

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

18th May 2017

Supervisor: Mats Carlsson
Reviewer: Pierre Flener



Outline

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

1

Background

2

The Compact-Table Algorithm

3

Evaluation

4

Summary and Conclusions



Outline

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

1 Background

2 The Compact-Table Algorithm

3 Evaluation

4 Summary and Conclusions



Kakuro puzzle

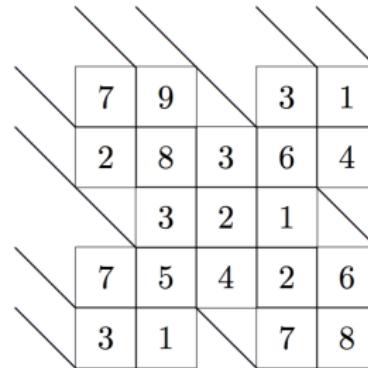
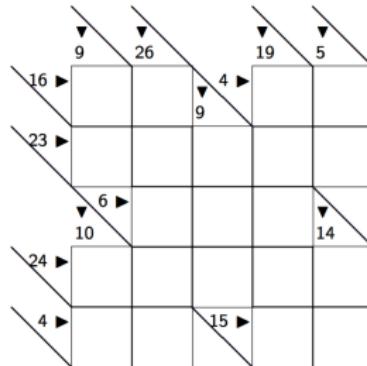
Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References



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)

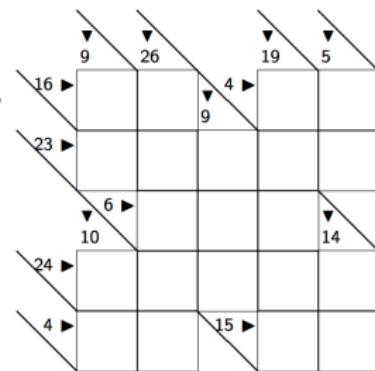
Background

The
Compact-
Table
Algorithm
Evaluation
Summary
and
Conclusions
References

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)

Background

The
Compact-
Table
Algorithm

Evaluation

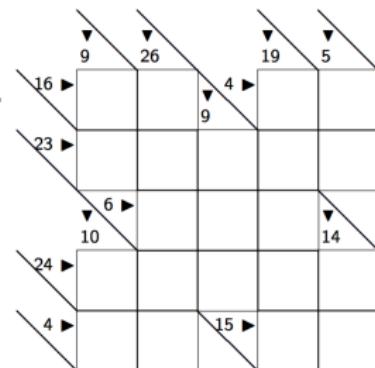
Summary
and
Conclusions

References

Variables One per cell.

Domains $\{1 \dots 9\}$ 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

Background

The
Compact-
Table
Algorithm

Evaluation

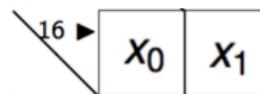
Summary
and
Conclusions

References

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

Background

The
Compact-
Table
Algorithm

Evaluation

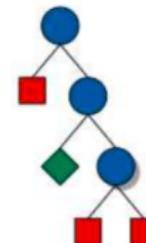
Summary
and
Conclusions

References

Solution A complete variable-value assignment satisfying the constraints.
(Sometimes: maximises/minimises given function.)

7	9		3	1
2	8	3	6	4
	3	2	1	
7	5	4	2	6
3	1		7	8

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





Solving Constraint Problems

Background

The
Compact-
Table
Algorithm

Evaluation

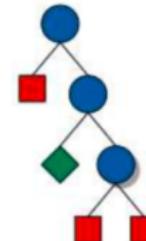
Summary
and
Conclusions

References

Solution A complete variable-value assignment satisfying the constraints.
(Sometimes: maximises/minimises given function.)

