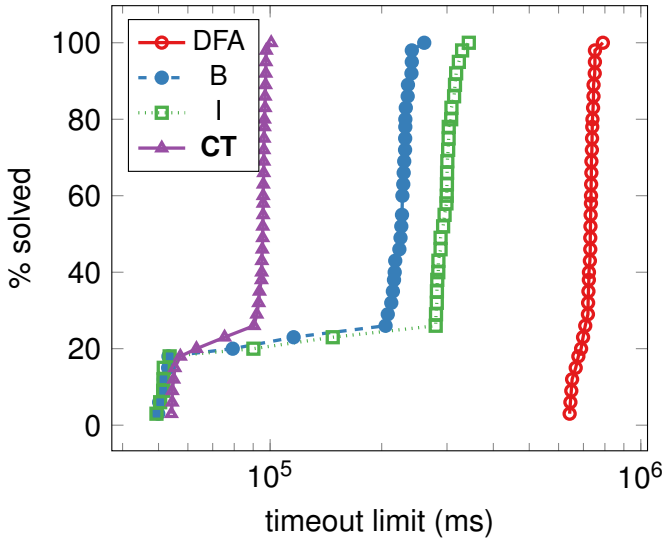


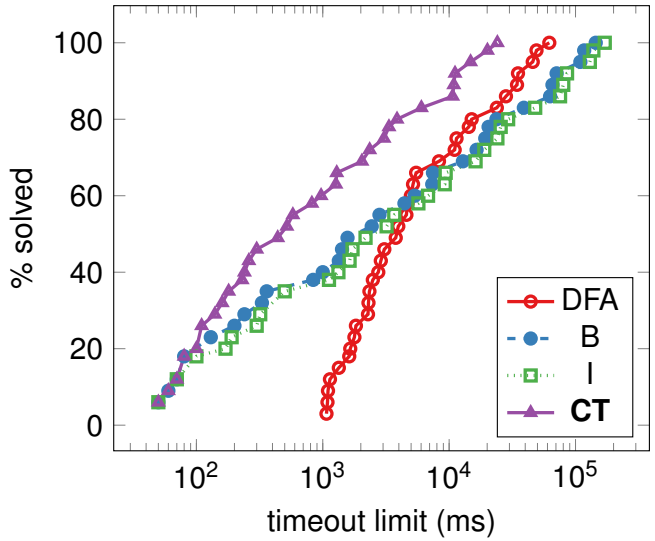


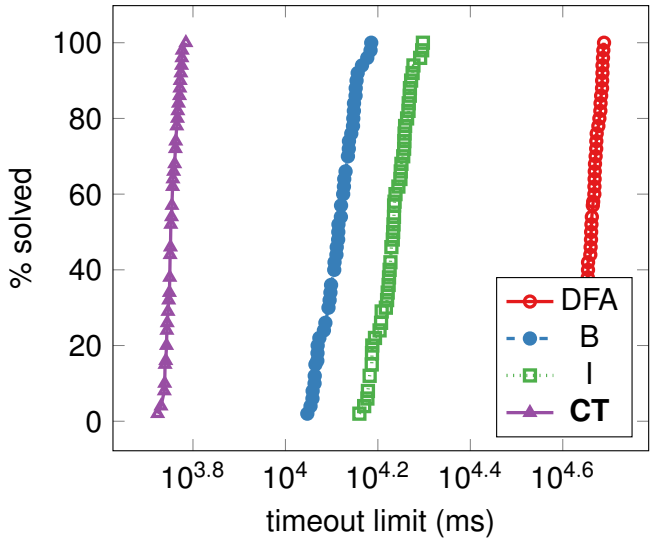


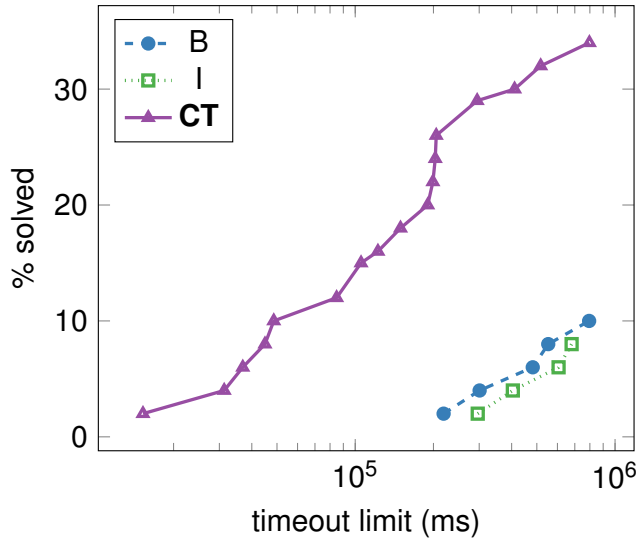
High arities

> 5 variables



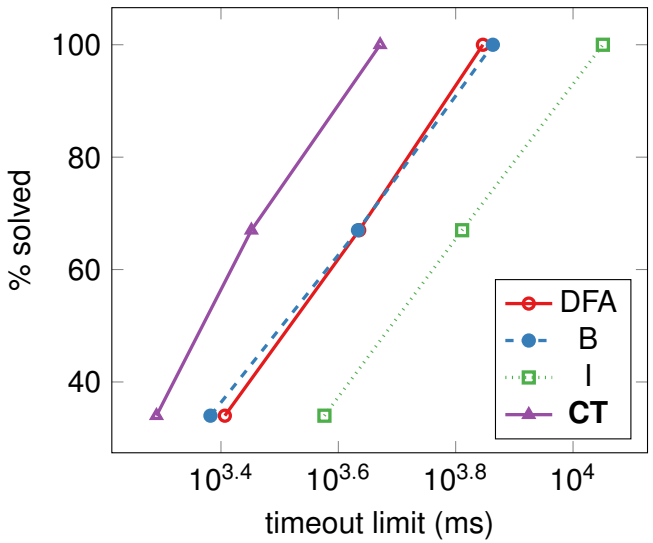


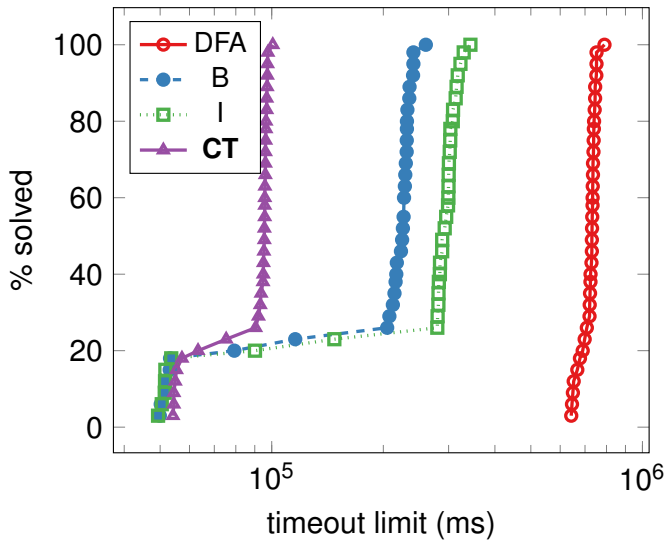




Small domains

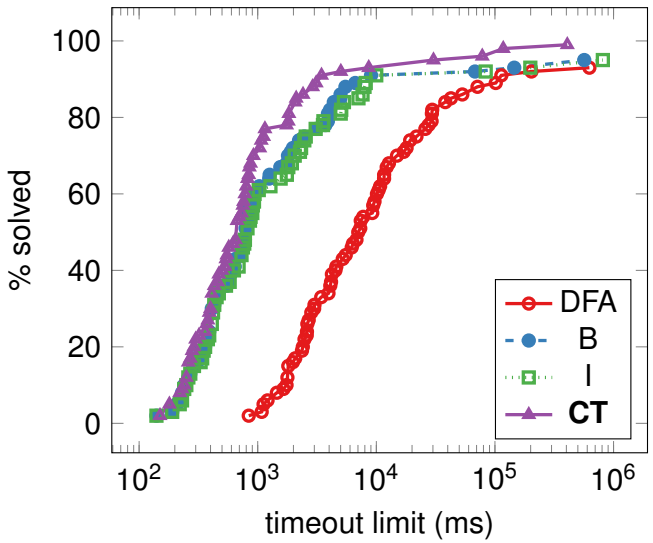
Domain size 2

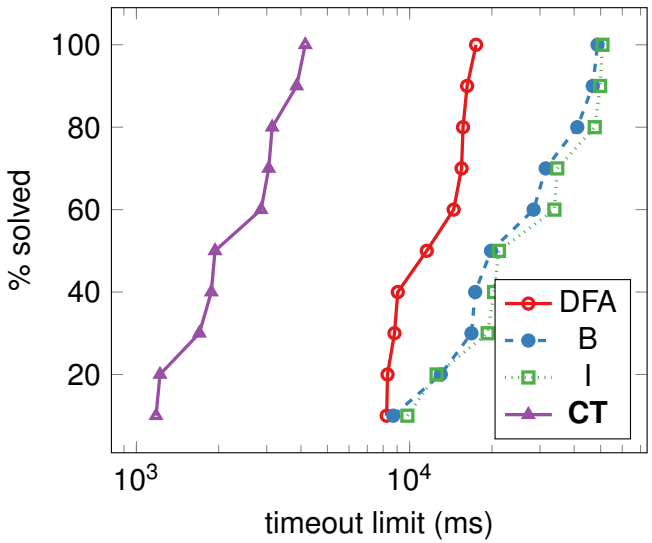




Large domains

Domain size ≥ 10





Results

- CT seems to outperfom the other propagators
- CT has largest performance gains on series with large tables
- DFA overall slowest

Outline

Summary and Conclusions

- What I have done:
  - Implemented Compact-Table (CT) in Gecode
  - Compared CT against existing propagators for TABLE, and with REGULAR
- The results:
  - CT seems to outperform existing propagators for TABLE
- Future work:
  - Proof of concept – not a final version
  - Inclusion into Gecode?
  - Implement and evaluate generalisations of CT described in **DBLP:conf/aaai/VerhaegheLS17**

UPPSALA UNIVERSITET

References