KU LEUVEN

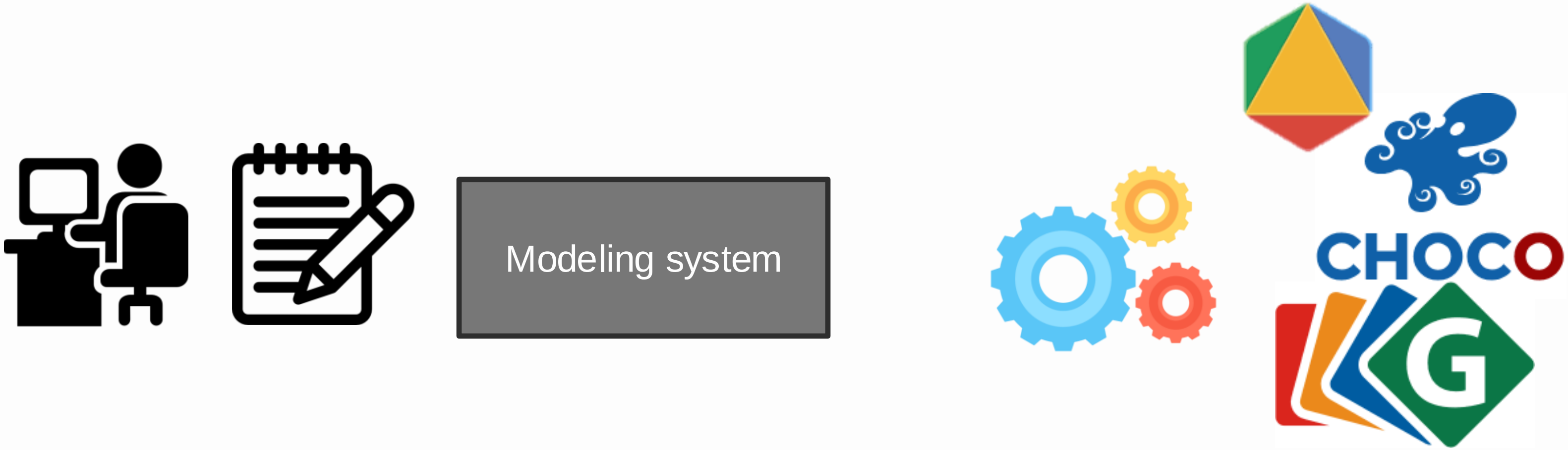
Efficient Modeling of Half-reified global constraints

Ignace Bleukx, Hélène Verhaeghe, Dimos Tsouros, Tias Guns

European Research Council
Established by the European Commission

Tuples
TRUSTWORTHY AI

Modeling systems



Modeling with global constraints

$AllDifferent(x, y, z)$

Modeling system

$AllDifferent(x, y, z)$

+ Easy for modeler

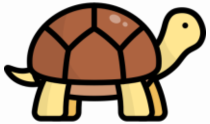
+ Fast propagation!

Decomposing global constraints

$AllDifferent(x, y, z)$

Modeling system

$x \neq y \land x \neq z \land y \neq z$



Efficient Modeling of **Half-reified** global constraints

Reified constraints

"Reification relates the truth value of a constraint to a Boolean variable"

Full-reification: $b \leftrightarrow AllDifferent(x, y, z)$

Reified constraints

"Reification relates the truth value of a constraint to a Boolean variable"

Full-reification: $b \leftrightarrow AllDifferent(x, y, z)$

But... Full reification is hard!
Requires to assert the negation of the constraint if b is False.

E.g., $\neg Circuit(x, y, z)$

Half-reified constraints

- Full reification can be "overkill"
  - $b \Leftrightarrow AllDifferent(x, y, z)$
- Often only need half-reification
  - $b \rightarrow AllDifferent(x, y, z)$
- Easy to derive half-reified propagators!