7	9		3	1
2	8	3	6	4
	3	2	1	
7	5	4	2	6
3	1		7	8

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





Constraint Propagation

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

Important concepts:

- Constraint store
- Propagator



Constraint Stores

Background

The
Compact-
Table
Algorithm

Evaluation

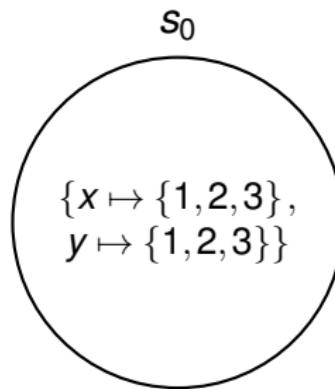
Summary
and
Conclusions

References

Definition (Constraint store)

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

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



Propagators

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

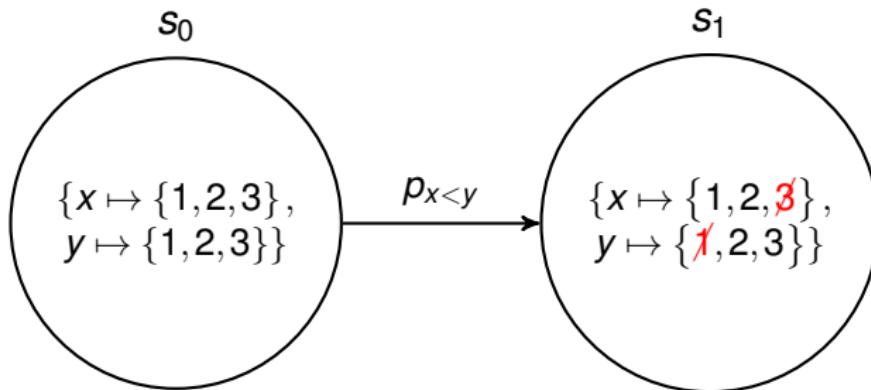
References

Definition (Propagator)

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

$$p : \text{store} \mapsto \text{store}$$

■ Implement constraints





Constraint Propagation

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

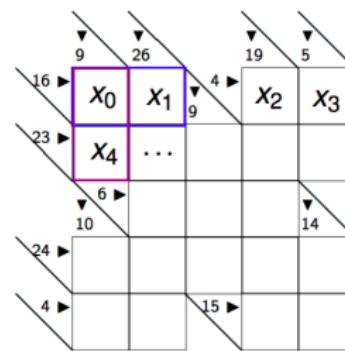
$$x_1 \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

...

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

...

x_0	x_4
1	8
x_0	x_1
2	7
3	6
4	5
5	4
6	3
7	2
8	1





Constraint Propagation

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

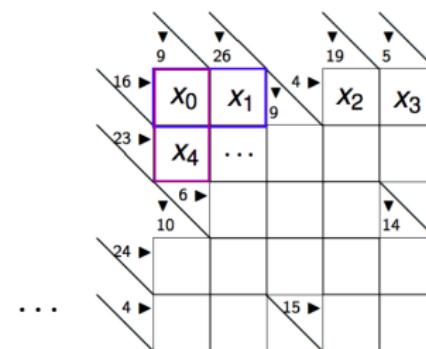
...

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

...

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

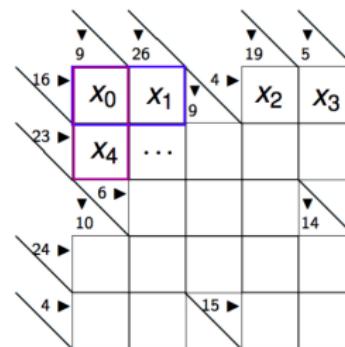
...

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

...

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

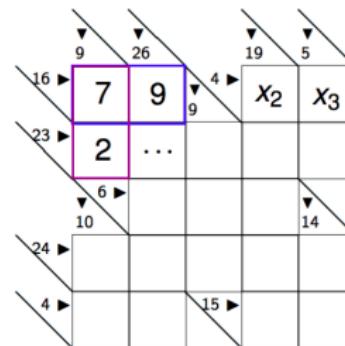
...

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

...

