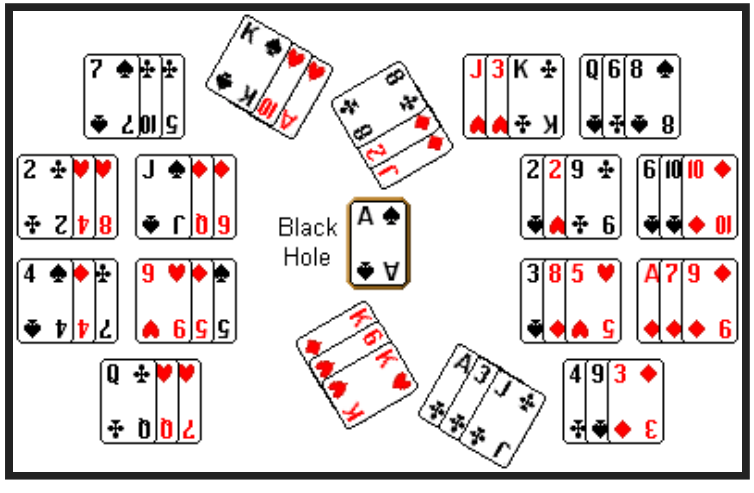
AUTO-TABLING FOR SUBPROBLEM PRESOLVING IN MINIZINC

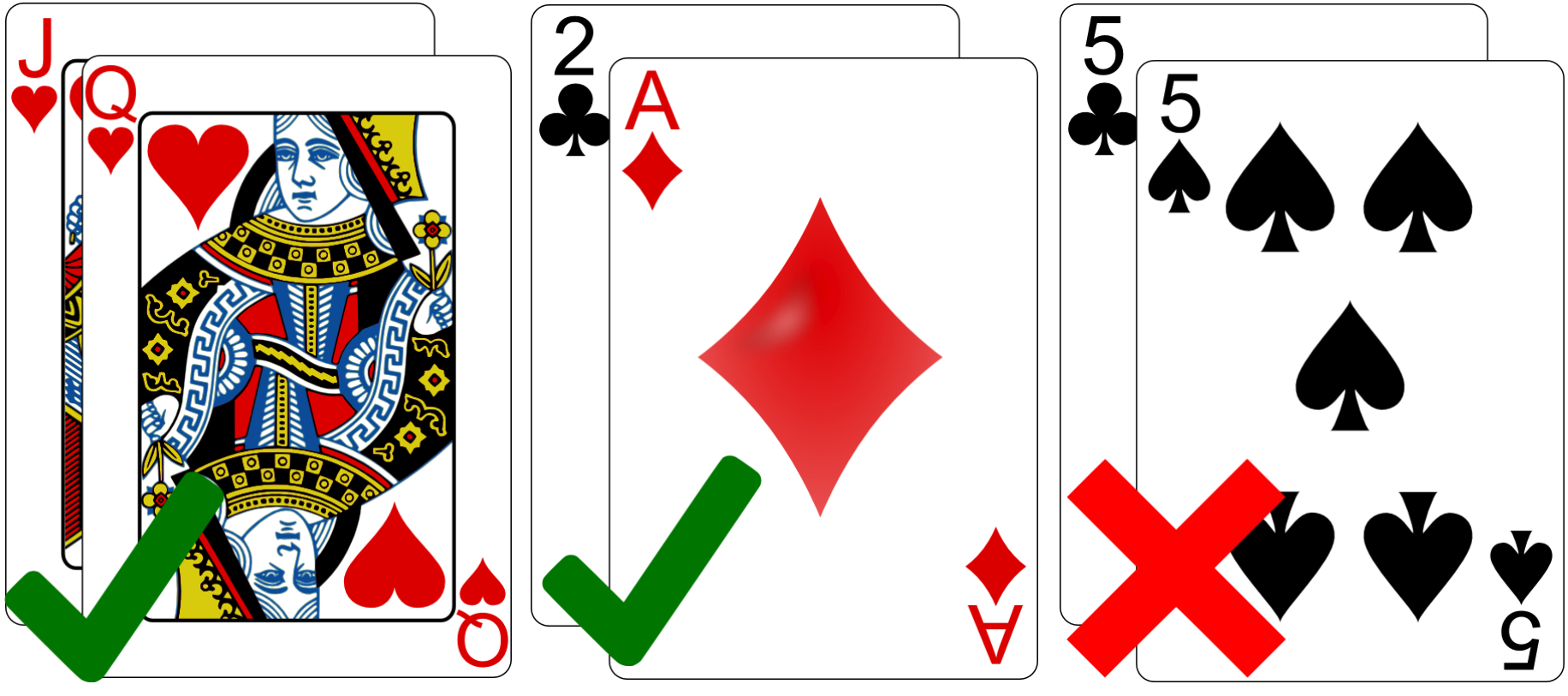
JIP J. DEKKER - UPPSALA UNIVERSITY
GUSTAV BJÖRDAL - UPPSALA UNIVERSITY
MATS CARLSSON - SICS
PIERRE FLENER - UPPSALA UNIVERSITY
JEAN-NOËL MONETTE - TACTON

