# Project Presentation

Multi Agent Pathfinding through Plan Merging

By
Tom Schmidt
Hannes Weichelt
Julian Bruns

# General Setting

- Warehouse logistics
- Simulation using Asprilo
- Automatic planning of robot tasks
- Focus : Robot Movements (M-Domain)
- Problem Setting:
  - Informed Asprilo approach for big instances **too time inefficient**
  - Calculate a **Plan for each single Robot** independently
  - **Merge** them together

# Merging via Iterative approach

Core Ideas:

- Detecting vertex and edge collisions
- Changing paths via wait or dodge maneuver before the collision
- "Changing paths" means creating a new set of positions and making older ones invalid
- Positions of the new path have a higher "CONFLICT_DEPTH" - variable

# Vertex collision

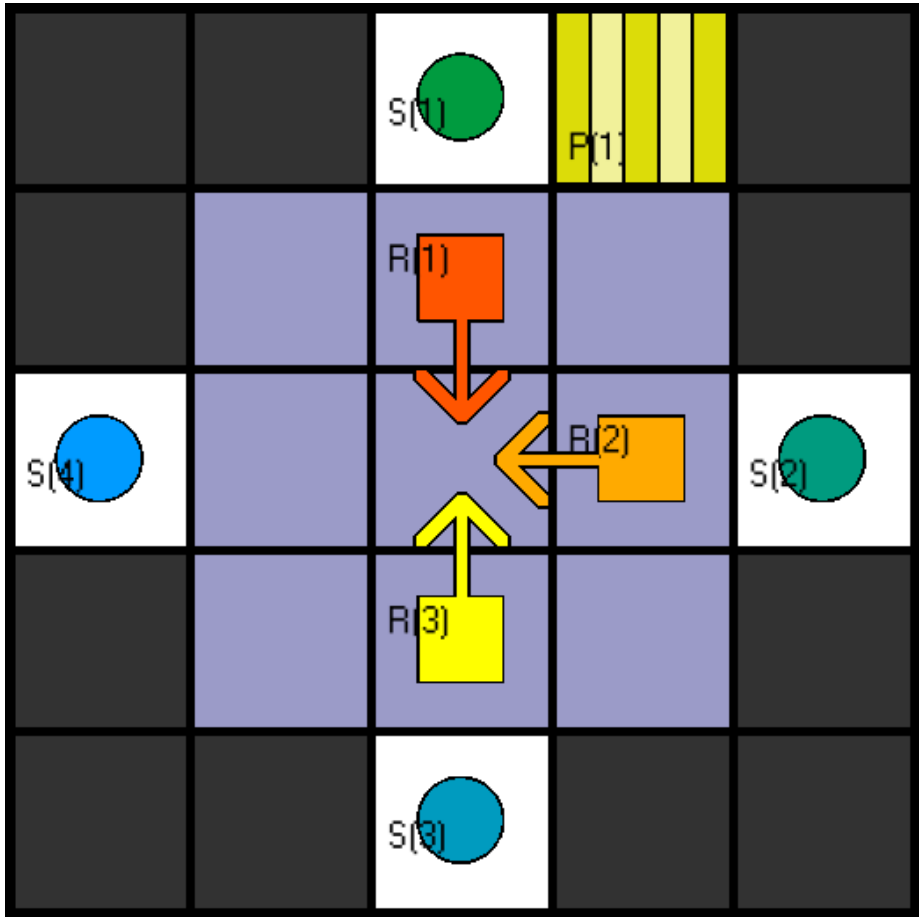
- Detection: Two different robots have same position at same time
- Assign the “wait”-statement to one of the involved robots
- Generate a new path for the robot with the wait statement
- Question: which of the involved robots should be assigned the “wait”-statement?

