Plan merging project

by Anton Rabe, Aaron Bishop and Louis Donath

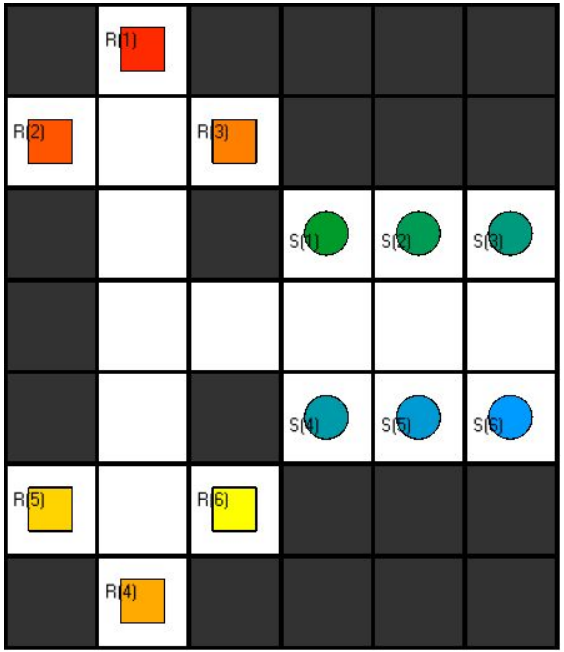
Overview

- The Problem
- Plan merging
- Approaches
- Evaluation
- Conclusion

The Problem

- Given:
  - set of robots
  - set of shelfs
  - plans between robot and shelf for each robot

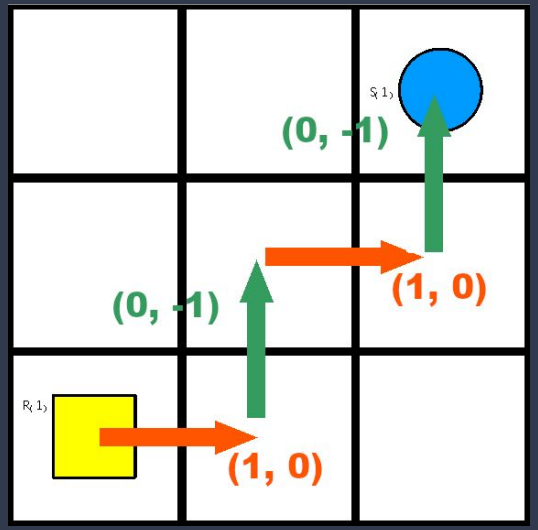
- Wanted:
  - new plans without collisions



Source: https://github.com/warpaint97/plan-merging-project/blob/main/benchmarks/08_lots_of_vertex_collisions/instance/x6_y7_n21_r6_s6_ps0_pr6_u6_o6_N008.png

Plans

```
occurs(object(robot,1),action(move,(1,0)),0).
occurs(object(robot,1),action(move,(0,-1)),1).
occurs(object(robot,1),action(move,(1,0)),2).
occurs(object(robot,1),action(move,(0,-1)),3).
...
```



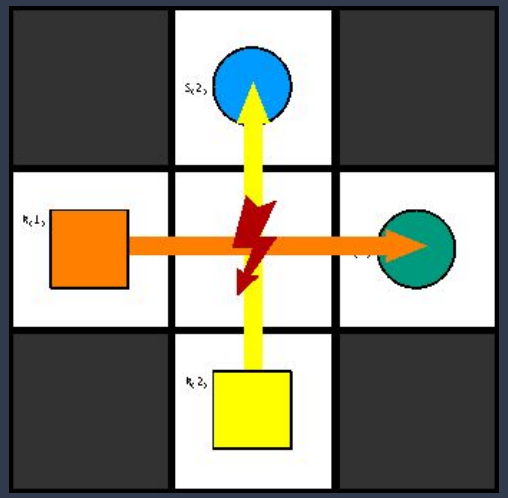
Source: https://github.com/warpaint97/plan-merging-project/blob/main/benchmarks/01_no_conflict/plans/full_plan.lp

From plans to positions

```
occurs(object(robot,1),action(move,(1,0)),0).
occurs(object(robot,2),action(move,(0,-1)),0).
occurs(object(robot,1),action(move,(1,0)),1).
occurs(object(robot,2),action(move,(0,-1)),1).

move(robot(R),(DX,DY),T) :-
    occurs(object(robot,R),action(move,(DX,DY)),T).

position(R,(X+DX,Y+DY),T+1) :-
    move(R,(DX,DY),T), position(R,(X,Y),T).
```

Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/waiting/choice_rules/merger_w_cr4.lp

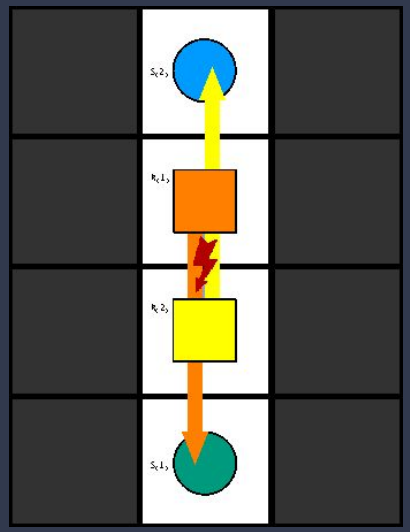
Vertex Collision



```
vertex_collision(C,T,L):- position_(R,C,T),
                          position_(R',C,T),
                          R!=R'.
```

Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/waiting/choice_rules/merger_w_inc.lp

Edge Collision



```
edge_collision(R,R',T+1) :- position(R,C,T), position(R,C',T+1),
                            position(R',C',T), position(R',C,T+1),
                            R!=R', C!=C'.
```

Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_ps_1w.lp

The idea behind plan merging

- Why not using an algorithm that explores the whole search space?
  - It could guarantee optimality
  - It would not require initial plans
  - disadvantage: exponential runtime

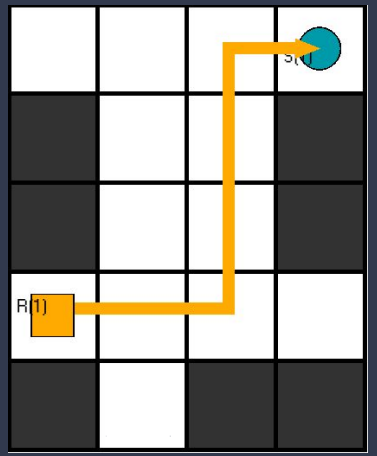
- Plan merging
  - Only collisions need to be considered
  - Our approach:
    - no leaving of original plans
    - plan switching allowed
    - switching and waiting multiple times
  - Runtime stays low

Approaches

Waiting → Positions

```
position_(R,C,T+1,N+1) :- position_(R,C,T,N), wait(R,_,T).
position_(R,C',T+1,N) :- position(R,C',T-N+1), not wait(R,_,T),position_(R,C,T,N).
```

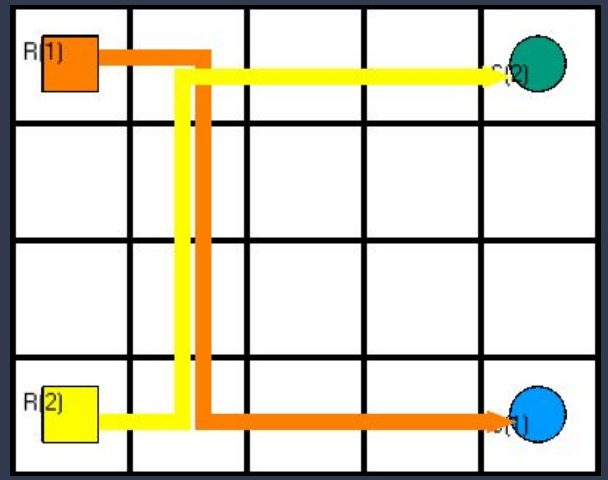
```
{wait(R,C,T)} :- position(R,C,T).
```



Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_ps_big_final.lp

Switching → Positions

```
position_(R,C',T+1,R'',D+1) :- position(R'',C',D+1),
                               position_(R,C,T,R',N),
                               switch(R',R'',N,D).
position_(R,C',T+1,R',N+1) :-  position(R',C',N+1),
                               position_(R,C,T,R',N),
                               not switch(R',_,N,_).

{switch(R,R',T,T')}:- intersect(R,R',C,T,T').
```

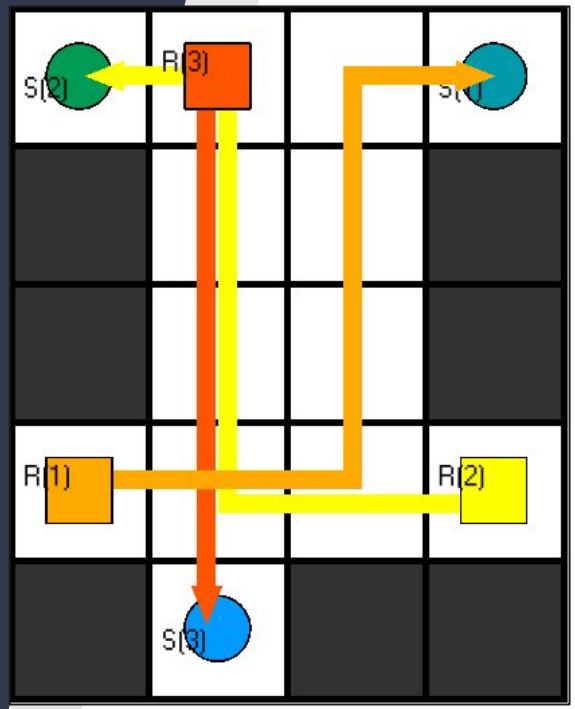


Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_ps_big_final.lp

The incremental approach

1. limit possibilities for assigning switching/ waiting positions
2. define a heuristic for optimization
3. Recursively execute the logic program on previous output until a fixed point is reached
4. Only select the optimal model at each step (greedy approach)

Plan-switching

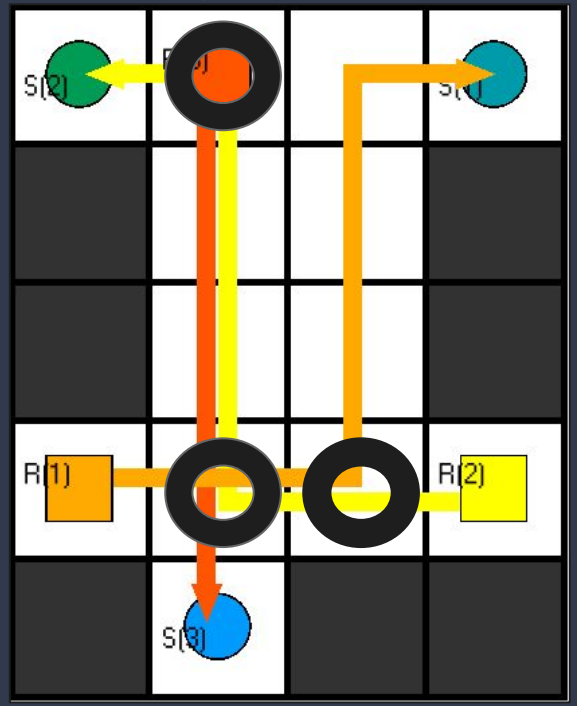


Source: https://github.com/warpaint97/plan-merging-project/tree/main/benchmarks/16_multiple_edge_collisions