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





Gecode

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

Gecode (Generic Constraint Development Environment)

- A constraint 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





Compact Table

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References





Compact-Table

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- A new propagation algorithm for the TABLE constraint
- First implemented in OR-tools
- Published in a 2016 paper [1] – the starting point of this project
- Promising results - outperforms previously known algorithms in the benchmarks in [1]
- No attempt to implement it in Gecode (until now)



Outline

1

Background

2

The Compact-Table Algorithm

3

Evaluation

4

Summary and Conclusions



Initialisation

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

 $\} \text{validTuples}$

$\langle x_0, 1 \rangle$	0	1	0	0	1	0	0	0
$\langle x_0, 2 \rangle$	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

$\} \text{supports}$

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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

1	1	1	1	1	1	1	1
---	---	---	---	---	---	---	---

 } validTuples

$\langle x_0, 1 \rangle$	0	1	0	0	1	0	0	0
$\langle x_0, 2 \rangle$	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

} supports

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

Background

The
Compact-
Table
Algorithm

Evaluation
Summary
and
Conclusions

References

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

$|\Delta_x| < |\text{dom}(x)|$ Reset-based update

$|\Delta_x| \geq |\text{dom}(x)|$ Incremental update

Δ_x : set of removed values



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation
Summary
and
Conclusions
References

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

} validTuples

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

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

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

} 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

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

} 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

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

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

$$\text{dom}(x_1) = \{\mathcal{X}, \mathcal{Z}, 3, 4\}$$

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words	1 1 1 1 1 1 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

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words	1	1	1	1	1	1	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

words = words & mask

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

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

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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



Variable Modifications

Reset-based update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words

0	1	1	0	0	0	1	1
0	1	1	0	0	0	1	1

{ validTuples }

mask

supports[$x_1, 3]$

0	1	0	0	0	0	1	1
0	0	1	0	0	0	0	0

supports[$x_1, 4]$ x_0

2	1	2	4	1	2	2	4
1	3	4	2	1	2	3	3
4	2	2	1	1	3	1	1

 x_1 x_2

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

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

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation
Summary
and
Conclusions
References

$$\begin{aligned}\text{dom}(x_0) &= \{1, 2, 4\} \\ \text{dom}(x_1) &= \{\mathcal{X}, \mathcal{Z}, 3, 4\} \\ \text{dom}(x_2) &= \{1, 2, 3, 4\}\end{aligned}$$



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

} validTuples

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

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

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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

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

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words	<table border="1"><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	} validTuples
1	1	1	1	1	1	1	1											
0	0	0	0	0	0	0	0											
mask	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0									
0	0	0	0	0	0	0	0											

supports[$x_1, 1$]	<table border="1"><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	1	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0
1	0	0	0	1	0	0	0										
0	0	0	1	0	1	0	0										
supports[$x_1, 2$]	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	0	0	0	1	0	1	0	0								
0	0	0	1	0	1	0	0										

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

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

$$\text{dom}(x_1) = \{\mathcal{X}, \mathcal{Z}, 3, 4\}$$

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words	1 1 1 1 1 1 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

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words	1 1 1 1 1 1 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

words = words & !mask

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

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

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

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



Variable Modifications

Incremental update

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

words

mask

0	1	1	0	0	0	1	1
1	0	0	1	1	1	0	0

{ validTuples }

supports[$x_1, 1$]supports[$x_1, 2$]

1	0	0	0	1	0	0	0
0	0	0	1	0	1	0	0

 x_0 x_1 x_2

2	1	2	4	1	2	2	4
1	3	4	2	1	2	3	3
4	2	2	1	1	3	1	1

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

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

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



Filtering

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- 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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- 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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- 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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- 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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- 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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- 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

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



Sparse bit-set

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- Bit-wise operations are performed on nonzero-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 =	4			



Sparse bit-set

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- Bit-wise operations are performed on nonzero-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 =	3			



Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

Outline

1

Background

2

The Compact-Table Algorithm

3

Evaluation

4

Summary and Conclusions



Experiments

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- Compared CT against
 - 2 previously existing propagators for TABLE
 - 1 propagator for REGULAR
- Same benchmark set that was used in [1] (compiled 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