# Vertex collision

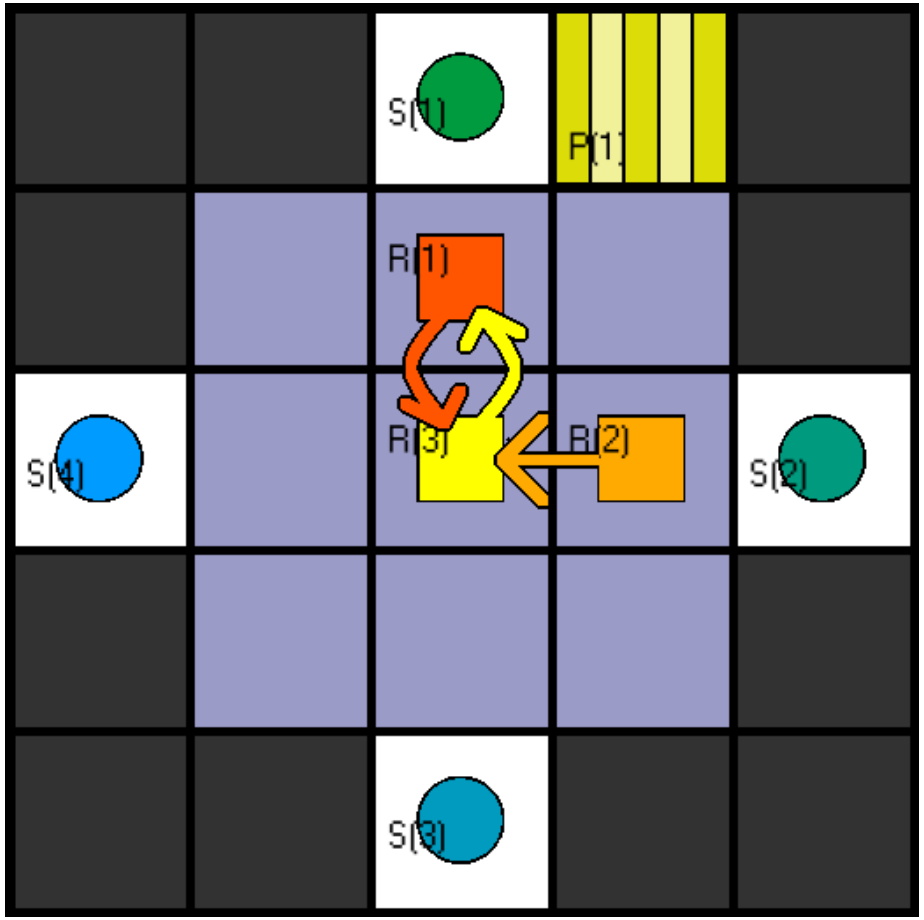
Multiple Ideas for the assignment of “wait”-statements:

- **First Idea:** Robot with the lowest ID waits.
- **Problem:** Collisions with more than 2 robots can cause problems
- This can be seen in the following example:

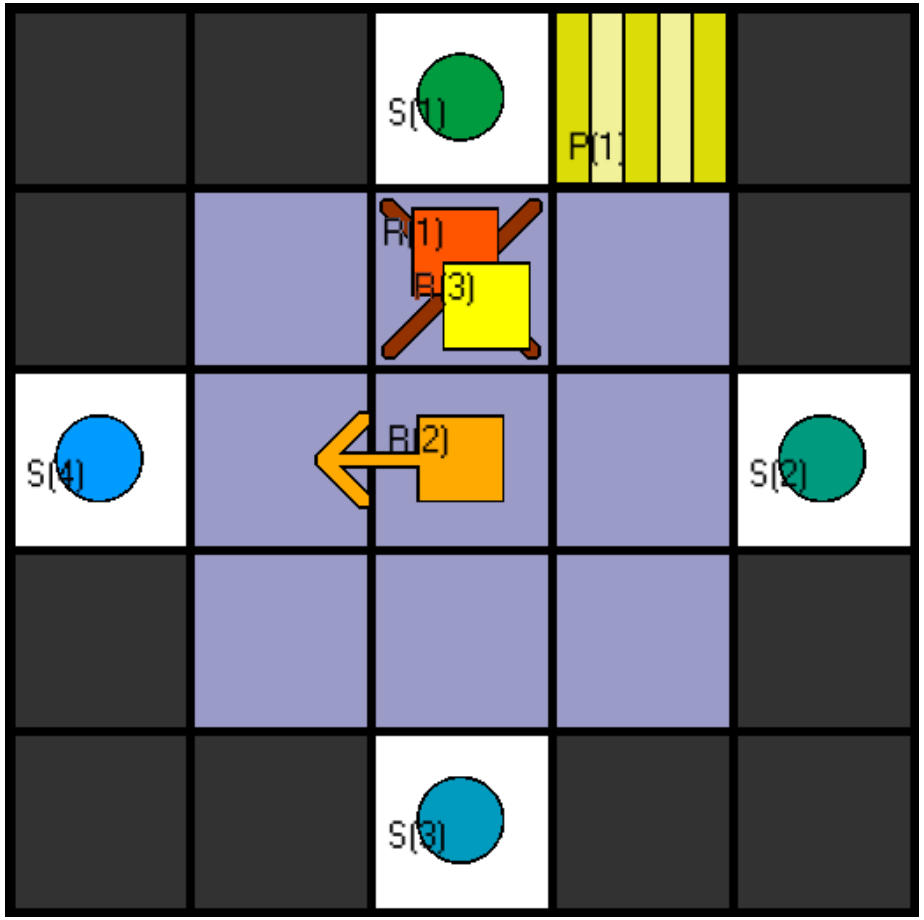
# Example (of a vertex collision problem)



