IMPLEMENTATION AND EVALUATION OF A SWEEP-BASED PROPAGATOR FOR DIFFN IN GECODE

Mikael Östlund
Supervised by: Mats Carlsson (RISE SICS AB)
Reviewed by: Pierre Flener (ASTRA)

1 June, 2017

Master Thesis (course 1DT540)
Uppsala University
Sweden

OUTLINE

OUTLINE

- Background
- Sweep Algorithm
- Evaluation Setup
- Results
- Discussion
- Future Work

BACKGROUND

CONSTRAINT PROGRAMMING

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

CONSTRAINT PROGRAMMING

- Sudoku
  - All rows must have distinct values
  - All columns must have distinct values
  - Each disjoint 3×3 region must have distinct values

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

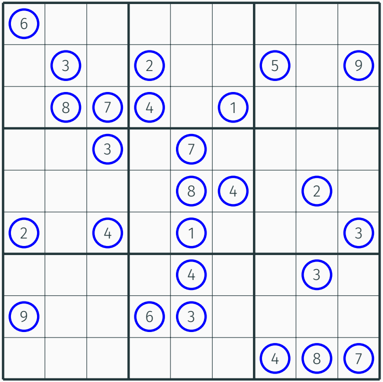
CONSTRAINT PROGRAMMING

- Variables

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

CONSTRAINT PROGRAMMING

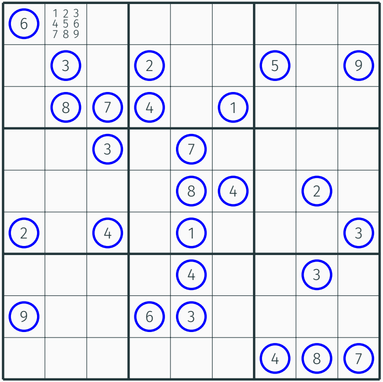
- Variables
- Values



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

CONSTRAINT PROGRAMMING

- Variables
- Values



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

CONSTRAINT PROGRAMMING

- Variables
- Values
- Constraints

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

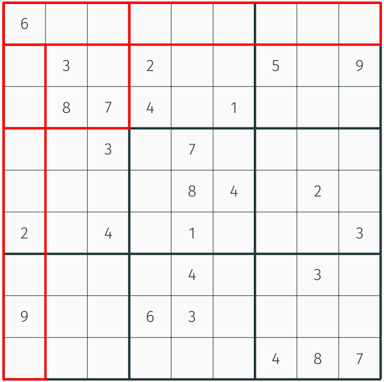
CONSTRAINT PROGRAMMING

- Variables
- Values
- Constraints

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

CONSTRAINT PROGRAMMING

- Variables
- Values
- Constraints



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

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

- Constraint programming solver
    - Model → Solution

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

- Constraint programming solver
    - Model → Solution
- Search

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

- Constraint programming solver
    - Model → Solution
- Search
- Inference (through propagation) ☜

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

- Constraint programming solver
    - Model → Solution
- Search
- Inference (through propagation) ☜
    - Propagator

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

- Constraint programming solver
    - Model → Solution
- Search
- Inference (through propagation) ☜
    - Propagator
        - Implements a constraint

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

- Constraint programming solver
    - Model → Solution
- Search
- Inference (through propagation) ☜
    - Propagator
        - Implements a constraint
        - *Infers* what values violate constraints and *prunes* them

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

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

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

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

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

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

SOLVING A CONSTRAINT PROGRAMMING PROBLEM

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

GECODE

- Constraint programming solver

GECODE

- Constraint programming solver
- Customisable
    - Custom propagators ☜
    - Search heuristics

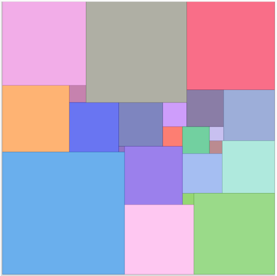
GECODE

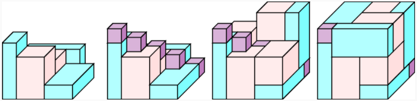
- Constraint programming solver
- Customisable
    - Custom propagators ☜
    - Search heuristics
- Award winning

GECODE

- Constraint programming solver
- Customisable
    - Custom propagators ☜
    - Search heuristics
- Award winning
- Copying solver

DIFFN

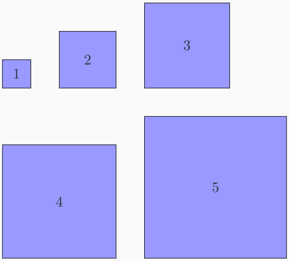




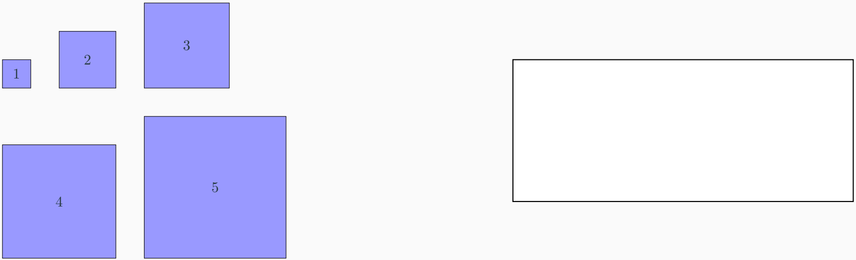
Source: `http://sofdem.github.io/gccat/gccat/KConway_packing_problem.html`