BLACK HOLE PATIENCE



- **Goal**: Find an order in which all cards can be moved into the middle (the black hole).
- **Constraint**: The solution order must match the order of the heaps.
- **Constraint: Each card in the solution order must be one rank apart from the next one.**

ONE RANK APART



WHAT WE WOULD LIKE TO WRITE:

```
predicate rank_apart(var 1..52: a, var 1..52: b)
    = abs( (a - b) mod 13 ) in {1,12};
```

WHAT YOU FIND IN THE MINIZINC CHALLENGE:

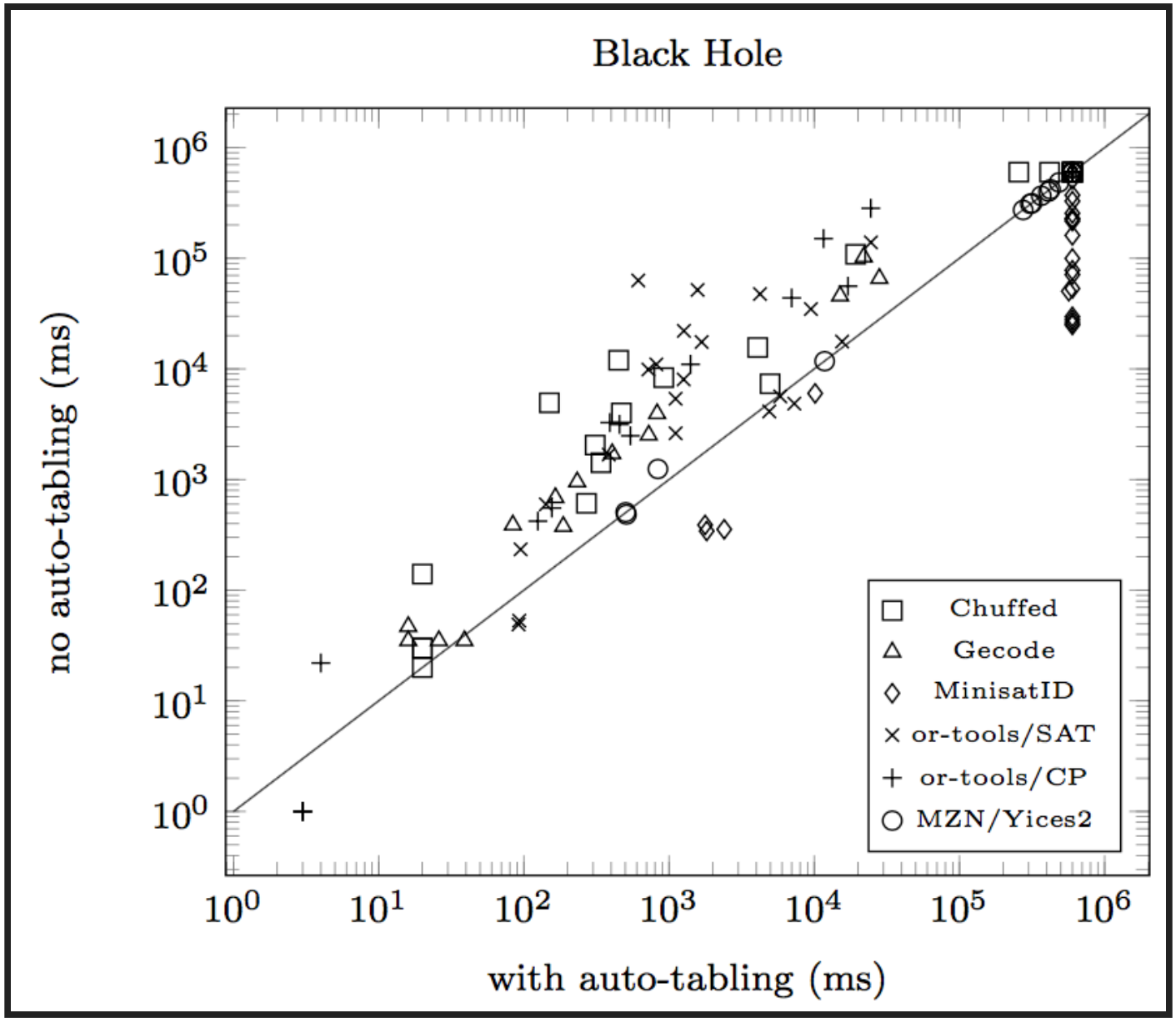
```
neighbours = array2d(1..(52*2*4), 1..2, [
    1, 2,
    1, 13,
    1, 15,
    1, 26,
    1, 28,
    1, 39,
    1, 41,
    1, 52,
    2, 1,
    2, 3,
    2, 14,
    ...
```