Incremental plan switching

- Limit switching positions to earliest intersections of plans

```
switch(R,R',T,T'):- switch(R,R'),
                    earliest_intersect(R,R',C,T,T').
```

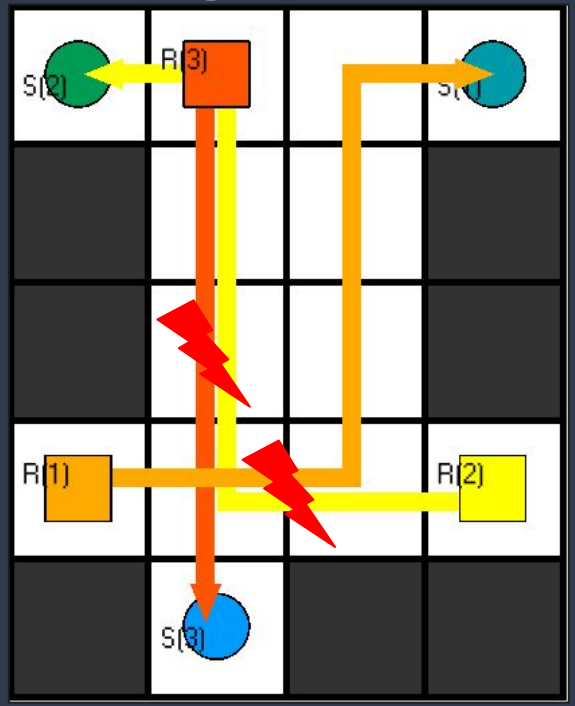


Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_ps_big_final.lp

Incremental plan switching

- Only allow pairwise switching if two robots have a collision

```
{switch(R,R')} :- collision(R,R',T).
switch(R',R) :- switch(R,R').

:- switch(R,R'), switch(R,R''), R'!=R''.
```



Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_ps_big_final.lp

incremental plan switching

- Only remove edge collision-like collisions

```
edge_collision(R,R',T+1) :- position(R,C,T), position(R,C',T+1), position(R',C',T),
                            position(R',C,T+1), R!=R'.
fake_edge_collision(R,R',T) :- position(R,C,T), position(R,C',T+1), position(R',C',T-1),
                               position(R',C,T), R!=R'.
overtake_edge_collision(R,R',T,T') :- last_position(R,C,T), position(R',C,T'), R!=R', T'>T.
```

- To prevent generating new edge collisions during waiting remove all edge-collision like intersections.

```
edge_collision_possible(R,R',T) :- intersect(R,R',C,T,T'), intersect(R,R',C',T+1,T'-1).
```

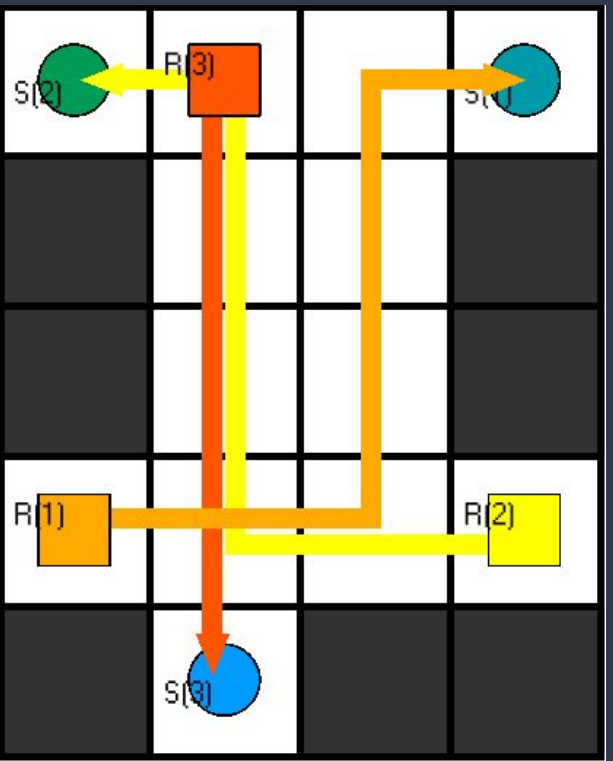
Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_ps_big_final.lp