Half Reification and Flattening

Thibaut Feydy[1], Zoltan Somogyi[1], and Peter J. Stuckey[1]

National ICT Australia and the University of Melbourne, Victoria, Australia
{tfeydy,zs,pjs}@csse.unimelb.edu.au

Half-reified constraints

- Full reification can be "overkill"
  - $b \Leftrightarrow AllDifferent(x, y, z)$
- Often only need half-reification
  - $b \rightarrow AllDifferent(x, y, z)$
- Easy to derive half-reified propagators!

$$\forall v \in \mathcal{V} : f_{b \rightarrow \mathrm{G}(\mathcal{V})}(\mathcal{D})[v] = \begin{cases} f_{\mathrm{G}(\mathcal{V})}(\mathcal{D})[v] & \text{if } \mathcal{D}[b] = \{\mathit{true}\} \\ \mathcal{D}[v] & \text{otherwise} \end{cases}$$

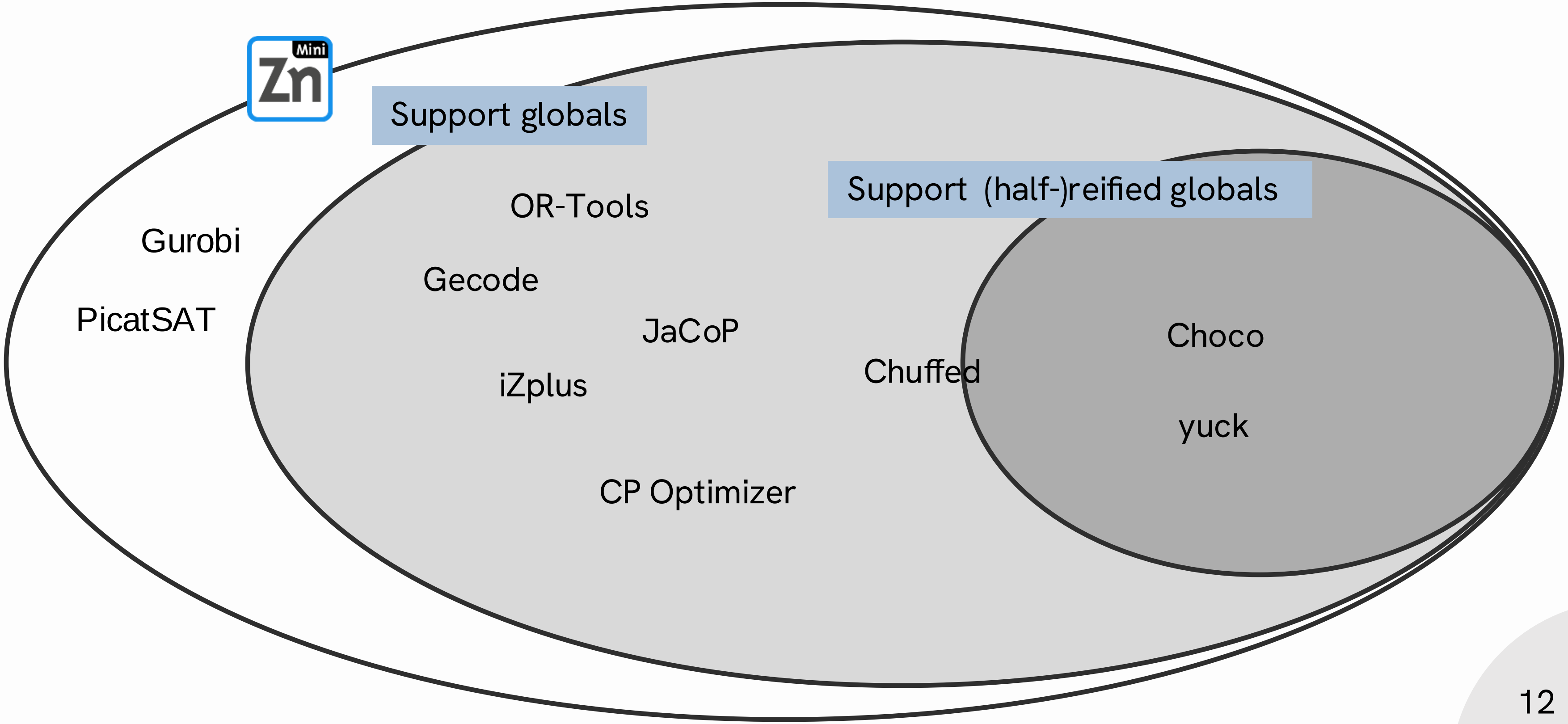
$$f_{b \rightarrow \mathrm{G}(\mathcal{V})}(\mathcal{D})[b] = \begin{cases} \mathcal{D}[b] \setminus \{\mathit{true}\} & \text{if } f_{\mathrm{G}(\mathcal{V})}(\mathcal{D}) \text{ is a false domain} \\ \mathcal{D}[b] & \text{otherwise} \end{cases}$$