TABLING

- Replacing part of the model by a precomputed table.
- This can work because:
  - Table constraint can provide domain-consistency.
  - Propagation is faster for `table` constraints.
- It works similarly with other extensional constraints like `regular` and `MDD`.

USING OUR TOOL

```
predicate rank_apart(var 1..52: a, var 1..52: b)
::presolve(autotable)
    = abs( (a - b) mod 13 ) in {1,12};
```


Black Hole
no auto-tabling (ms)
with auto-tabling (ms)
10^0
10^1
10^2
10^3
10^4
10^5
10^6
Chuffed
Gecode
MinisatID
or-tools/SAT
or-tools/CP
MZN/Yices2

- MiniZinc is a high-level constraint modelling language
- Focus on modelling, not programming!
- Model once, run everywhere! Not just on CP backends.

PREDICATES AND ANNOTATIONS

- A way to form “sub-models” within MiniZinc

```
predicate rank_apart(var 1..52: a, var 1..52: b)
::presolve(autotable)
    = abs( (a - b) mod 13 ) in {1,12};
```

OUR GOAL

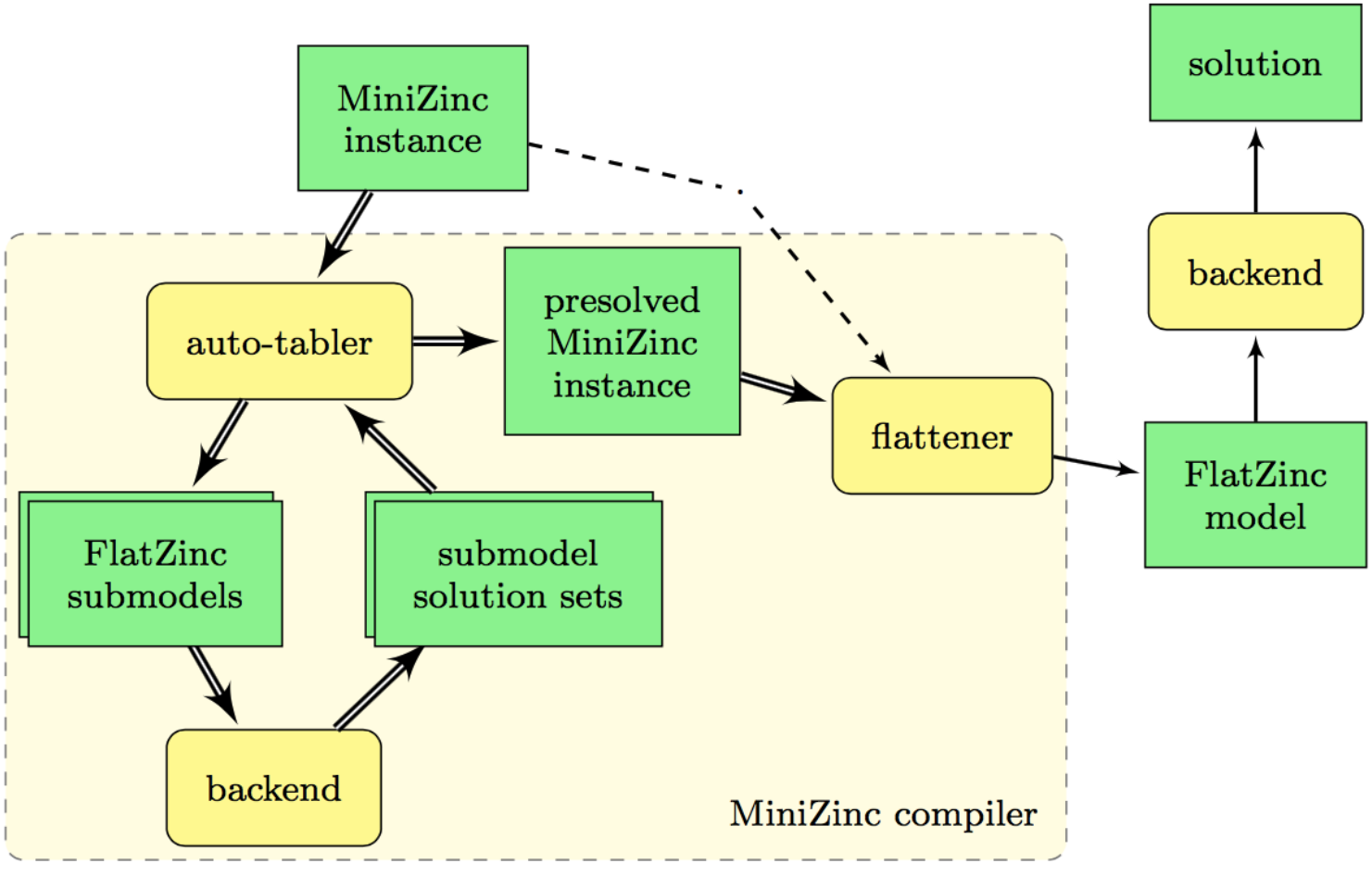
Automate the tabulation of MiniZinc predicates.