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

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

Background

The
Compact-
Table
Algorithm

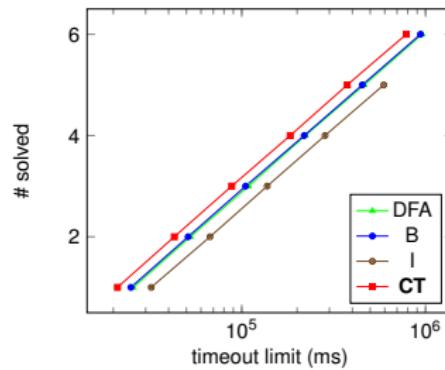
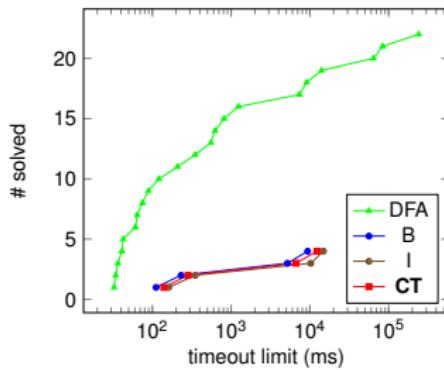
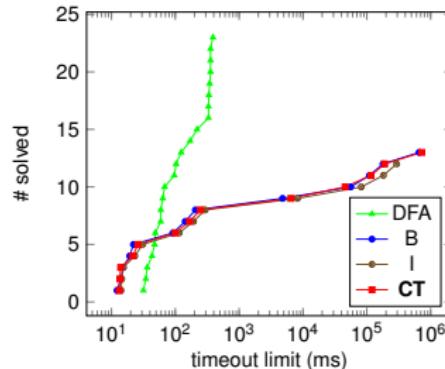
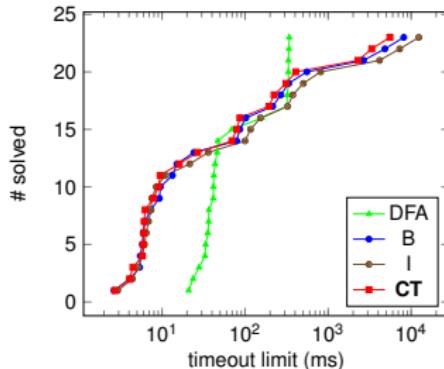
Evaluation

Summary
and
Conclusions

References

Small tables

3 – 7 tuples



Background

The
Compact-
Table
Algorithm

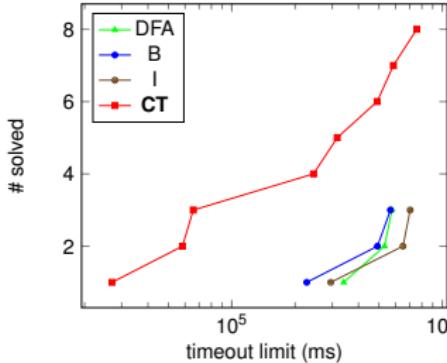
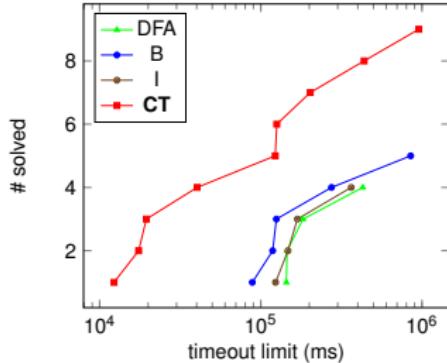
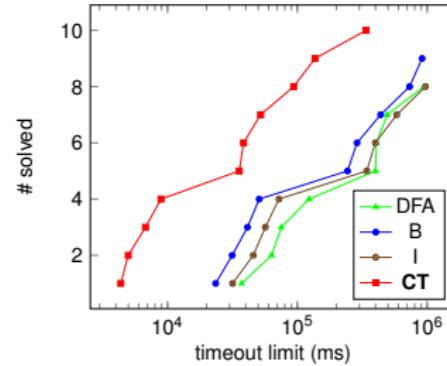
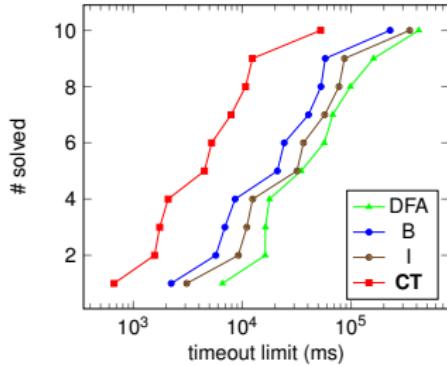
Evaluation