Half Reification and Flattening

Thibaut Feydy[1], Zoltan Somogyi[1], and Peter J. Stuckey[1]

National ICT Australia and the University of Melbourne, Victoria, Australia
{tfeydy,zs,pjs}@csse.unimelb.edu.au

Solver support for (half-)reification



Decomposing reified gobal constraints

$b \rightarrow AllDifferent(x, y, z)$

Modeling system

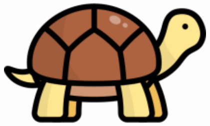
OR-Tools

Gecode JaCoP

iZplus

CP Optimizer

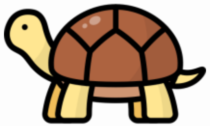
$b \rightarrow (x \neq y \wedge x \neq y \wedge y \neq z)$



Why should you care about half-reified global constraints?

Many applications

- Max-CSP solving
- Incremental solving
- Assumption-based solving
- MUS-computation
- Flattening [by modeling languages]
- ...



$$Maximize \sum w_c \cdot b_c$$
$$st. \quad b_c \rightarrow c \quad \forall c \in C$$

Instead, rewrite with aux vars!

$$b \rightarrow AllDifferent(x, y, z)$$

Instead, rewrite with aux vars!

$$b \rightarrow AllDifferent(x, y, z)$$

$$b \rightarrow (AllDifferent(x', y', z') \land x = x' \land y = y' \land z = z')$$

Instead, rewrite with aux vars!

$$b \rightarrow AllDifferent(x, y, z)$$

$$b \rightarrow (AllDifferent(x', y', z') \land x = x' \land y = y' \land z = z')$$

$$AllDifferent(x', y', z') \land (b \rightarrow (x = x' \land y = y' \land z = z'))$$

Instead, rewrite with aux vars!

$$AllDifferent(x', y', z') \land (b \rightarrow (x \; = x' \land y = y' \land z = z'))$$

When b is True:

$$AllDifferent(x', y', z') \land x = x' \land y = y' \land z = z'$$
$$AllDifferent(x, y, z\ )$$

When b is False:

$$AllDifferent(x', y', z') \land True$$

Instead, rewrite with aux vars!

$$AllDifferent(x', y', z') \wedge (b \rightarrow (x = x' \wedge y = y' \wedge z = z'))$$

When b is True:

$$AllDifferent(x', y', z') \wedge x = x' \wedge y = y' \wedge z = z'$$
$$AllDifferent(x, y, z)$$

When b is False:

$$AllDifferent(x', y', z') \wedge True$$

+ No decomposition
- Extra variables in model
- Worse propagation compared to native solver support

Special case: functional constraints

$b \rightarrow Min(v, X)$

$Min(v', X) \wedge (b \rightarrow v = v')$

When b is True:

$Min(v', X) \wedge v = v'$
$Min(v, X)$

When b is False:

$Min(v', X)$

Same propagation strength as native support!

Special case: functional constraints

$b \rightarrow Min(v, X)$

$Min(v', X) \wedge (b \rightarrow v = v')$

When b is True:

$Min(v', X) \wedge v = v'$
$Min(v, X)$

When b is False:

$Min(v', X)$

Same propagation strength as native support!

Also works for full-reification!

Implemented in most modeling systems!