- Hyperrectangles must not overlap

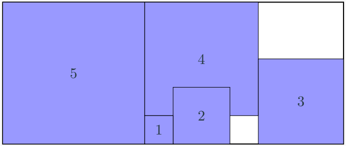
DIFFN



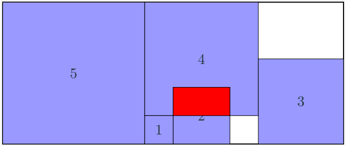
DIFFN



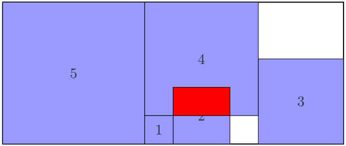
DIFFN

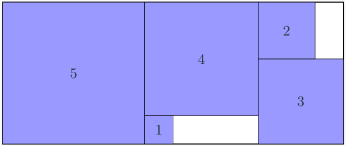


DIFFN



DIFFN

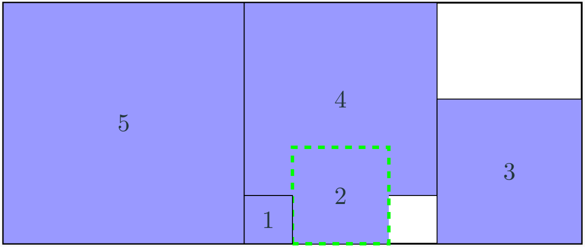




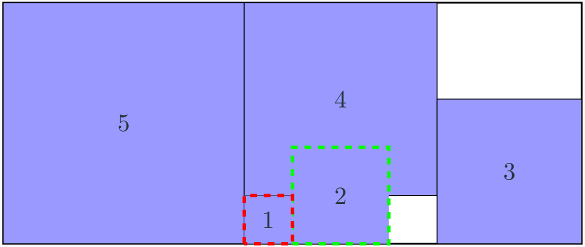
DIFFN

- Common approach to DIFFN: pairwise reasoning

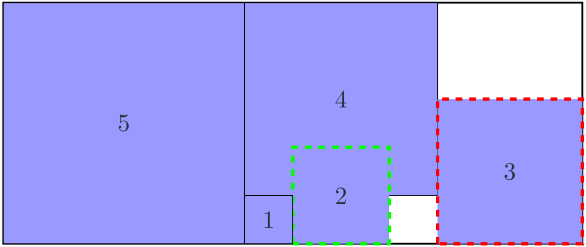
DIFFN - PAIRWISE



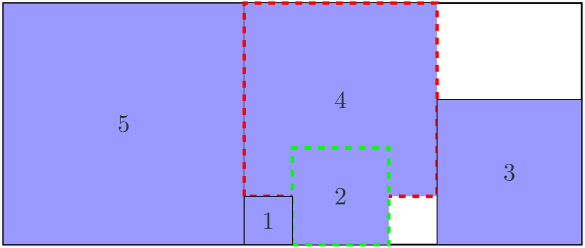
DIFFN - PAIRWISE



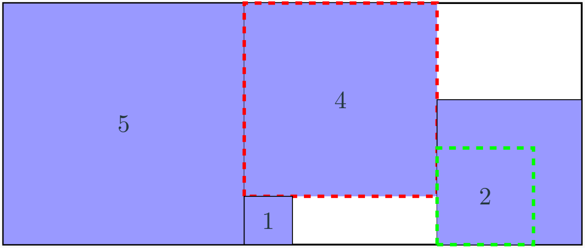
DIFFN - PAIRWISE



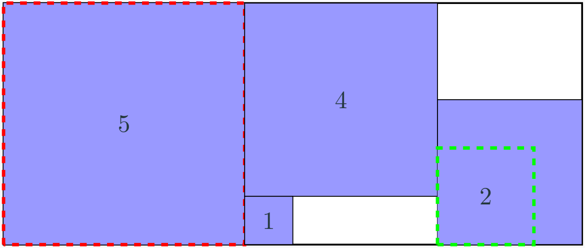
DIFFN - PAIRWISE



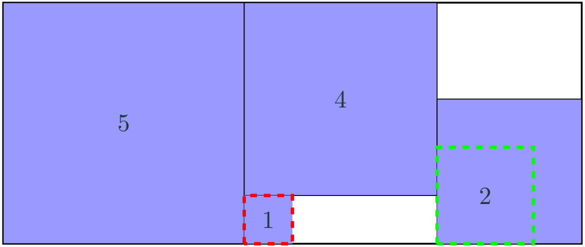
DIFFN - PAIRWISE



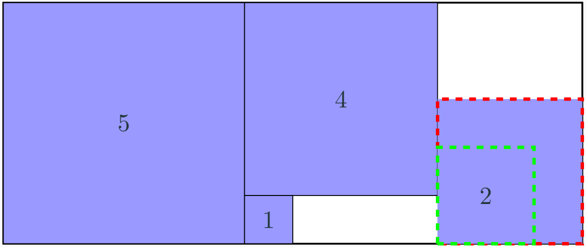
DIFFN - PAIRWISE



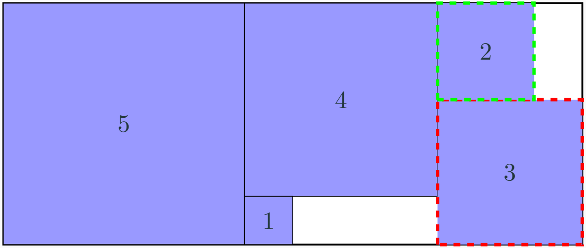
DIFFN - PAIRWISE



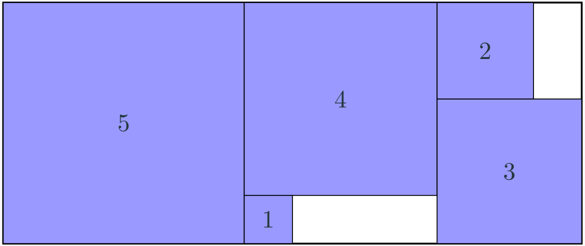
DIFFN - PAIRWISE



DIFFN - PAIRWISE



DIFFN - PAIRWISE



SWEEP ALGORITHM

SWEEP ALGORITHM

- Proposed by Nicolas Beldiceanu and Mats Carlsson (2001)

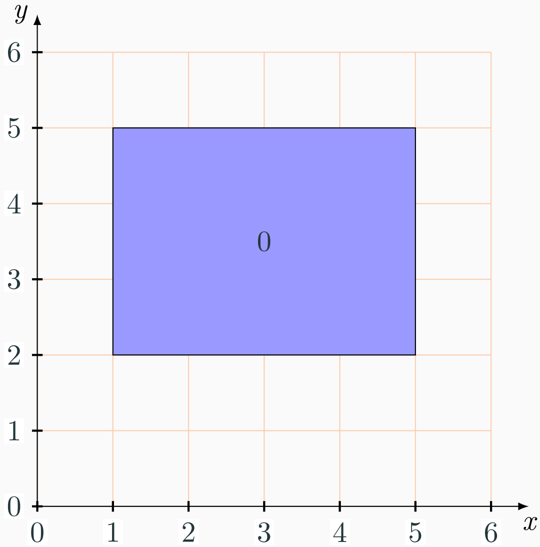
SWEEP ALGORITHM

- Proposed by Nicolas Beldiceanu and Mats Carlsson (2001)
- Potential for stronger inference ($\Rightarrow$ less search)