Summary
and
Conclusions

References

Large tables

2500 – 10000 tuples



Background

The
Compact-
Table
Algorithm

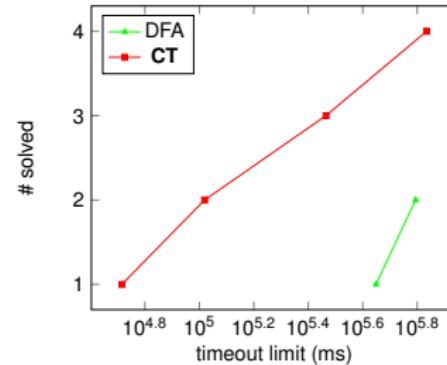
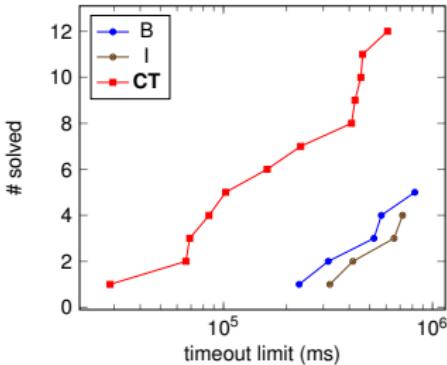
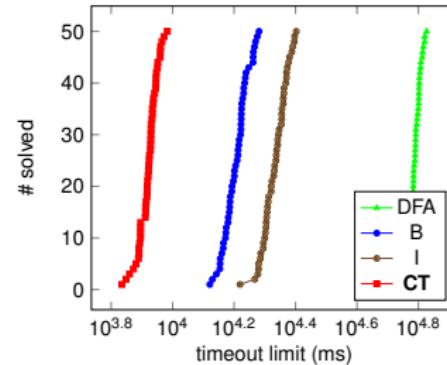
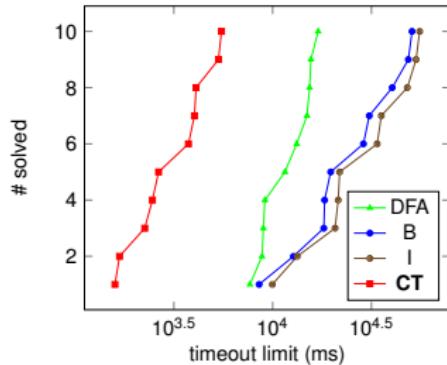
Evaluation