On the reification of global constraints

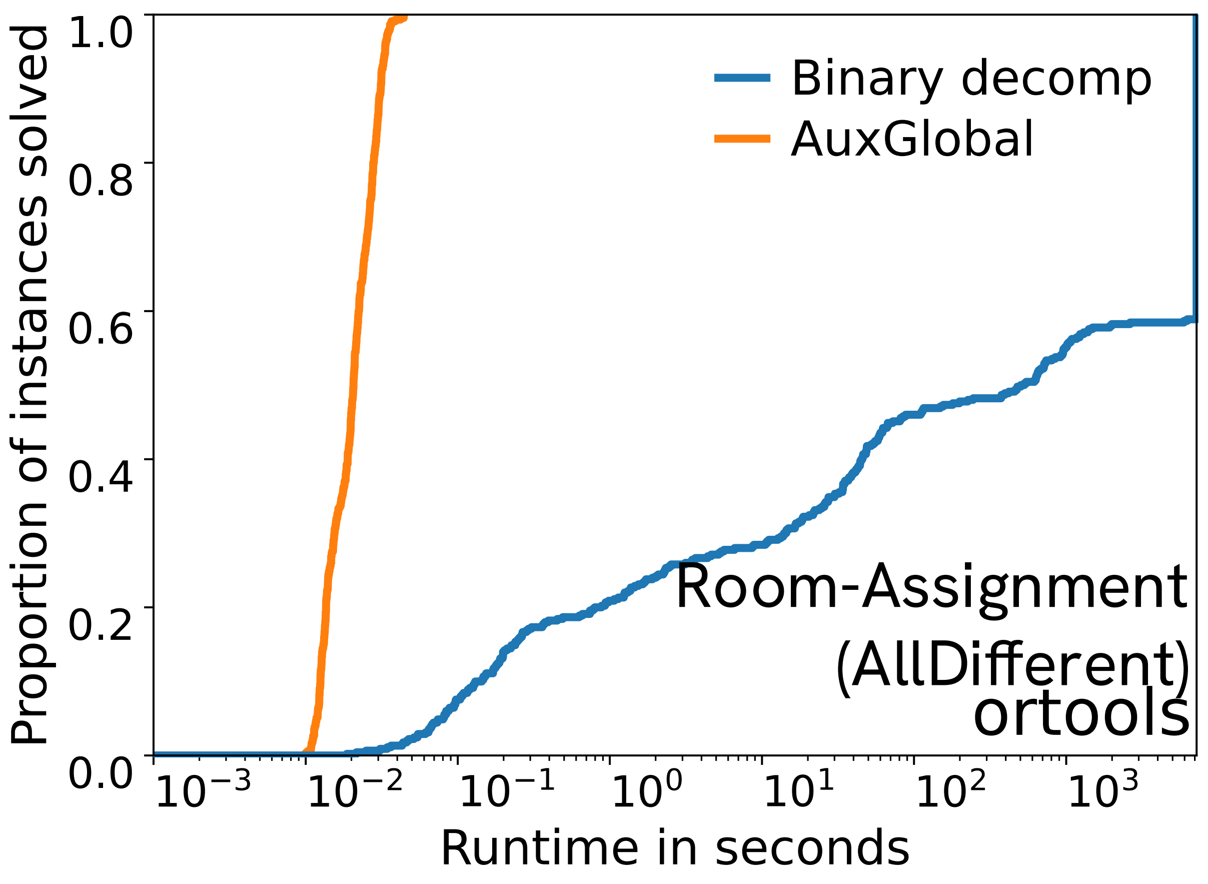
Nicolas Beldiceanu · Mats Carlsson · Pierre Flener · Justin Pearson

So.... Does it work?