# Example (of a vertex collision problem)



# Example (of a vertex collision problem)



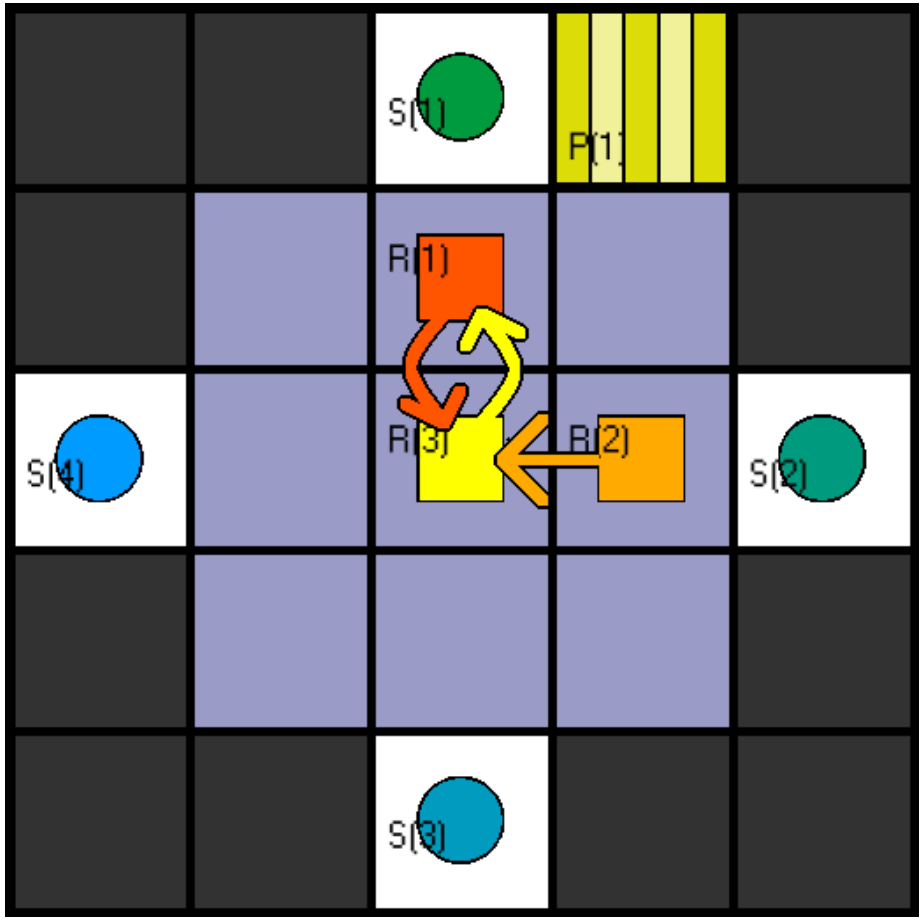
# Vertex collision

Multiple Ideas for the assignment of “wait”-statements:

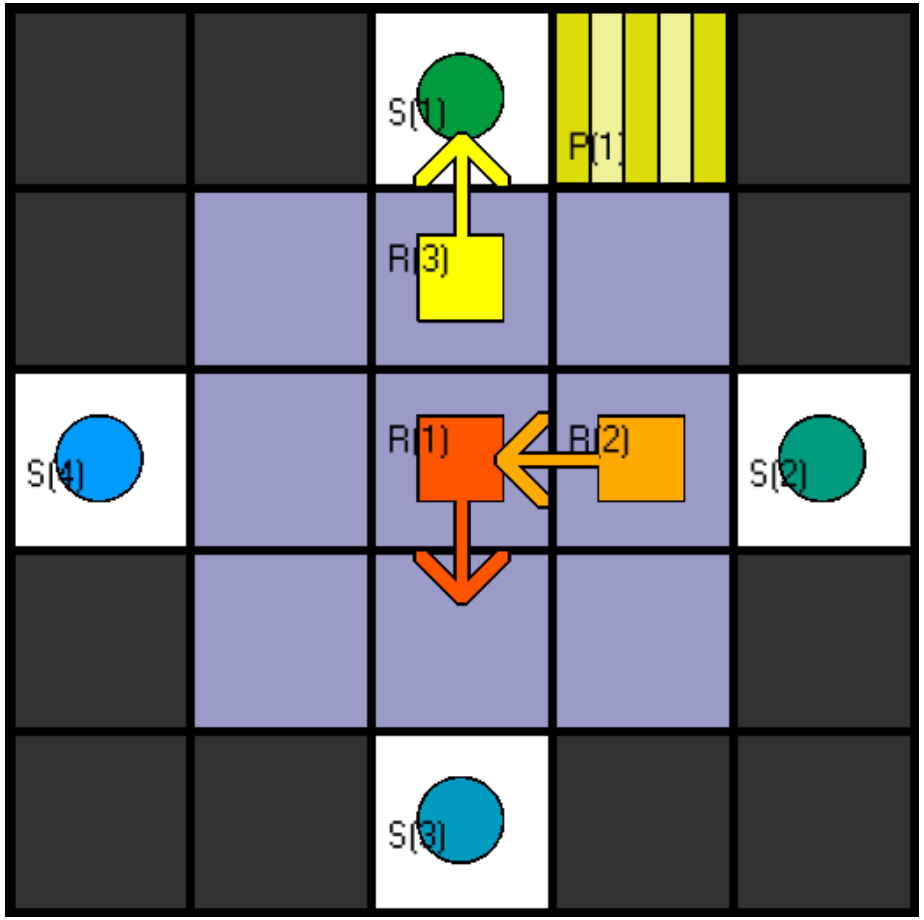
- First Idea: Robot with the lowest ID waits.
- Problem: Collisions with more than 2 robots can cause problems

- **Final Idea:**
  - Differentiating between if 2,3 or 4 robots collide
  - depending on that number, let a maximum of 1,2 or 3 robots wait.
  - Add constraint that makes vertex collisions impossible