Summary
and
Conclusions

References

Large tables

> 12000 tuples



Background

The
Compact-
Table
Algorithm

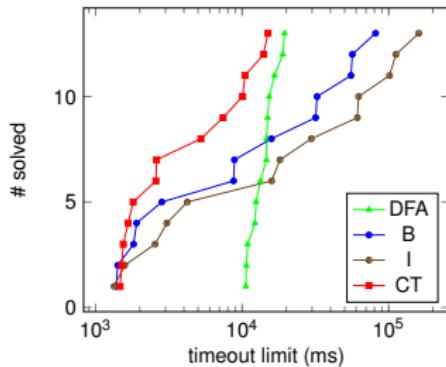
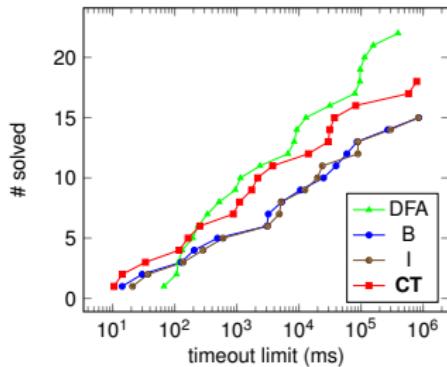
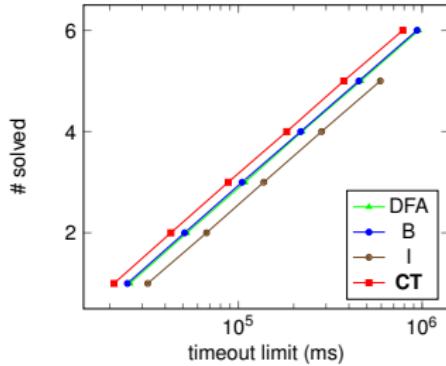
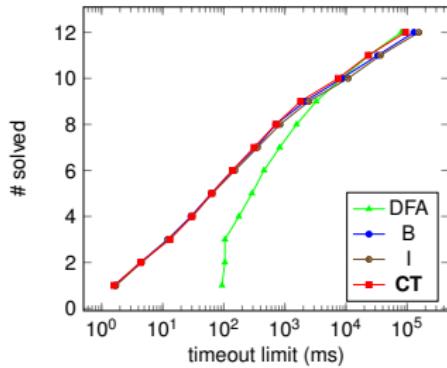
Evaluation