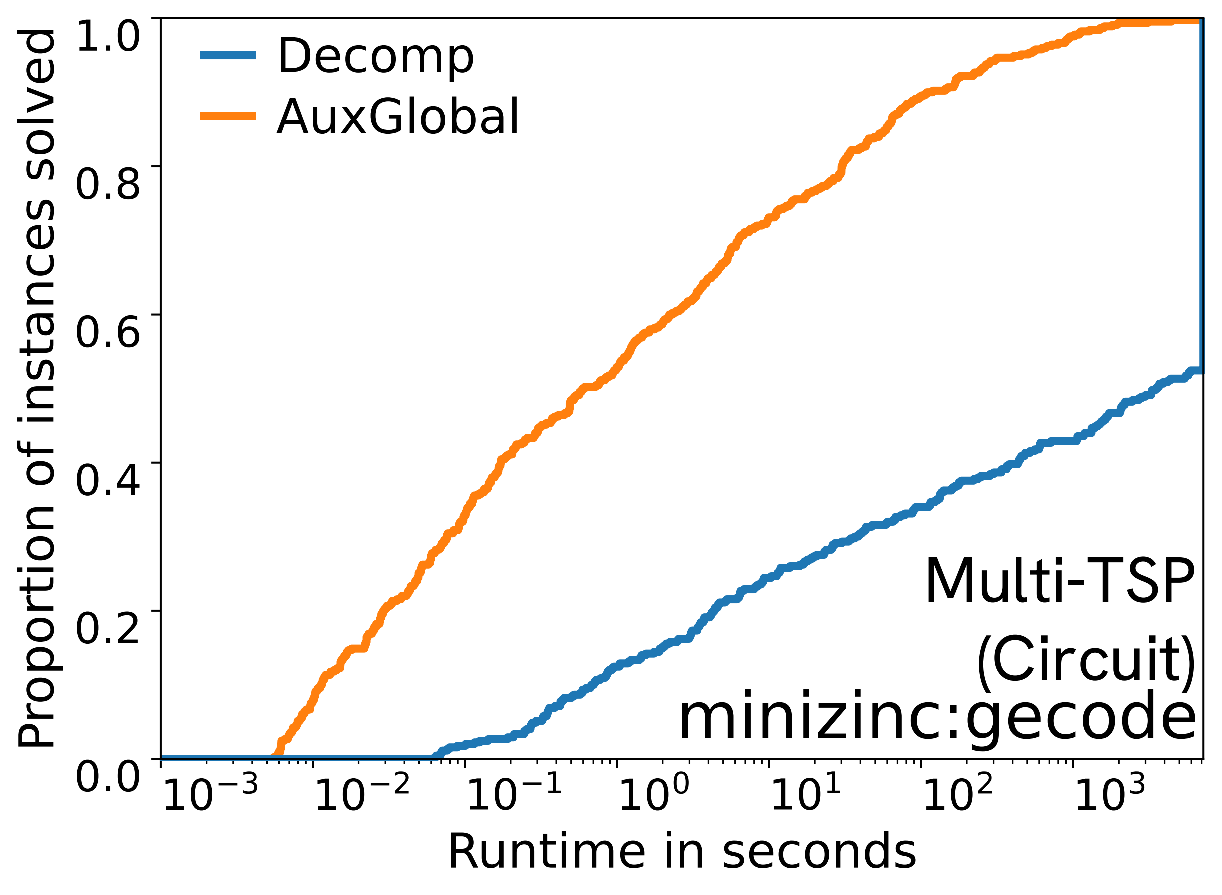
- Max-CSP benchmarks with different global constraints:
  - Room-assignment: AllDifferent
  - Multi-TSP: Circuit
  - RCPSP: Cumulative