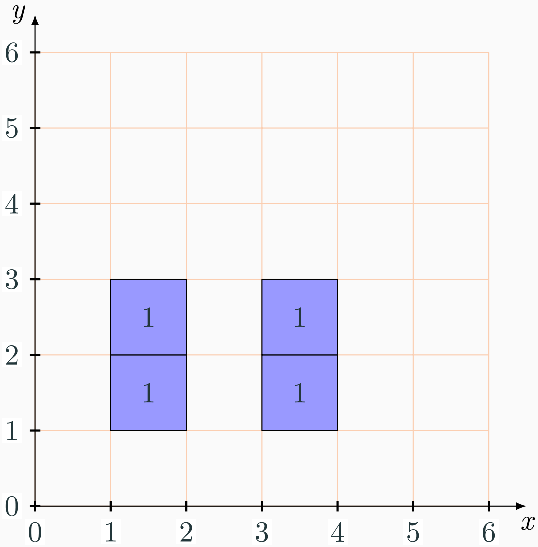
SWEEP ALGORITHM

- Proposed by Nicolas Beldiceanu and Mats Carlsson (2001)
- Potential for stronger inference ($\Rightarrow$ less search)
- Aggregated reasoning

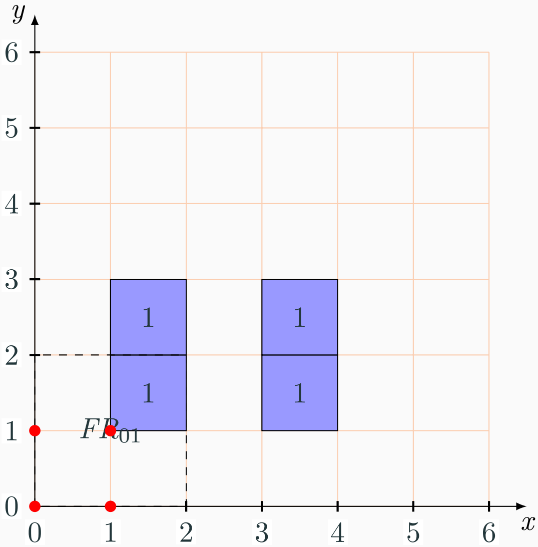
FORBIDDEN REGIONS



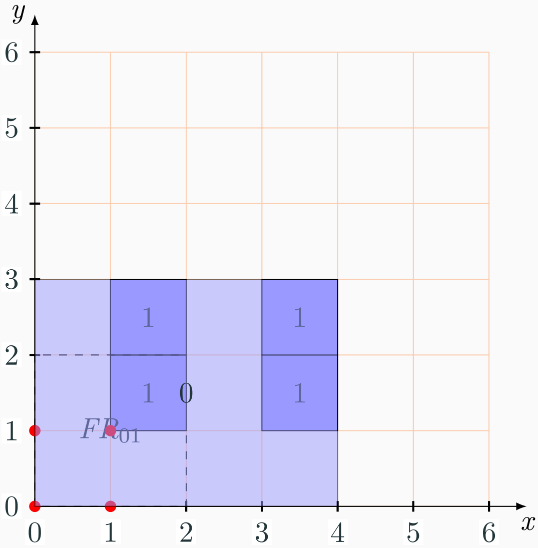
FORBIDDEN REGIONS



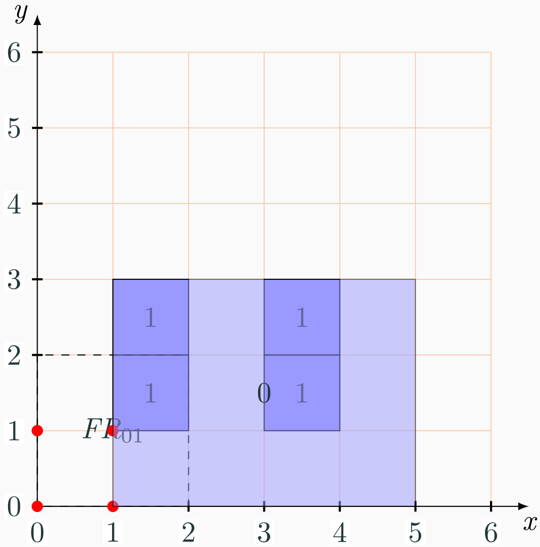
FORBIDDEN REGIONS



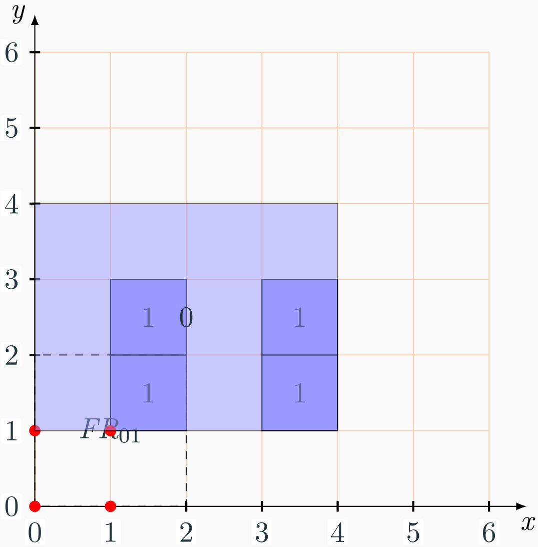
FORBIDDEN REGIONS



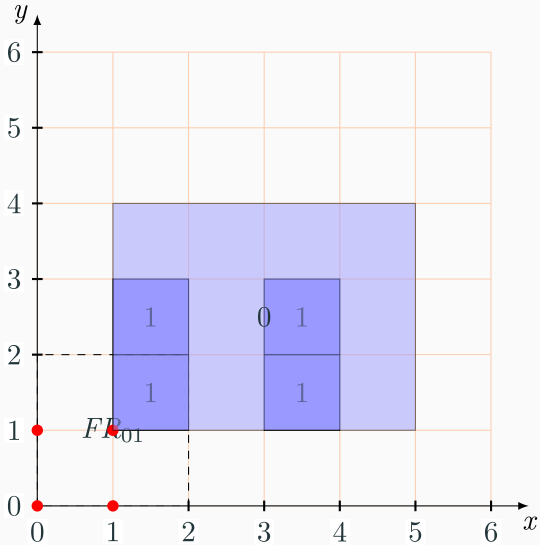