ADDITIONAL GOALS

- Integrate this automation within the MiniZinc compiler.
- The automated tabling should not require change to the solver backends.

Ease of use is key!

THE COMPILATION PROCESS



BLACK HOLE MODEL

```
...
% Card at position
array[1..52] of var 1..52: card;
...
predicate rank_apart(var 1..52: a, var 1..52: b)
::presolve(autotable)
    = abs( (a - b) mod 13 ) in {1,12};
...
constraint forall(i in 1..51)(
    rank_apart(card[i], card[i+1])
);
...
solve satisfy;
```

THE AUTOTABLE SUBMODEL

```
var 1..52: a;
var 1..52: b;

predicate rank_apart(var 1..52: a, var 1..52: b)
    = abs( (a - b) mod 13 ) in {1,12};

constraint rank_apart(a, b);

solve satisfy;
```

REPLACING THE PREDICATE

```
% Card at position
array[1..52] of var 1..52: card;

predicate rank_apart(var 1..52: a, var 1..52: b)
    = table_int(
        [a, b],
        array2d(1..416, index_set([a, b]),
            [2, 1, 13, 1, 15, 1, 26, 1, 28, ...])
    );

constraint forall(i in 1..51)(
    rank_apart(card[i], card[i+1])
);

solve satisfy;
```