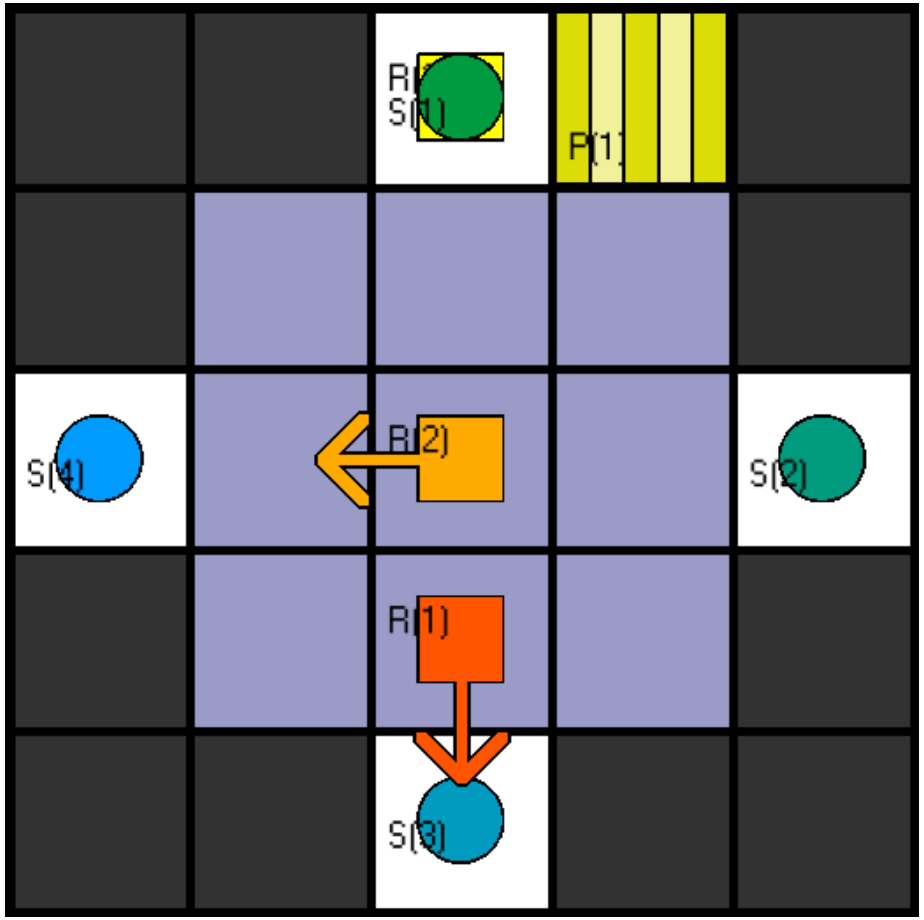
# Vertex collision problem solved



# Vertex collision problem solved



# Vertex collision problem solved



# Edge collision

- Detection: check if two robots sit next to each other and switch their positions
- Choose one of the possible dodge-statements for one of the involved robots
- Create constraint that makes dodge statements, that move robots into a wall, invalid

# Evaluation

- Can solve many simple benchmarks
- Two major problems:
  - Collision depth can increase computation time dramatically
  - Merger only changes the path of a robot slightly before a collision occurs
- Complex benchmarks can lead to unsatisfiability or a very high computation time
- Conclusion: Try a different approach

# Merging as Constraint Satisfaction Problem (CSP)

- Set of variables whose state must satisfy several constraints
- Subproblems:
  - Vertex conflicts
  - Edge conflicts
  - Out of bounds moves
  - Compliance with time horizon

# Functionality of the CSP Merger

- 2 possible actions per robot at each time step
  - Wait
  - Dodge
- Generate all possible combinations of waits and dodges using available time
- Constrain new plans:
  - Vertex constraint
  - Edge constraint
  - Out of bounds constraint
  - Time horizon/ goal constraint

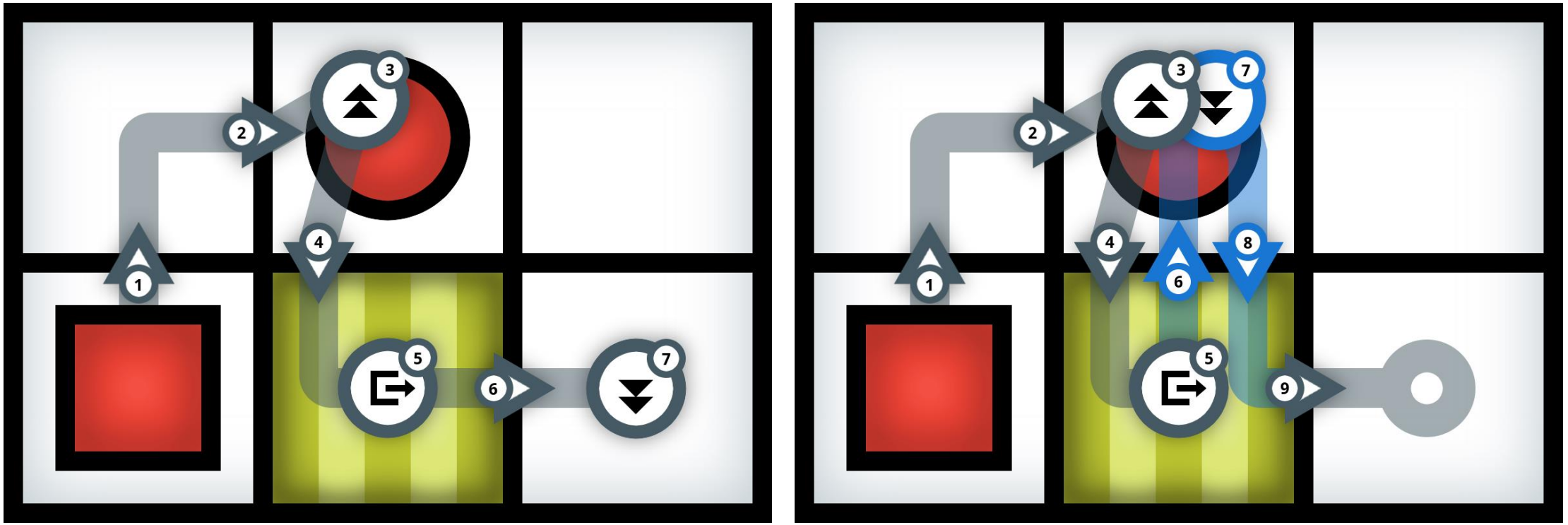
# Evaluation of CSP Merger

- Can solve almost any benchmark
- Very good scalability
- Still 2 problems:
  - Multi-dodge problems unsatisfiable (e.g., benchmark 20)
  - Performance depends on chosen horizon -> Python script
- Simple code and functionality -> easy modification, additional features:
  - Locking
  - A-Domain