FORBIDDEN REGIONS



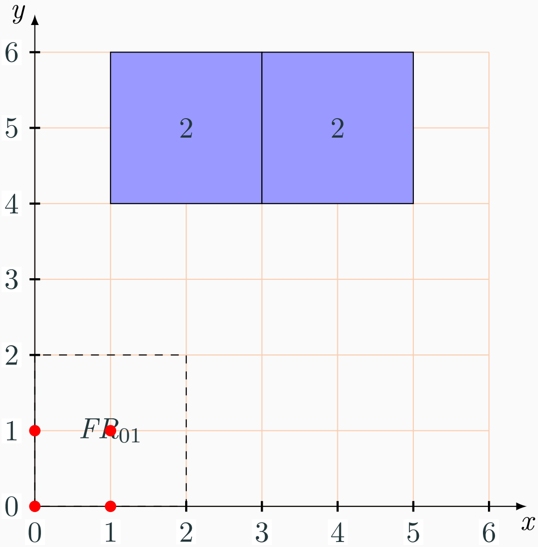
FORBIDDEN REGIONS



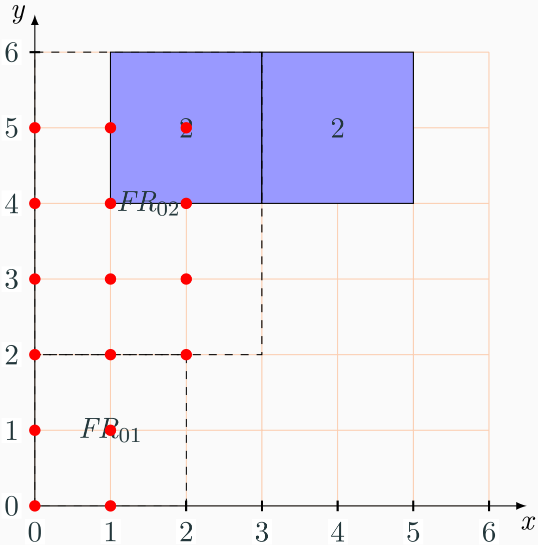
FORBIDDEN REGIONS



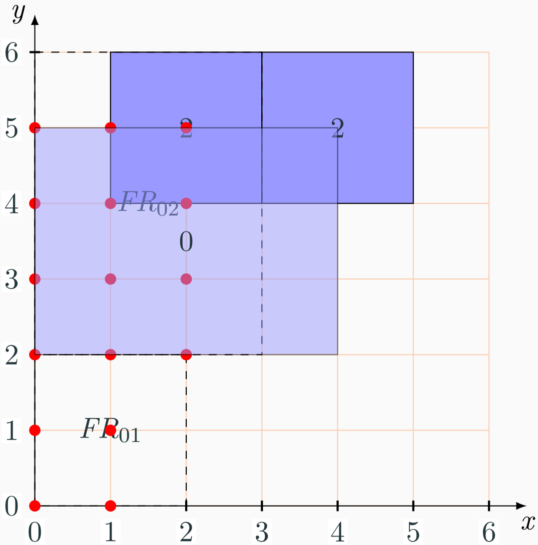
FORBIDDEN REGIONS



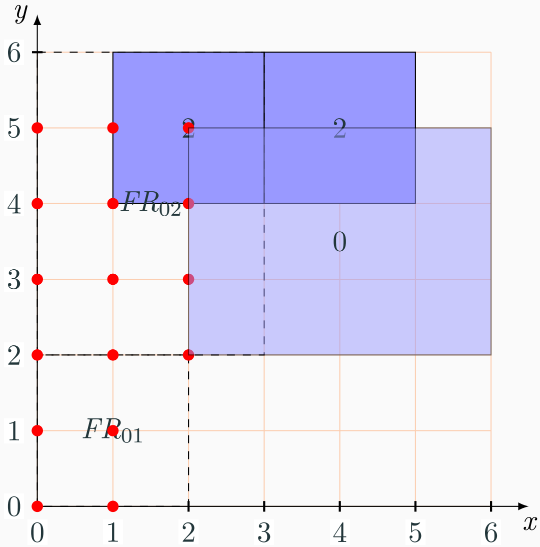
FORBIDDEN REGIONS



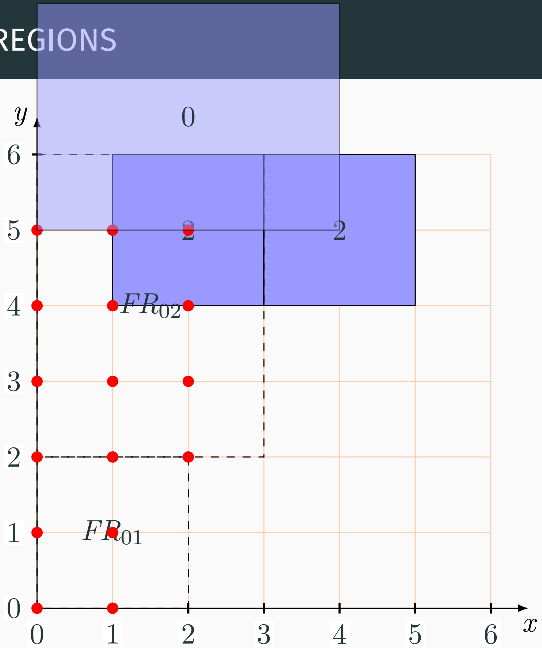
FORBIDDEN REGIONS



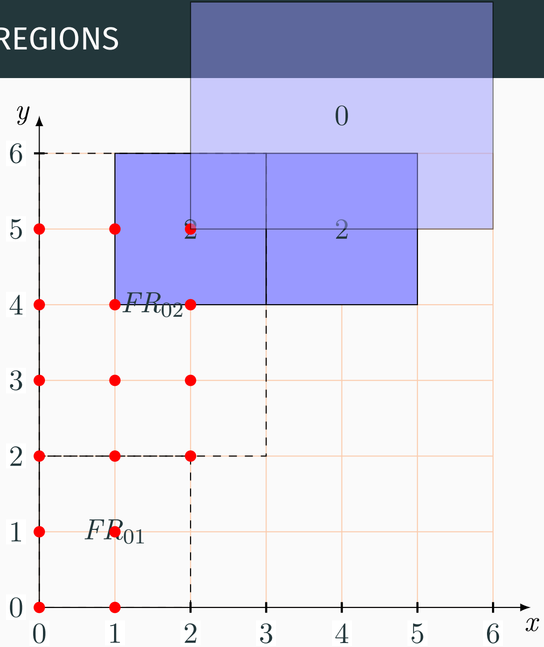
FORBIDDEN REGIONS



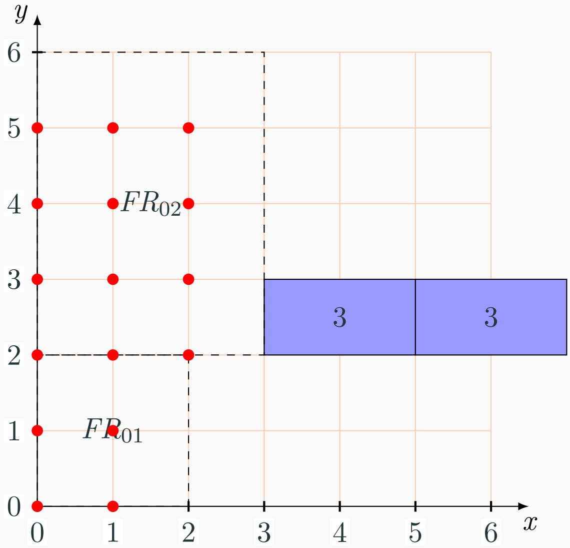
FORBIDDEN REGIONS



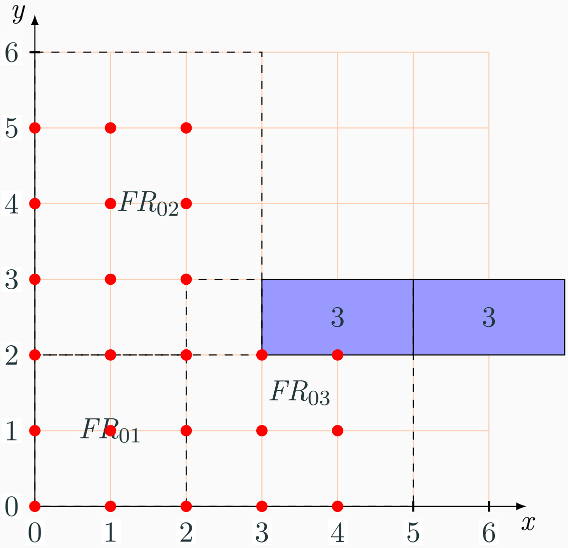
FORBIDDEN REGIONS