- Tested on:
  - OR-Tools
  - Gecode
  - Choco

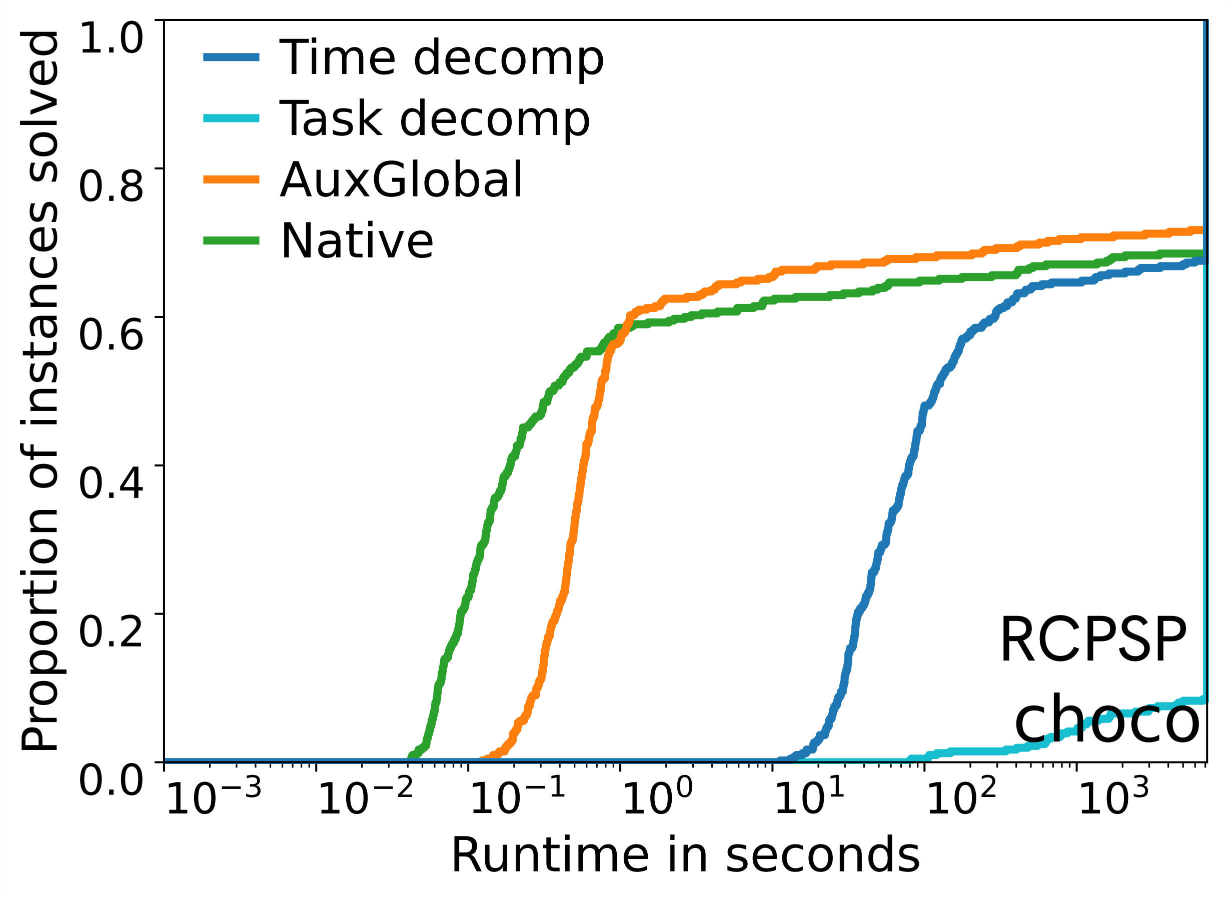
- Through the CPMpy modeling system



Binary decomp
AuxGlobal
Room-Assignment
(AllDifferent)
ortools
Proportion of instances solved
1.0
0.8
0.6
0.4
0.2
0.0
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
Runtime in seconds