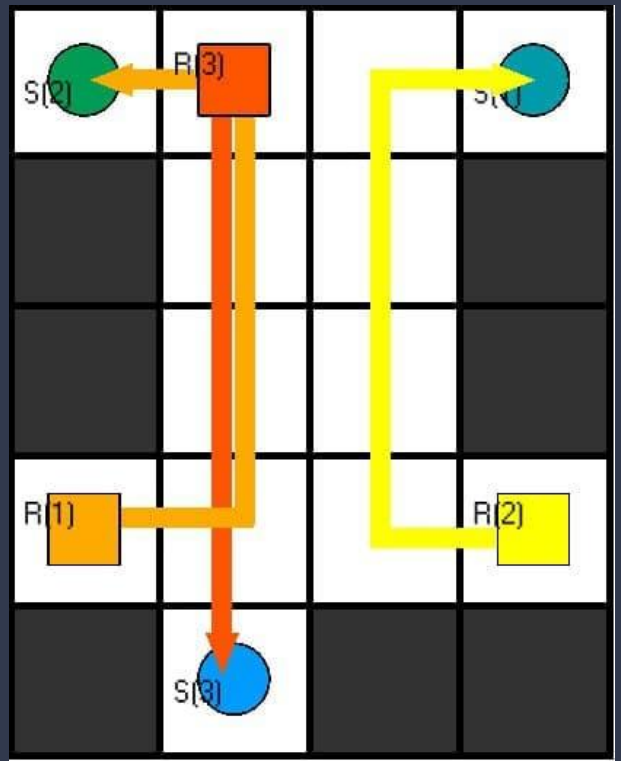
incremental plan switching

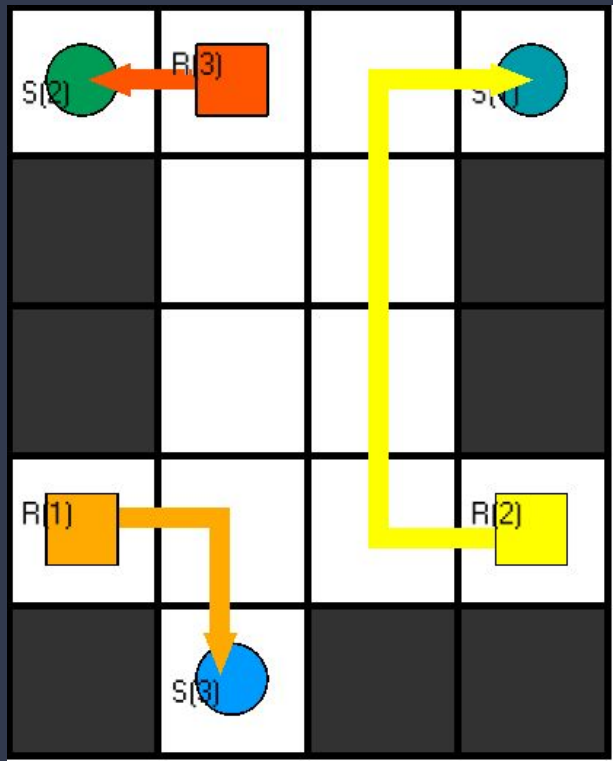
- Minimize horizon and number of positions in final plan

```
horizon(T_MAX) :- T_MAX == #max{T : position_(R,C,T,X,Y)}.
#minimize{T@5 : horizon(T)}.

#minimize {1@4,T,R : position_(R,C,T,R',D)}.

#minimize {1@3,R,R' : final_collision(R,R',T)}.

#minimize {1@1,R,R' : final_overtake_edge_collision(R,R',T,T')}.

#minimize {1@2,R,R' : final_overtake_edge_collision(R,R',T,T'),
overtake_edge_collision(R,R',T,T')}.
```

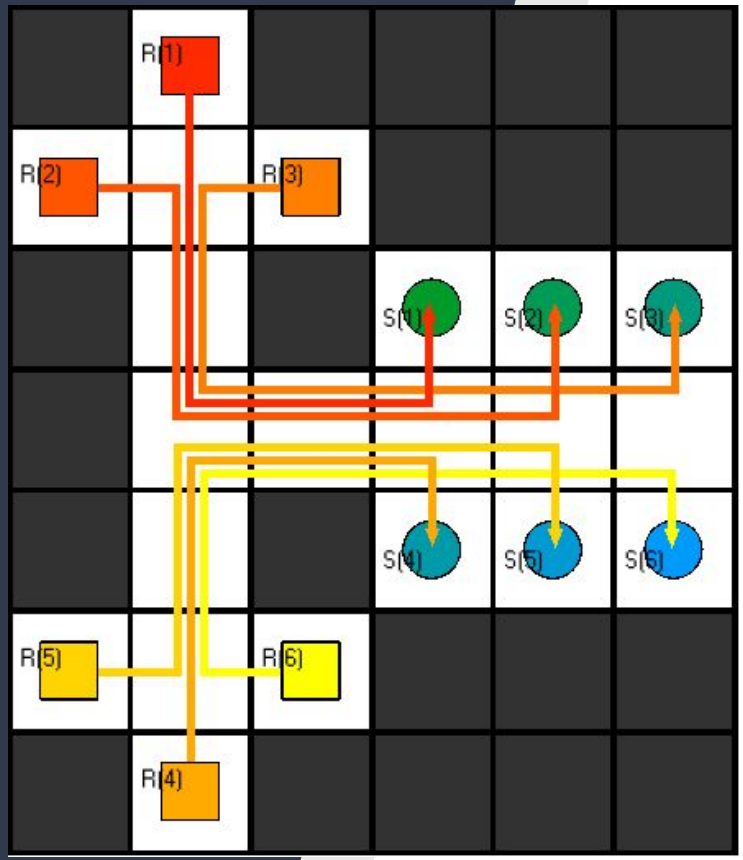
incremental plan switching







waiting



Source: https://github.com/warpaint97/plan-merging-project/tree/main/benchmarks/08_lots_of_vertex_collisions

incremental waiting

Only allow one wait per robot one step before the earliest intersection with another plan.

```
waitable(R,T) :- has_vc(R,R'), position(R,C,T+1),
            position(R',C,T'), R!=R'.
waitable(R,T) :- waitable(R',T), position(R,C,T+1),
               position(R',C,T'), R!=R'.
not_earliest_waitable(R,T) :- waitable(R,T), waitable(R,T'), T>T'.
earliest_waitable(R,T) :- not not_earliest_waitable(R,T),
                          waitable(R,T).
```