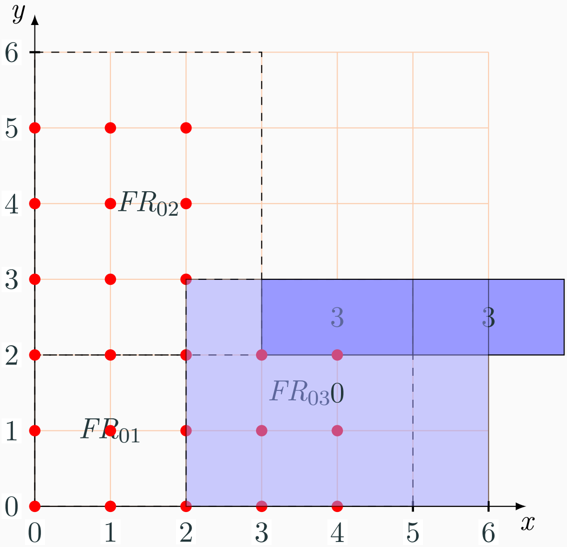
FORBIDDEN REGIONS



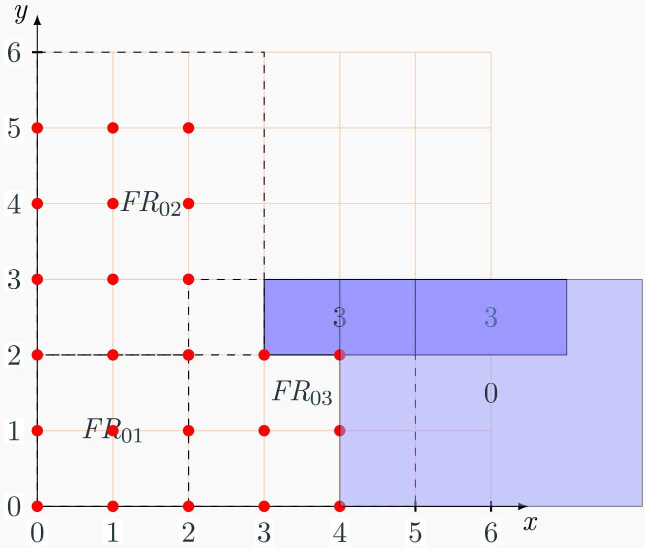
FORBIDDEN REGIONS



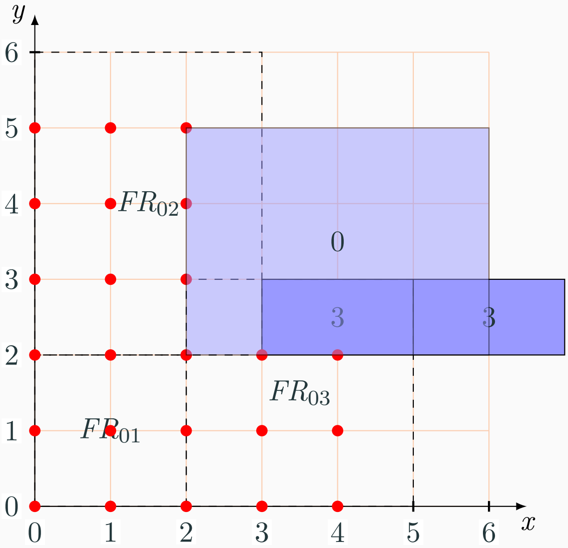
FORBIDDEN REGIONS



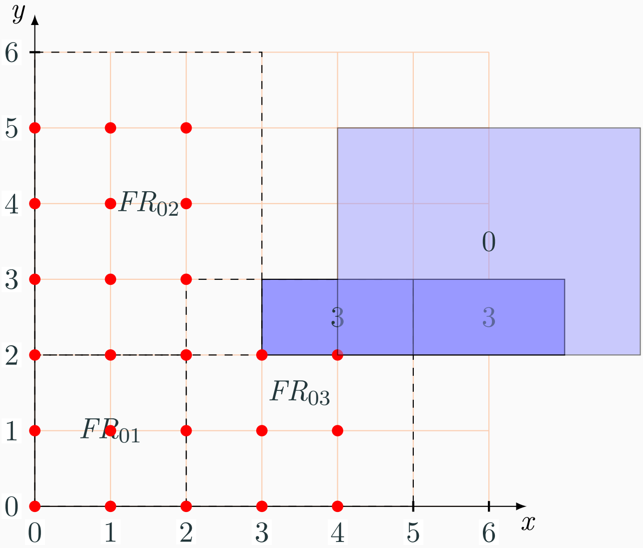
FORBIDDEN REGIONS



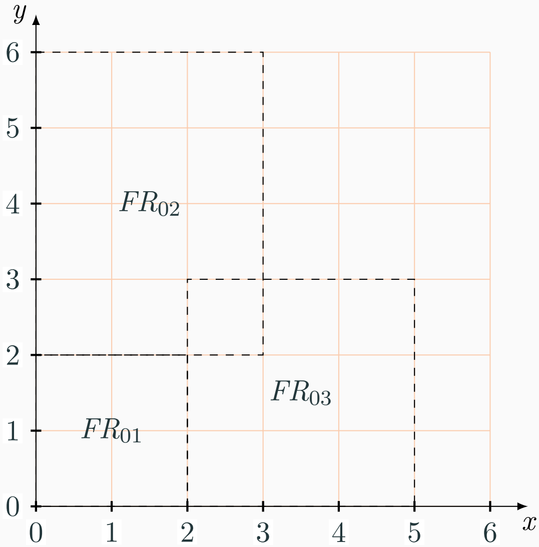
FORBIDDEN REGIONS



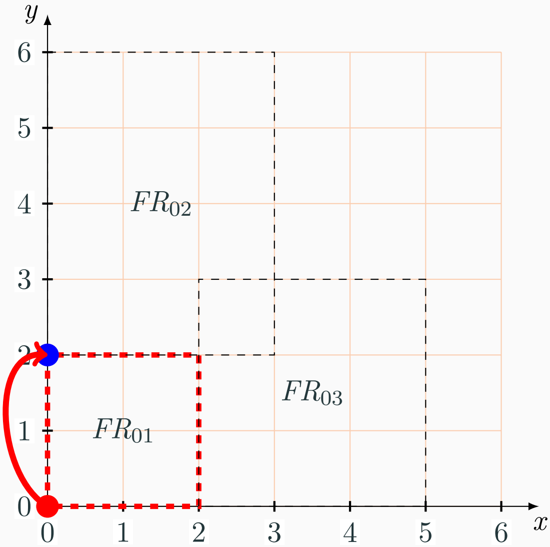
FORBIDDEN REGIONS



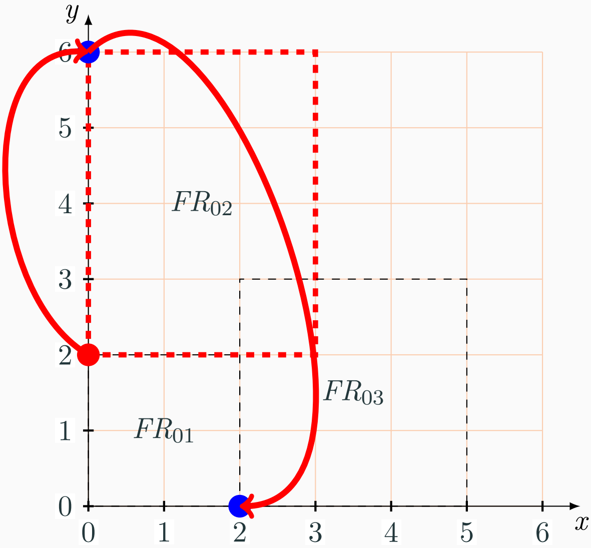
SWEEP ALGORITHM



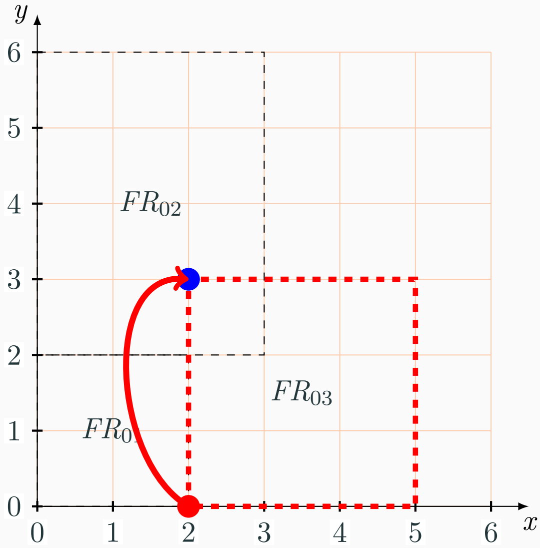
SWEEP ALGORITHM