# Extension for A-Domain

- New statement adomain/0
- Modify original plan
- Apply CSP merger
- Adjust horizon constraint
- Add new constraints:
  - No dodges into shelves
  - No pickup of carried shelves

# Plan modification



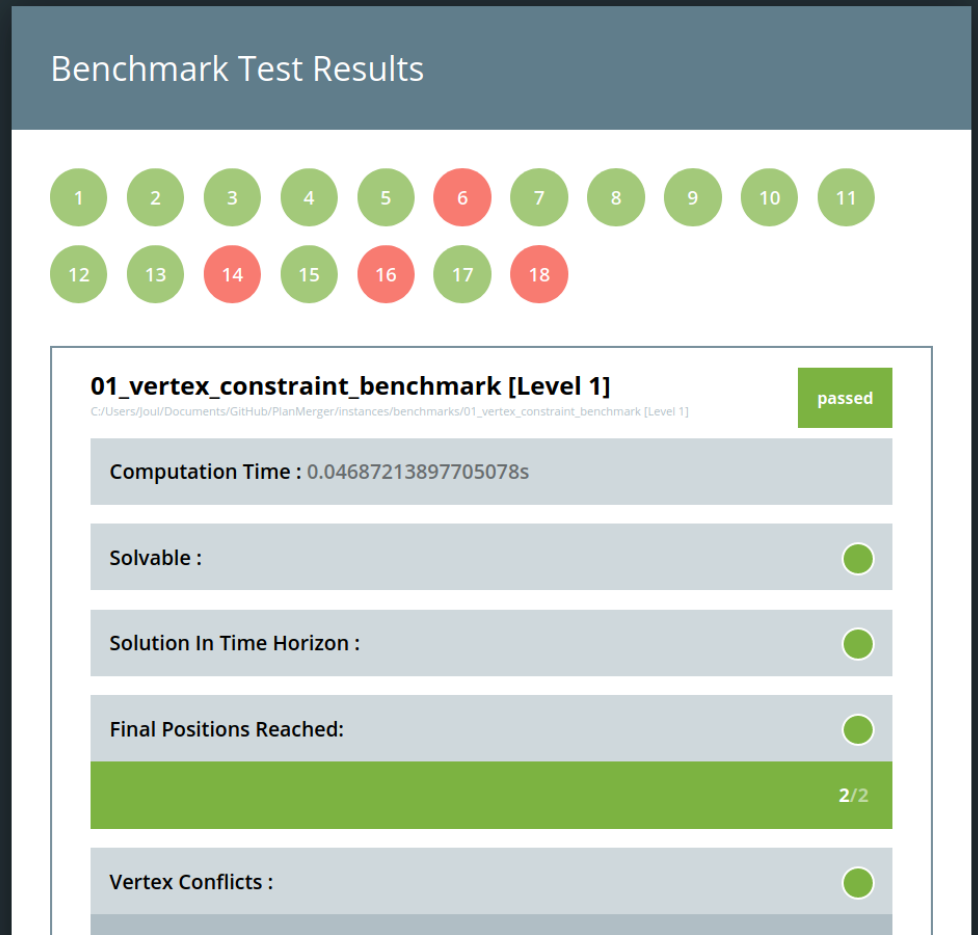
# Extension for A-Domain

- New statement adomain/0
- Modify original plan
- Apply CSP Merger
- Adjust horizon constraint
- Add new constraints:
  - No dodges into shelves while carrying a shelf
  - No pickup of carried shelves

# Example

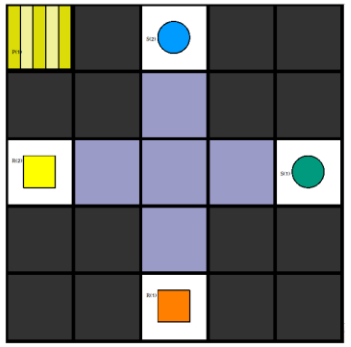
# Benchmark Engine

- script to evaluate benchmarks
- wirtten in python
- features :
  - test for all 4 key consraints
  - solve benchmarks using the clingo python API (tracking time, …)
  - generate Random Benchmarks
  - still a work in progress