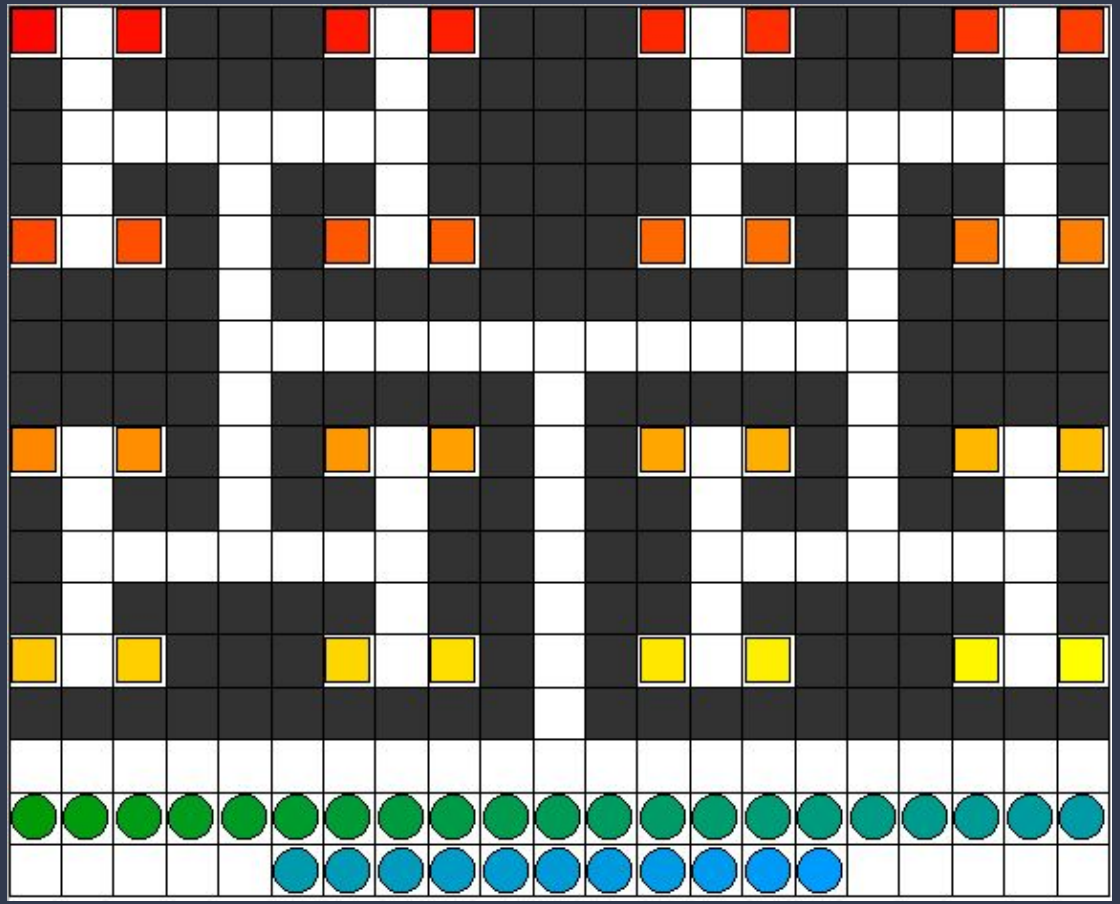
Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_w_inc_final.lp

incremental waiting

Minimize the squared distance between collision and last timestep.

```
vc_cost(R,T,(T_MAX-T)*(T_MAX-T)) :- position_(R,C,T),
                                    position_(R',C,T),
                                    R!=R', horizon(R,T_MAX).

#minimize {C@2,R,T : vc_cost(R,T,C)}.
```

Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_w_inc_final.lp



Source: https://github.com/warpaint97/plan-merging-project/tree/main/benchmarks/22_tons_of_vertex_collisions

One model approach

Calculate waiting positions deterministically
robots with longer paths ahead have higher priority.

```
higher(R,R'):- earliest_collision(R,R',C,T), last_time(R,D), last_time(R',D'), D-T<D'-T.
higher(robot(A),robot(B)):- earliest_collision(robot(A),robot(B),C,T),
      last_time(robot(A),D), last_time(robot(B),D'), D-T=D'-T, A<B.
```

Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_w_det_big_final.lp

One model approach

Wait A times at the time of a collision for A robots with higher priority. Only count robots that meet for the first time

```
do_wait(R,C,T-1,A) :- A == #count{1,R' : earliest_collision(R,R',C,T), higher(R,R')},
earliest_collision(R,R'',C,T).
```

Source: https://github.com/warpaint97/plan-merging-project/blob/main/encodings/sequential/merger_w_det_big_final.lp