TEST CASES

All our test cases are available on GitHub

MiniZinc Challenges

- Black Hole Patience
- JP Encoding Problem
- Elitserien Handball

Master's Thesis

- Block Party Meta-cube

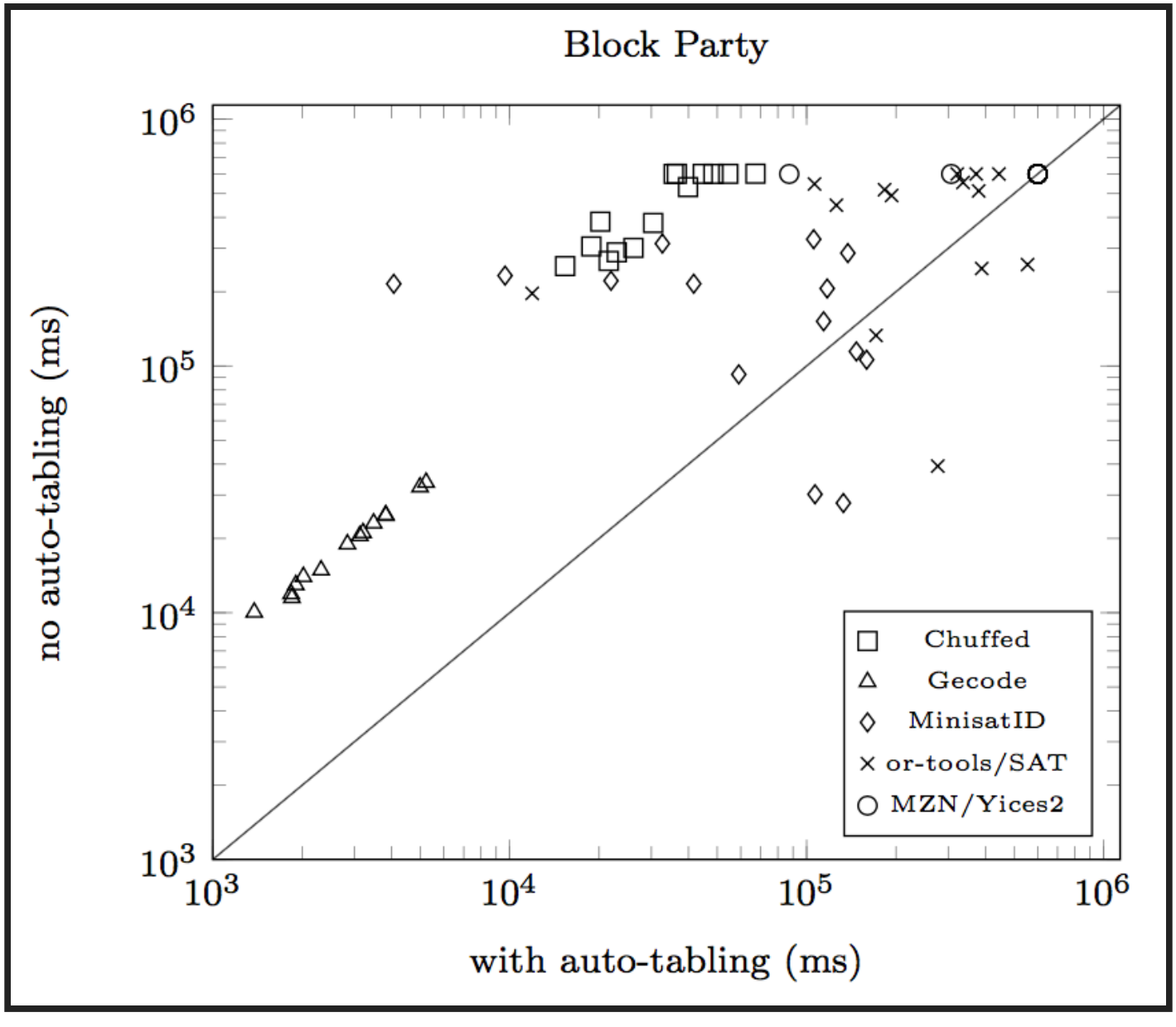
TESTED BACKENDS

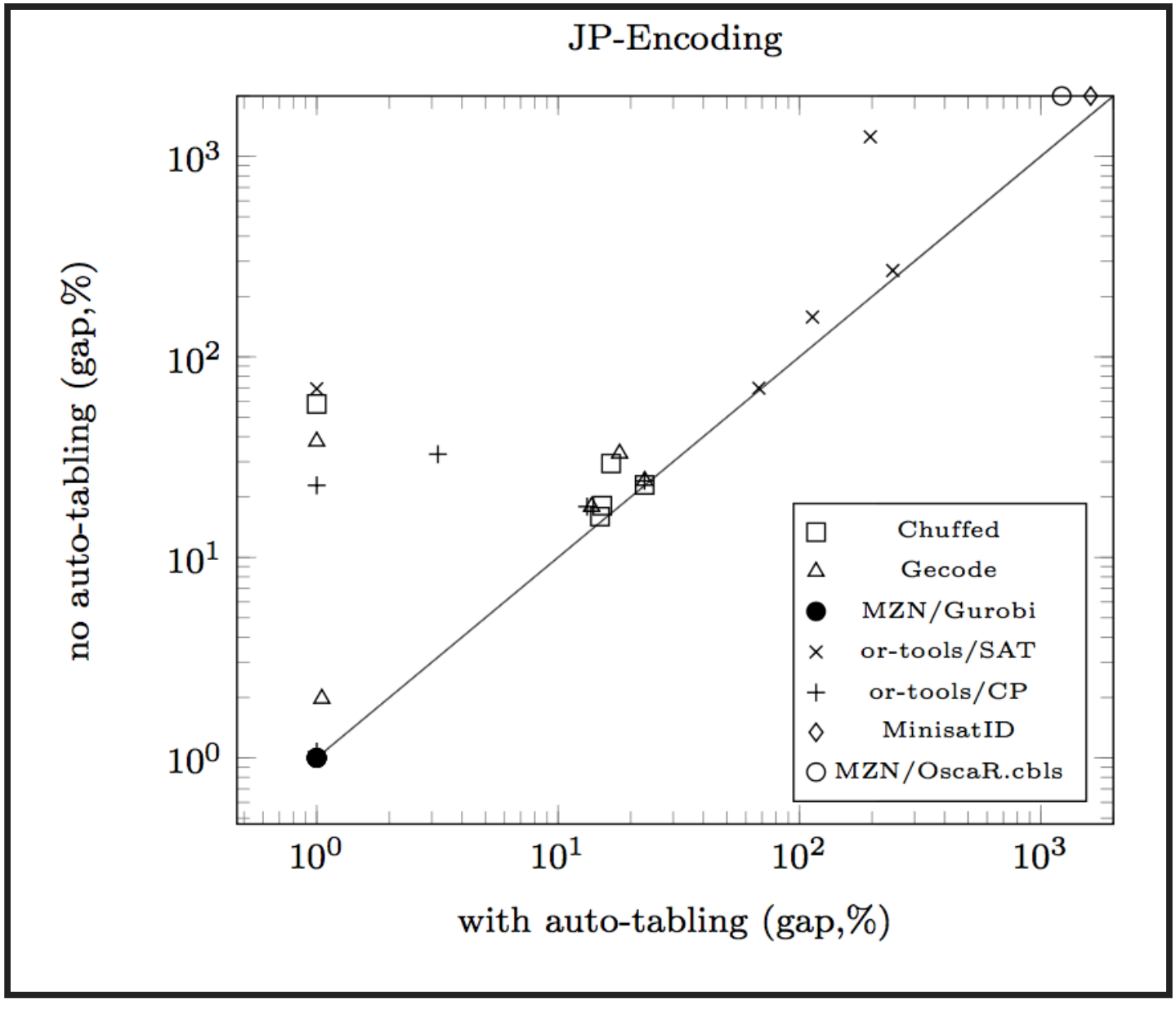
Constraint Programming

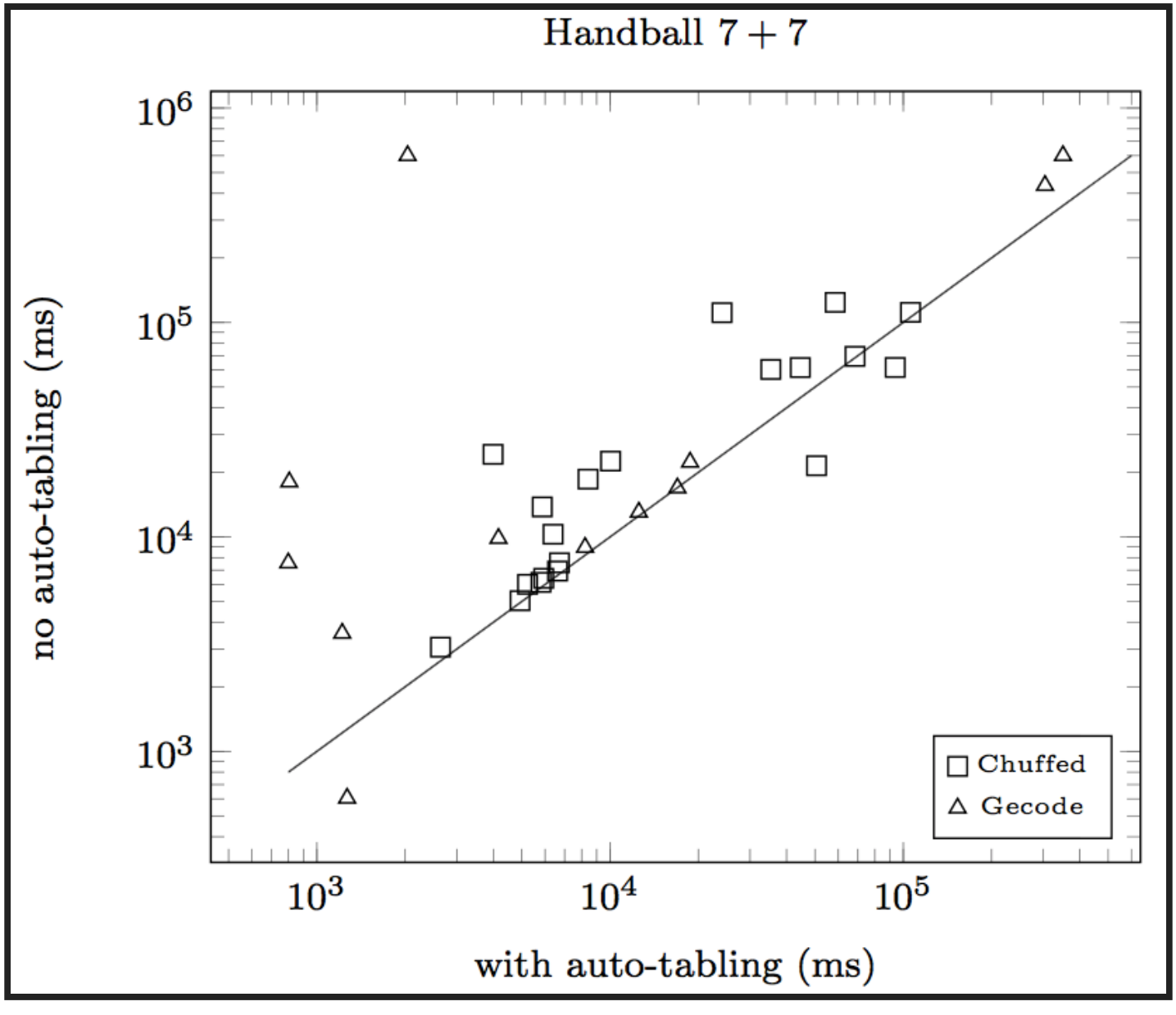
- Gecode
- Chuffed
- or-tools

Other Backends

- or-tools/SAT — *SAT solver*
- MinisatID — *hybrid solver*
- MZN/Yices2 — *SMT solver*
- MZN/OscaR.cbls — *CBLS solver*