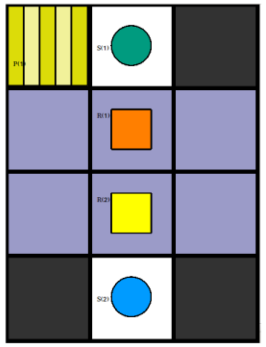


# Results

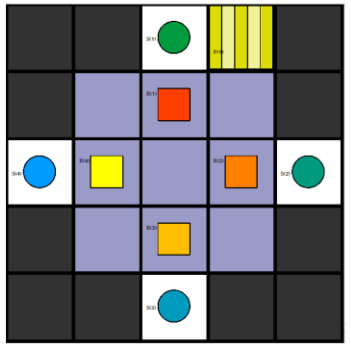
- internal comparison
  - CSP Merger
    - more powerful
    - better scalable
    - Faster



Benchmark 01

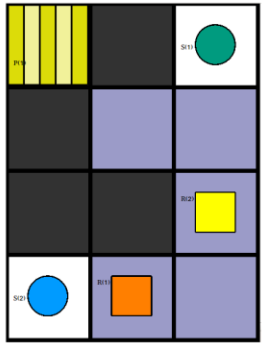


Benchmark 02



Benchmark 03


Benchmark 06



Benchmark 20

solvable
unsolvable

| Merger | Benchmark 01 | Benchmark 02 | Benchmark 03 | Benchmark 06 | Benchmark 20 |
|---|---|---|---|---|---|
| Iterative | 0.0468s | 0.0468s | 3.6939s | 0.6623s | 0.8285s |
| CSP | 0.0192s | 0.0181s | 0.0829s | 0.1198s | 0.0578s |