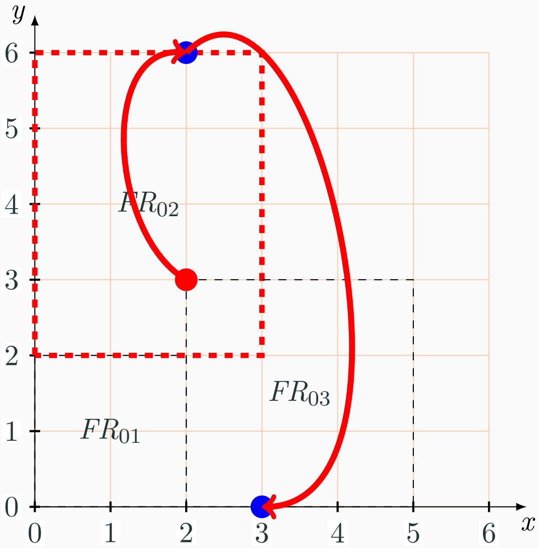
SWEEP ALGORITHM



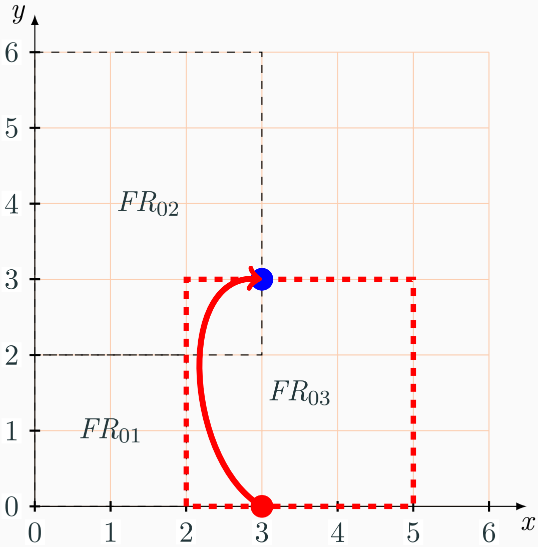
SWEEP ALGORITHM



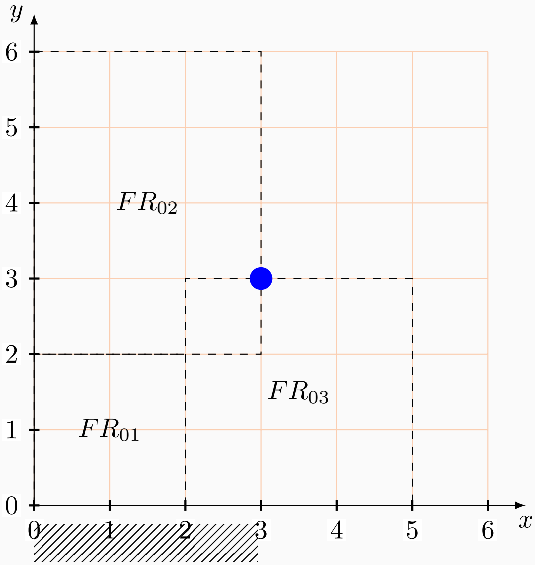
SWEEP ALGORITHM



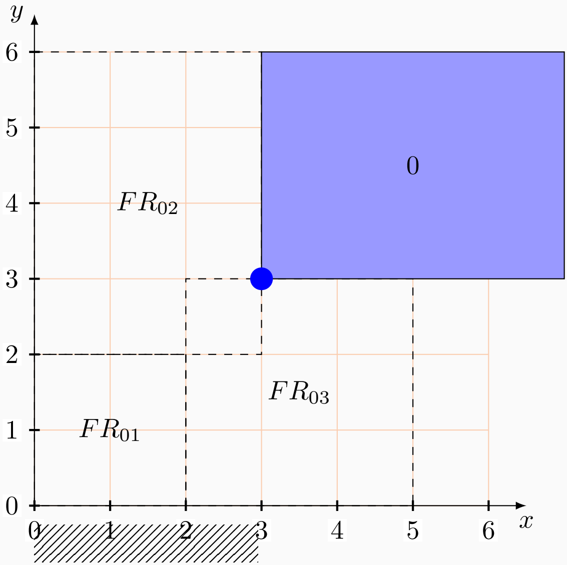
SWEEP ALGORITHM



SWEEP ALGORITHM



SWEEP ALGORITHM



EVALUATION

EVALUATION SETUP

- The best DIFFN configuration against Gecode's NOOVERLAP propagator.

EVALUATION SETUP

- The best DIFFN configuration against Gecode's NOOVERLAP propagator.
- Microbenchmarks and Macrobenchmarks