Block Party
no auto-tabling (ms)
with auto-tabling (ms)
10^3
10^4
10^5
10^6
Chuffed
Gecode
MinisatID
or-tools/SAT
MZN/Yices2


JP-Encoding
no auto-tabling (gap,%)
with auto-tabling (gap,%)
10^0
10^1
10^2
10^3
Chuffed
Gecode
MZN/Gurobi
or-tools/SAT
or-tools/CP
MinisatID
MZN/OscaR.cbls


Handball 7 + 7
10^6
10^5
10^4
10^3
no auto-tabling (ms)
10^3
10^4
10^5
with auto-tabling (ms)
Chuffed
Gecode

CONCLUSIONS

- Tabling is made easy to use and nonintrusive.
- Auto-tabling may make a big difference in model performance.
- Try it! It's open source!

FUTURE WORK

- The caching of presolving results.
- Support for `float` and `set` variables
- **Done:** Presolve after flattening.
- Study whether our presolving observations generalise.
- Presolve into an MDD instead of a table

THANKS TO

- Justin Pearson
- Peter Stuckey
- Guido Tack
- Diego de Uña Gomez
- The anonymous referees for their helpful comments.

Pierre Flener and Gustav Björdal are supported by the Swedish Research Council (VR) under grant 2015-4910.

THANK YOU FOR LISTENING!