Combining waiting and switching

1. apply plan switching

2. apply preventive plan switching

3. choose a waiting approach/ can be done automatically

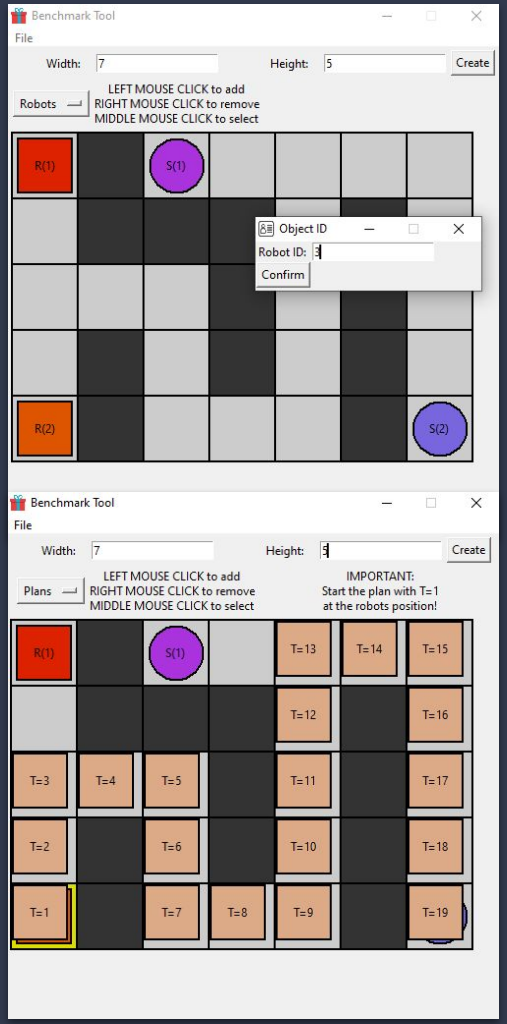
Evaluation

Benchmarks

1. for testing and verifying quality
2. to compare approaches

Benchmark Tool

1. custom benchmark creation tool for the groups
2. in Python with intuitive GUI
3. exports/imports correct format
4. saving time

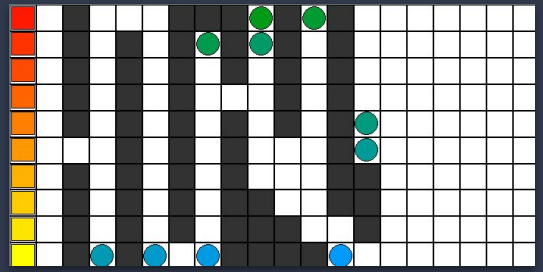


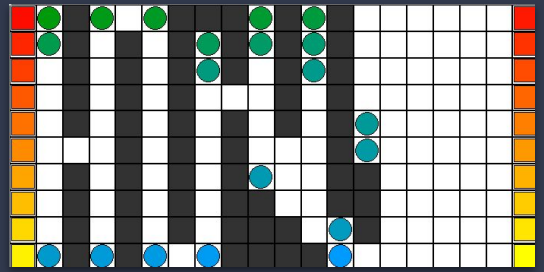
Source: https://github.com/warpaint97/plan-merging-project/tree/main/benchmark_tool/images/

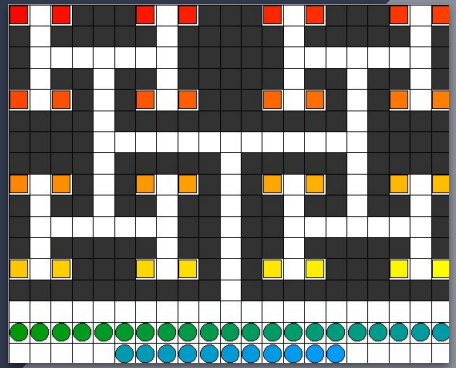
Metrics

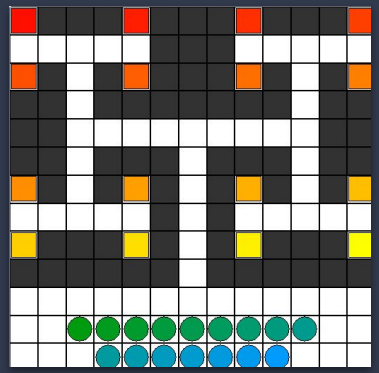
1. **Satisfiability**
   - no model
2. **Validity**
   - validity_checker.lp
3. **Performance**
   - total computation time
4. **Optimality**
   - not similarity
   - sum of all paths (SOAP)
   - or ΔSOAP = new SOAP - orig. SOAP

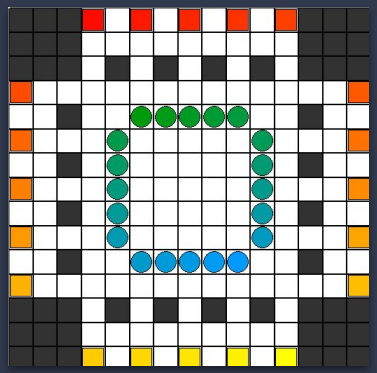
Benchmark Selection

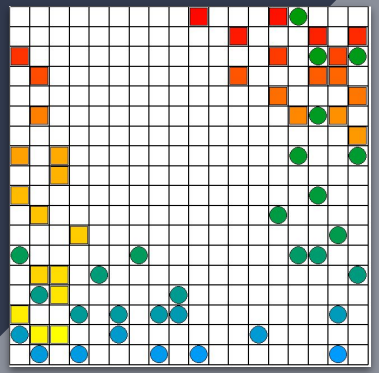


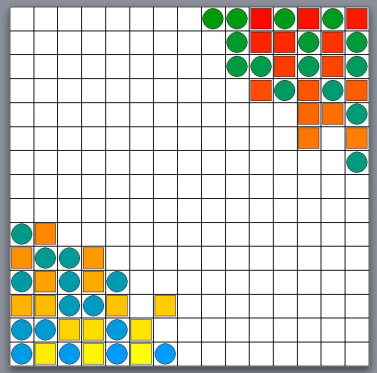












Source: https://github.com/warpaint97/plan-merging-project/tree/main/benchmarks/

Approach Selection

1. Naive Layer & Choice rule approaches too slow

2. Sequential Approaches yielded highest performance

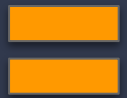
1) Preventive Plan Switching + Incremental Waiting
2) Preventive Plan Switching + Deterministic Waiting

Internal Comparison

1. deterministic vs. non-deterministic incremental waiting
2. look for drastic differences between them
3. what is causing that difference?