EVALUATION SETUP

- The best DIFFN configuration against Gecode's NOOVERLAP propagator.
- Microbenchmarks and Macrobenchmarks
- Measured each instance 5 times.

EVALUATION SETUP

- The best DIFFN configuration against Gecode's NOOVERLAP propagator.
- Microbenchmarks and Macrobenchmarks
- Measured each instance 5 times.
- Number of failures to evaluate propagation strength

RESEARCH QUESTIONS

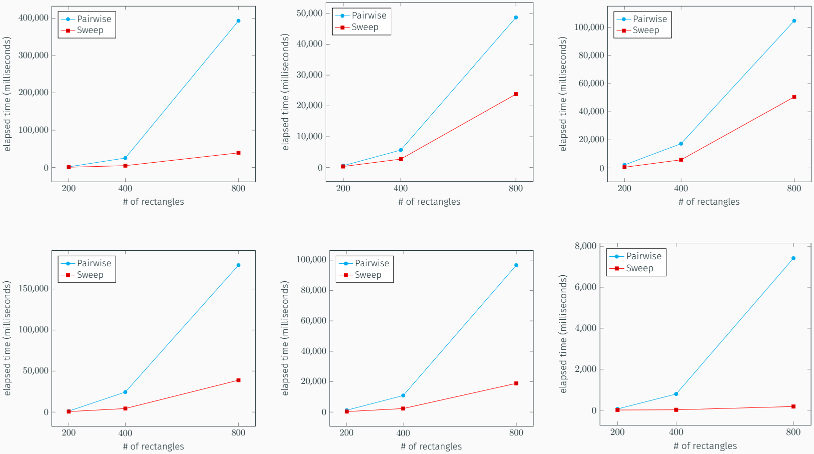
1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?