Summary
and
Conclusions

References

Low arities

2 – 3 variables



Background

The
Compact-
Table
Algorithm

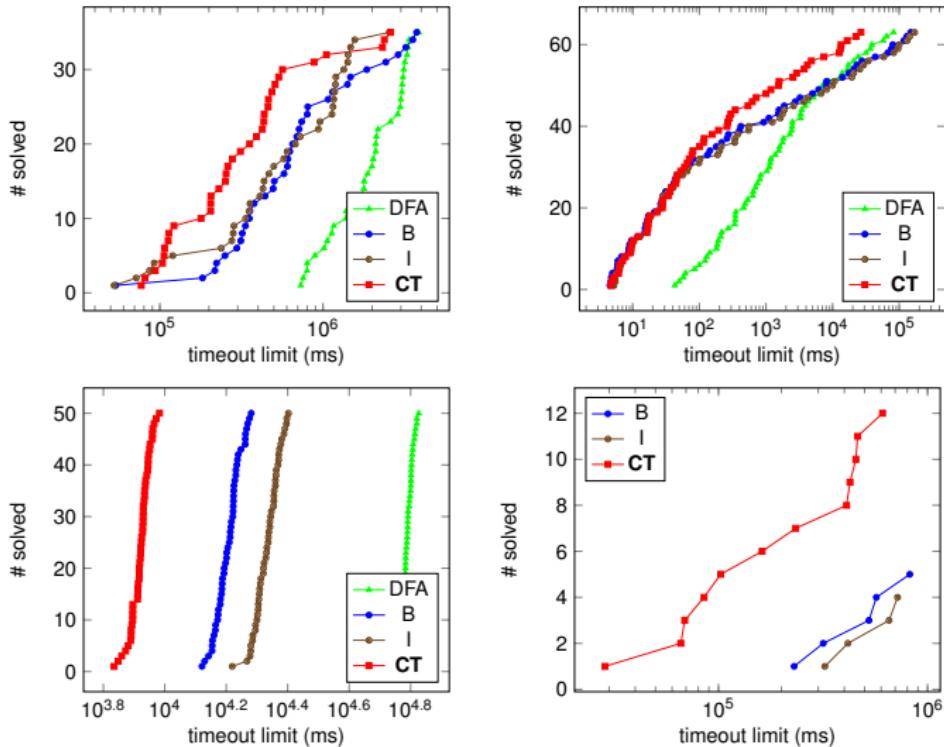
Evaluation

Summary
and
Conclusions

References

High arities

> 5 variables





Small domains

Domain size 2

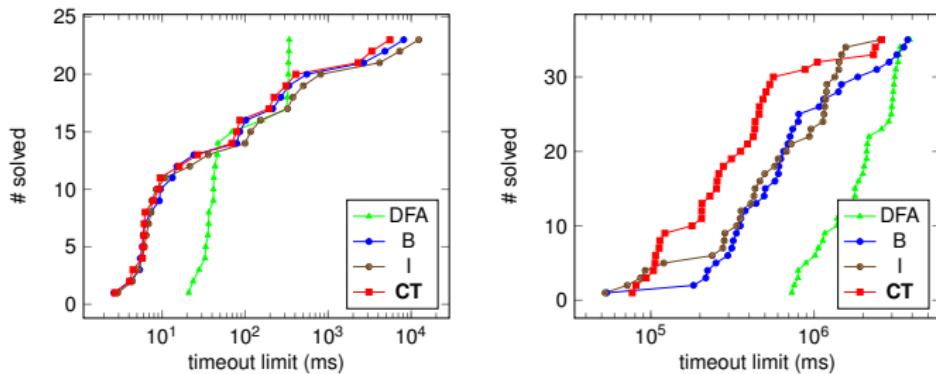
Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References





Large domains

Domain size ≥ 10

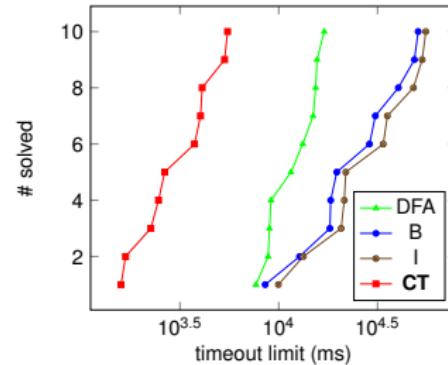
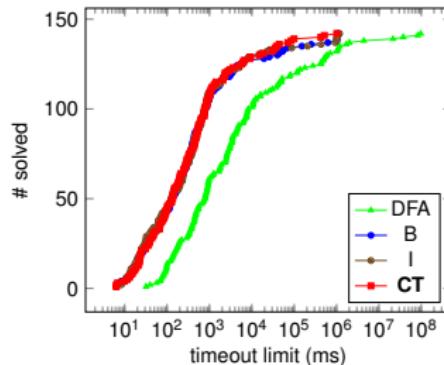
Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References





Results

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

- On all series, CT either outperforms or performs as well as the existing propagators for TABLE
- DFA overall slowest, but fastest on some series
- CT has largest performance gains on series with large tables



Outline

Background

The
Compact-
Table
Algorithm

Evaluation

Summary
and
Conclusions

References

1

Background

2

The Compact-Table Algorithm

3

Evaluation

4

Summary and Conclusions



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:

- This project is to be seen as a proof of concept - not a final version
- Inclusion into Gecode?
- Implement and evaluate generalisations of CT described in [2].



References

-  J. Demeulenaere, R. Hartert, C. Lecoutre, G. Perez, L. Perron, J. Régin and P. Schaus, “Compact-table: Efficiently filtering table constraints with reversible sparse bit-sets”, in *CP 2016*, M. Rueher, Ed., ser. Lecture Notes in Computer Science, vol. 9892, Springer, 2016, pp. 207–223. DOI: [10.1007/978-3-319-44953-1_14](https://doi.org/10.1007/978-3-319-44953-1_14).
-  H. Verhaeghe, C. Lecoutre and P. Schaus, “Extending compact-table to negative and short tables”, in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, S. P. Singh and S. Markovitch, Eds., AAAI Press, 2017, pp. 3951–3957. [Online]. Available: <http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14359>.