# Results

- external comparison
  - compare to other groups

| Benchmark Name | Instance 1 | Instance 5 | Insatnce 6 | Instance 7 | bench_test_2 | bench_test_3 | bench_test_16_mod_1 | benchmark-5 | benchmark-6 | benchmark-42 | benchmark-51 | B_03 | B_05 | B_R1 | B_R2 | benchmark_1 | benchmark_2 | benchmark_3 | benchmark_4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Gruppe | Adrian | | | | Max + Marcus | | | Niklas + Marius | | | | TJH | | | | Tarek | | | |
| Num. Robots | 2 | 4 | 2 | 8 | 2 | 2 | 4 | 4 | 8 | 5 | 6 | 4 | 3 | 50 | 30 | 3 | 2 | 3 | 2 |
| Map-Dimension X | 5 | 4 | 7 | 8 | 4 | 4 | 5 | 5 | 4 | 10 | 15 | 5 | 4 | 15 | 40 | 3 | 7 | 6 | 9 |
| Map-Dimension Y | 3 | 3 | 2 | 8 | 4 | 5 | 5 | 8 | 7 | 10 | 15 | 5 | 5 | 15 | 40 | 3 | 5 | 6 | 2 |
| Max Horizon | 5 | 3 | 14 | 10 | 7 | 4 | 9 | 11 | 15 | 10 | 21 | 10 | 10 | 40 | 100 | 5 | 19 | 10 | 16 |
| c_nr [for Adrian] | 1 | 3 | 9 | 3 | 1 | 1 | 5 | 14 | 6 | 1 | 1 | 4 | 1 | 1 | 1 | 8 | 10 | 2 | 4 |
| rm [for Tarek] | 4 | 10 | 9 | 10 | 2 | 3 | 4 | 7 | 5 | 2 | 3 | 4 | 3 | 5 | 5 | 4 | 16 | 6 | 8 |
| | | | | | | | | | | | | | | | | | | | |
| Merger of Group : | | | | | | | | | | | | | | | | | | | |
| **Adrian** | | | | | | | | | | | | | | | | | | | |
| **Correct** | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | FALSCH | FALSCH | FALSCH | WAHR | WAHR |
| Satisfiable | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | WAHR | WAHR | WAHR | WAHR | KILLED | KILLED | FALSCH | FALSCH | WAHR | WAHR |
| Timeout | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | KILLED | KILLED | FALSCH | WAHR | FALSCH | FALSCH |
| Time (in s) | 0,0114 | 0,0607 | 123,7117 | 35,6638 | 0,0122 | 0,0067 | 3,2160 | 13,4690 | 358,7314 | 3,1784 | 218,2776 | 5,3833 | 0,0354 | - | - | 0,3224 | 1800,0000+ | 0,1982 | 0,5161 |
| **Max + Marcus** | | | | | | | | | | | | | | | | | | | |
| **Correct** | WAHR | FALSCH | FALSCH | FALSCH | WAHR | FALSCH | FALSE (M) | FALSE (H+) | FALSCH | FALSE (H+) | FALSCH | FALSE (M) | FALSE (M) | FALSCH | FALSCH | FALSCH | FALSCH | FALSE (M) | FALSE (M) |
| Satisfiable | WAHR | FALSCH | WAHR | KILLED | WAHR | WAHR | WAHR | WAHR | KILLED | WAHR | KILLED | WAHR | WAHR | KILLED | KILLED | FALSCH | FALSCH | WAHR | WAHR |
| Timeout | FALSCH | FALSCH | FALSCH | KILLED | FALSCH | FALSCH | FALSCH | FALSCH | KILLED | FALSCH | KILLED | FALSCH | FALSCH | KILLED | KILLED | FALSCH | FALSCH | FALSCH | FALSCH |
| Time (in s) | 0,2238 | 0,1713 | 1,4714 | - | 0,3973 | 0,0519 | 3,6219 | 25,9768 | - | 540,0421 | - | 8,3662 | 1,4516 | - | - | 0,1227 | 3,7384 | 13,0752 | 4,0606 |
| **Niklas + Marius** | | | | | | | | | | | | | | | | | | | |
| **Correct** | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | FALSCH | FALSCH | FALSCH | WAHR | WAHR | WAHR |
| Satisfiable | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | KILLED | KILLED | FALSCH | WAHR | WAHR | WAHR |
| Timeout | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | KILLED | KILLED | FALSCH | FALSCH | FALSCH | FALSCH |
| Time (in s) | 0,0055 | 0,0070 | 0,0328 | 0,1137 | 0,0142 | 0,0050 | 0,0244 | 0,1260 | 0,5552 | 0,1082 | 0,0938 | 0,0316 | 0,0125 | - | - | 0,0049 | 0,3070 | 0,0336 | 0,0528 |
| **TJH** | | | | | | | | | | | | | | | | | | | |
| **Correct** | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | FALSCH | WAHR | WAHR |
| Satisfiable | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | FALSCH | WAHR | WAHR |
| Timeout | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH |
| Time (in s) | 0,0269 | 0,0171 | 0,0270 | 0,2555 | 0,0399 | 0,0190 | 0,1108 | 0,0888 | 1,7390 | 0,1399 | 2,3586 | 0,0848 | 0,0183 | 87,6203 | 618,3541 | 0,0274 | 1,5511 | 0,0862 | 0,3454 |
| **Tarek** | | | | | | | | | | | | | | | | | | | |
| **Correct** | WAHR | FALSCH | WAHR | FALSCH | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | WAHR | WAHR | FALSCH | FALSCH | WAHR | WAHR | FALSCH | WAHR |
| Satisfiable | WAHR | FALSCH | WAHR | FALSCH | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | WAHR | FALSCH | KILLED | WAHR | WAHR | WAHR | WAHR |
| Timeout | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | FALSCH | KILLED | FALSCH | FALSCH | FALSCH | FALSCH |
| Time (in s) | 0,0186 | 0,0090 | 0,0310 | 1,0490 | 0,0094 | 0,0099 | 0,0173 | 0,0759 | 0,1245 | 0,2054 | 1,9946 | 0,0192 | 0,0099 | 91.175s | - | 0,0143 | 0,2817 | 0,0786 | 0,0281 |

# Results

- external comparison
  - compare to other groups
- very scalable
- only merger that can deal with big benchmarks
- can‘t solve multi dodge benchmarks but everything else
- not always the fastest merger
  - random generation mergers often faster (on smaller benchmarks)

# Further information …

- GitHub Repository:
  - https://github.com/tzschmidt/PlanMerger/tree/main
- Project Report
- Table Link :
  - https://unipotsdamde-my.sharepoint.com/:x:/g/personal/hweichelt_uni-potsdam_de/EY8Sa-6OsTtLqimDGTLBqyEBfQxNp3pl-HRtOqKi2h_Tnw?e=sxDvkr.