RESEARCH QUESTIONS

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?
2. What is the performance of the sweep-based algorithm compared to the pairwise algorithm?

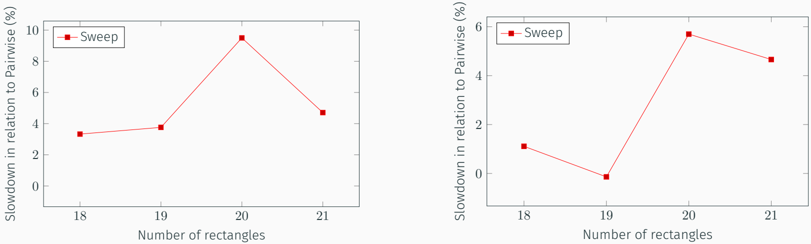
RESULTS

RESULTS - MICROBENCHMARKS



Elapsed time for the microbenchmarks. (Lower is better)

RESULTS - MACROBENCHMARKS: RECTANGLE PACKING PROBLEM



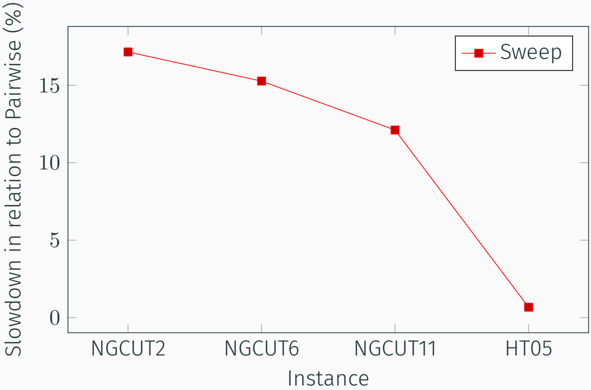
(a) Not considering the 1-unit square.

(b) Considering the 1-unit square.

Slowdown percentages for sweep relative to the pairwise implementation. (Lower is better)

Constraints: DIFFN, CUMULATIVE, arithmetical constraints.

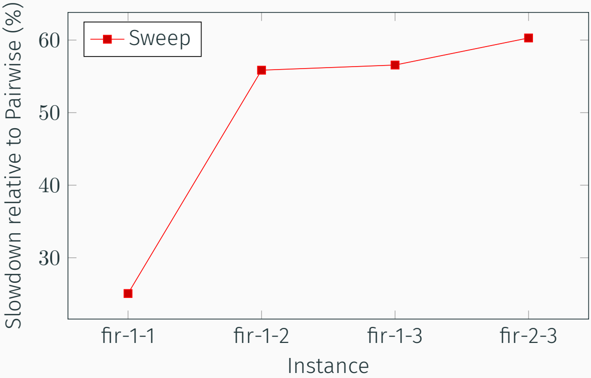
RESULTS - MACROBENCHMARKS: 2D STRIP PACKING



Slowdown percentages for sweep relative to the pairwise implementation. (Lower is better)

Constraints: DIFFN, CUMULATIVE, MEMBER, LEX_LESS.

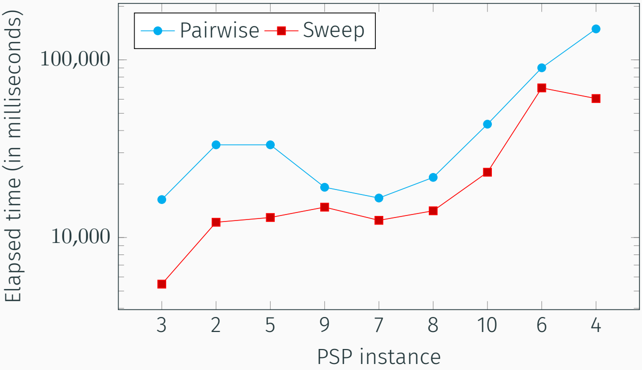
RESULTS - MACROBENCHMARKS: FILTER



Elapsed time (ms) for different instances of the Filter benchmark. (Lower is better)

Constraints: Diffn, arithmetical constraints.

RESULTS - MACROBENCHMARKS: PERFECT SQUARE PACKING



Elapsed times (ms) for the Perfect Square Packing benchmark. (Lower is better)

Constraints: DIFFN, arithmetical constraints.

DISCUSSION

DISCUSSION

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?

DISCUSSION

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?
    - Depends on the problem/model

DISCUSSION

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?
    - Depends on the problem/model
        - **PSP:** Between 1.47 to 3.38 times more.
        - **Micro:** Sweep 0 failures, Pairwise up to 61421 failures.
        - **Other:** No significant difference

DISCUSSION

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?
    - Depends on the problem/model
        - **PSP:** Between 1.47 to 3.38 times more.
        - **Micro:** Sweep 0 failures, Pairwise up to 61421 failures.
        - **Other:** No significant difference
2. What is the performance of the sweep-based algorithm compared to the pairwise algorithm?

DISCUSSION

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?
    - Depends on the problem/model
        - **PSP:** Between 1.47 to 3.38 times more.
        - **Micro:** Sweep 0 failures, Pairwise up to 61421 failures.
        - **Other:** No significant difference
2. What is the performance of the sweep-based algorithm compared to the pairwise algorithm?
    - Identical pruning $\Rightarrow$ slowdowns between 2 to 60 %

DISCUSSION

1. How much extra pruning does the sweep-based algorithm achieve compared to the pairwise algorithm?
    - Depends on the problem/model
        - **PSP:** Between 1.47 to 3.38 times more.
        - **Micro:** Sweep 0 failures, Pairwise up to 61421 failures.
        - **Other:** No significant difference
2. What is the performance of the sweep-based algorithm compared to the pairwise algorithm?
    - Identical pruning $\Rightarrow$ slowdowns between 2 to 60 %
    - At least 1.47 times more pruning $\Rightarrow$ Sweep outperforms Pairwise

WHAT'S NEXT?

FUTURE WORK

- Extend the implementation to support variable lengths
- Evaluate performance in 3 dimensions

THANK YOU!