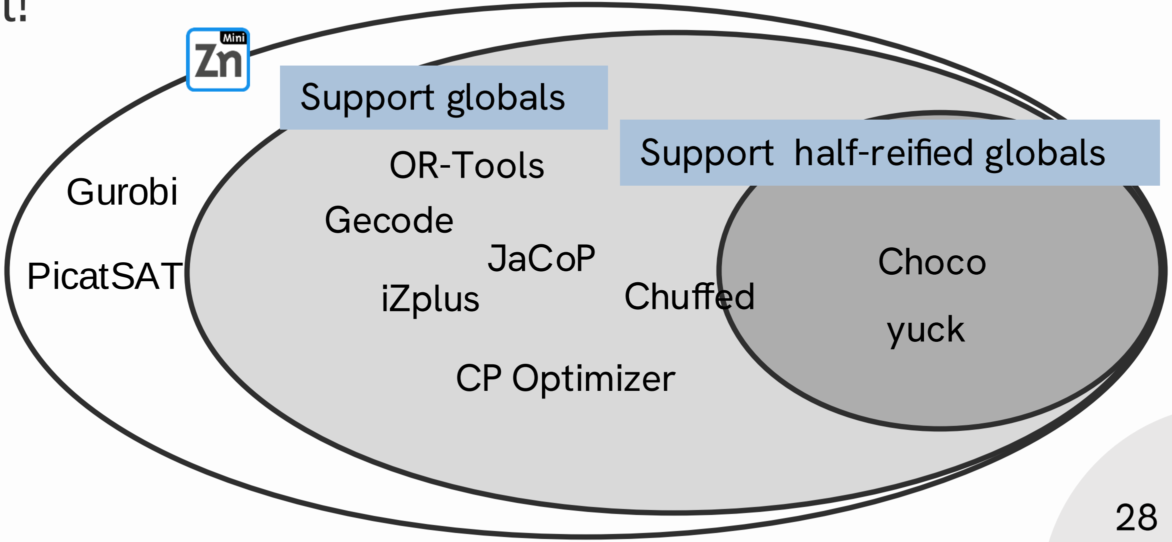


Decomp
AuxGlobal
Proportion of instances solved
1.0
0.8
0.6
0.4
0.2
0.0
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
Runtime in seconds
Multi-TSP
(Circuit)
minizinc:gecode


Proportion of instances solved
1.0
0.8
0.6
0.4
0.2
0.0
Time decomp
Task decomp
AuxGlobal
Native
RCPSP
choco
10^{-3}
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
Runtime in seconds

Conclusions

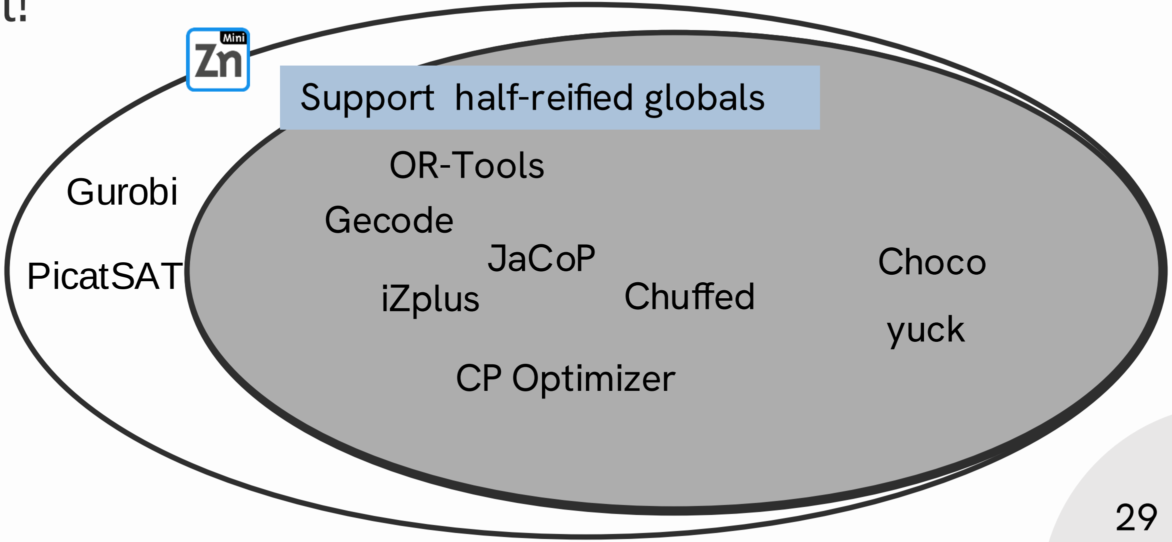
- Reformulation with auxiliary variables is much faster compared to decomposing
- Competitive with solver-native approaches
- Easy to implement!



Conclusions

- Reformulation with auxiliary variables is much faster compared to decomposing
- Competitive with solver-native approaches
- Easy to implement!

Enable support for half-reified global constraints for ANY CP-solver



Next steps

- Compare with state-of-the-art Max-CSP solvers
- Evaluate flattening use-cases?
- Minimize auxiliary variables for non-functional constraints

Questions?