JIP J. DEKKER

GitHub: @Dekker1
Twitter: @DekkerOne
E-mail: jip@dekker.one

EXTRA SLIDES

Press the down-button

DIFFERENT STRATEGIES

- Solve in different scopes
- Default: the *instance*-strategy

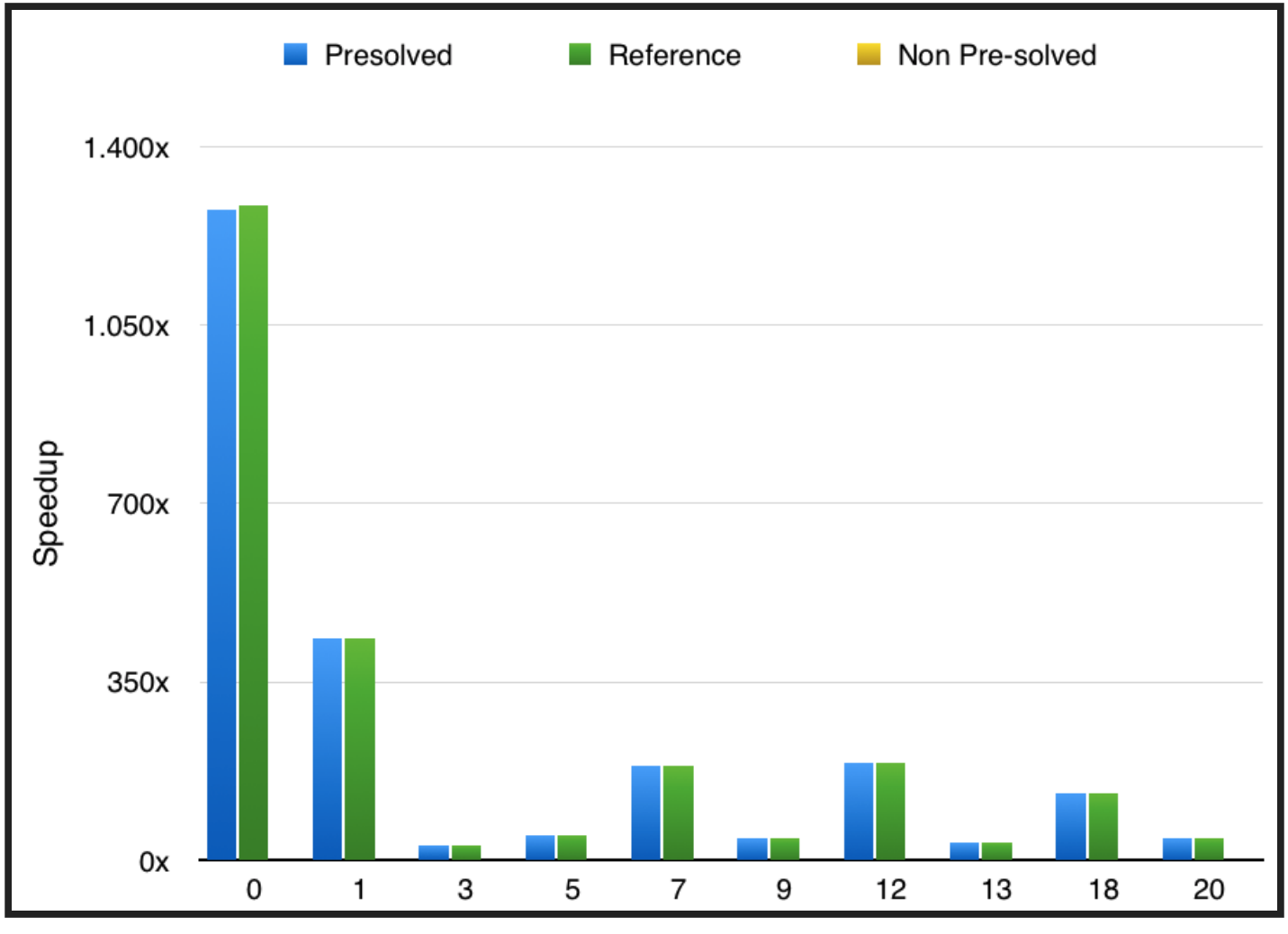
MODEL STRATEGY

- Solve according to the predicate definition.
- Advantages
  - Allows you to save the result and use for more instances
- Disadvantages
  - Can't use variable array sizes
  - Can't use external data
  - Big resulting tables

CALLS STRATEGY

- Solve for every FlatZinc call separately
- Advantages
  - Small resulting tables
  - Can use different amounts of variables per call
- Disadvantages
  - High presolving times

BLACK HOLE REFERENCING



BLOCK PARTY

BLOCK PARTY

```
predicate link_cube_and_symbols(
    array[1..4] of var int: cs
) :: presolve(autotable)
= let{
   var 1..24: pos;
   var int: cube = cs[1];
} in forall(i in 1..3)(
    data[cube,pp[pos,i]]=cs[i+1]
);
```