det	total	dSOAP
BM13	0.3587	20
BM14	0.7117	-353
BM20	0.0748	-116
BM22	3.3584	546
BM23	0.4872	120
BM25	1.4126	-177
BM26	1.7704	-429

inc	total	dSOAP
BM13	0.6936	20.0
BM14	0.7176	-349.0
BM20	0.0727	-116.0
BM22	NaN	NaN
BM23	2.8266	120.0
BM25	0.6497	-65.0
BM26	1.6687	-362.0



	total(det) - total(inc)	dSOAP(det) - dSOAP(inc)
BM13	-0.3349	0.0
BM14	-0.0059	-4.0
BM20	0.0021	0.0
BM22	NaN	NaN
BM23	-2.3394	0.0
BM25	0.7629	-112.0
BM26	0.1016	-67.0

Source: https://github.com/warpaint97/plan-merging-project/tree/main/benchmark_data/comparison_data/

Attributes of a Benchmark Instance

1. Benchmark instances have different attributes
2. look at preprocessed benchmark instances
3. find attributes that are correlated or causal for the difference

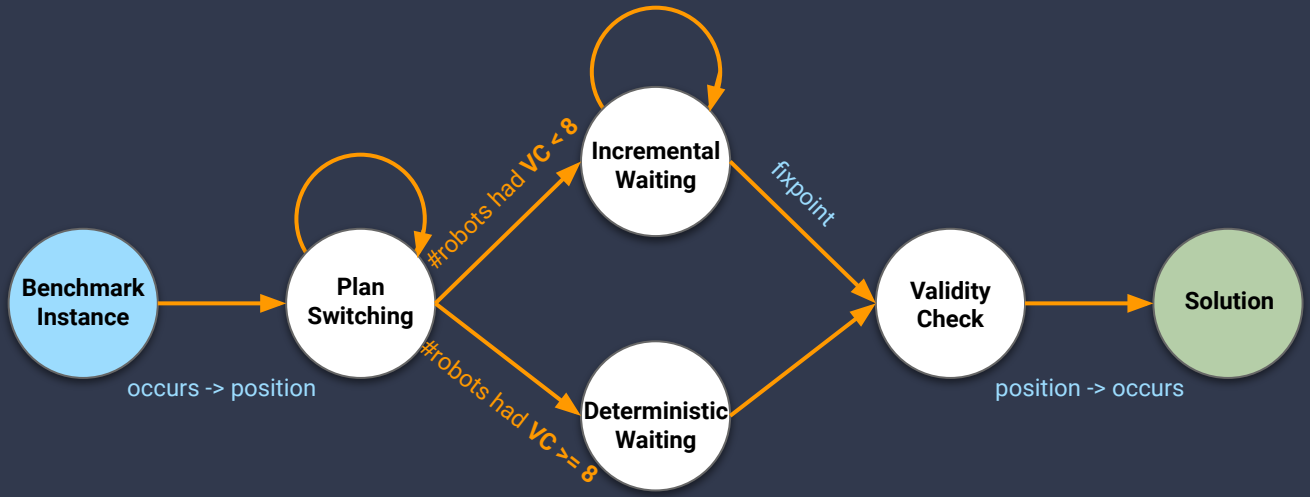
→ #robots involved in vertex collisions

	n robots	n pos	longest path	n vcs	n robots had vcs	n ecs	n robots had ecs	n fake ecs	n robots had fake ecs
BM1	1	5	5	0	0	0	0	0	0
BM2	2	12	6	0	0	0	0	0	0
BM3	2	10	5	1	2	0	0	0	0
BM4	4	8	2	1	4	0	0	0	0
BM5	2	10	5	2	2	0	0	0	0
BM6	4	12	3	2	4	0	0	0	0
BM7	3	15	5	3	3	0	0	0	0
BM8	6	48	9	9	6	0	0	0	0
BM9	3	19	7	3	2	0	0	0	0
BM10	3	11	4	0	0	0	0	0	0
BM11	3	17	6	4	2	0	0	0	0
BM12	2	13	7	0	0	0	0	0	0
BM13	10	341	47	82	8	0	0	0	0
BM14	20	326	32	32	6	0	0	0	0
BM15	2	4	2	0	0	0	0	0	0
BM16	3	11	6	0	0	0	0	0	0
BM17	7	32	9	0	0	0	0	0	0
BM18	4	20	5	6	4	0	0	0	0
BM19	8	56	7	12	8	0	0	0	0
BM20	20	124	7	8	8	0	0	0	0
BM21	12	56	5	4	8	0	0	0	0
BM22	32	983	36	122	32	0	0	0	0
BM23	16	327	22	56	16	0	0	0	0
BM24	8	116	16	26	8	0	0	0	0
BM25	30	538	26	18	9	0	0	0	0
BM26	0	0	0	0	0	0	0	0	0

Source: https://github.com/warpaint97/plan-merging-project/blob/main/benchmark_data/benchmark_metrics/bm_metrics_after_ps_small_table.PNG

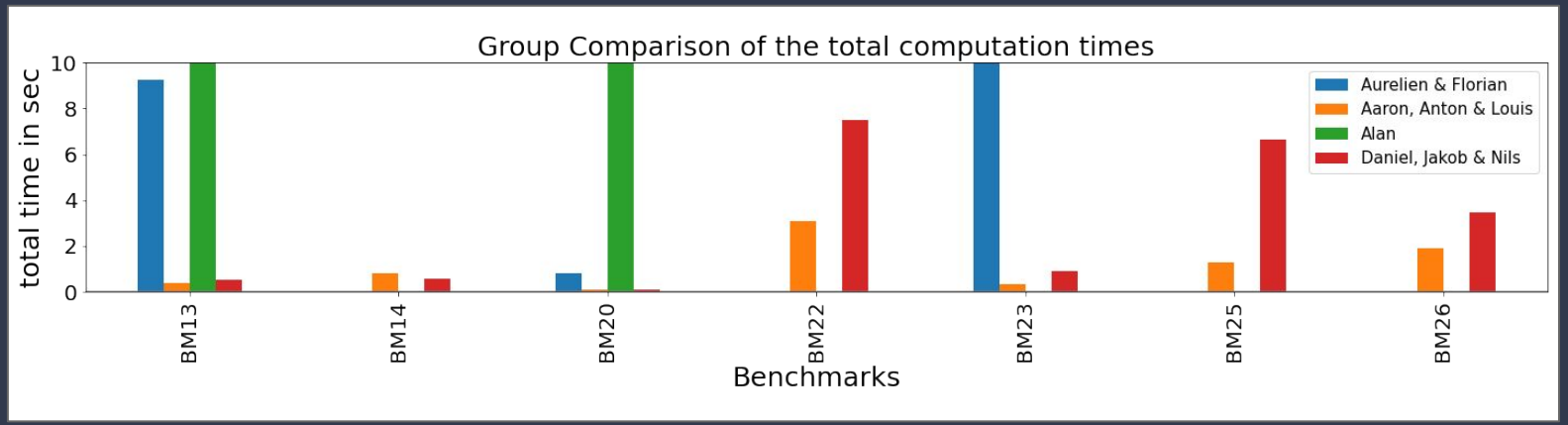
Final Approach

1. Merger class implements automatic merge method



Group Comparison (1)

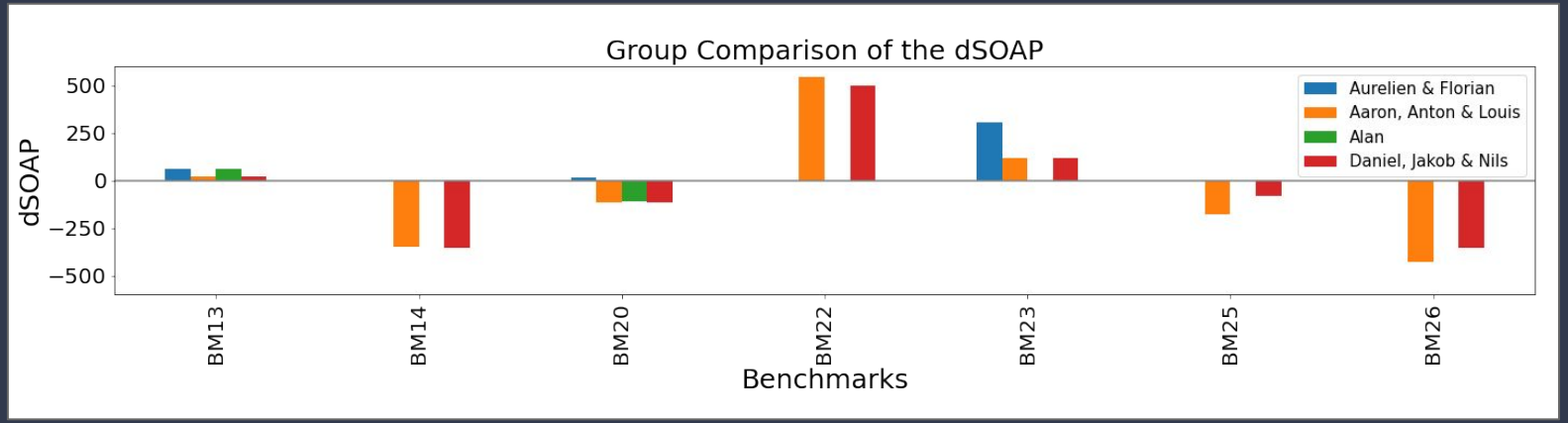
1. total computation time on large benchmarks



Source: https://github.com/warpaint97/plan-merging-project/blob/main/figures/group_comparison_total.png

Group Comparison (2)

1. difference in sum of all positions (dSOAP) on large benchmarks



Source: https://github.com/warpaint97/plan-merging-project/blob/main/figures/group_comparison_SOAP.png

Conclusion

- **Sequential approach**
  - plan switching (preprocessing)
  - waiting
- **Greedy Incremental Approach**
  - repeatedly optimizing heuristics
  - trade-off: speed vs. optimality
- **Combined Approach**
  - Why incremental waiting?
    - deterministic waiting might have risks
    - not tested on random BMs
- **Outlook**
  - improve deterministic waiting
  - test on random benchmarks

We’ll gladly answer all of your questions :)

Questions?

References

1. (2019). What robots do (and don't do) at Amazon fulfilment centres. https://www.aboutamazon.co.uk/news/operations/what-robots-do-and-dont-do-at-amazon-fulfilment-centres

2. M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, S. Thiele. (2010). A User's Guide to gringo, clasp, clingo, and iclingo. http://wp.doc.ic.ac.uk/arusso/wp-content/uploads/sites/47/2015/01/clingo_guide.pdf

3. P. Obermeier and T. Otto. (2018) Asprilo. https://asprilo.github.io

4. Potassco. (n.d.). clingo API documentation. https://potassco.org/clingo/python-api/5.4/

5. A. Bishop, A. Rabe, L. Donath. (2022). Plan Merging Project. https://github.com/warpaint97/plan-merging-project

6. A. Simon, F. Franz. (2022). KRR Project - Plan merging. https://github.com/Owrel/Project-KRR

7. A. Romer. (2022). KRR Project. https://github.com/AlanRomer/KRR-Projekt

8. D. Schreitter Ritter von Schwarzenfeld, J. Westphal and N. Bröcker. (2022). Asprilo-Project-DJN. https://gitup.uni-potsdam.de/nbroecker/asprilo-project-djn