MAPF Project

D. Andreev[1] P. Moek[2]

Bachelor of Computer Science
University of Potsdam

Multi-Agent Path Finding Project, 2022

Outline

MAPF

Multi-Agent Path Finding

- Automized robots moving in a warehouse
- Plan routes while avoiding collisions

Figure: Example of an automated Amazon-warehouse.[2]

[2] source: https://fba.help/wp-content/uploads/2019/05/Amazon-automated-warehouses-FB.jpg

Goals For The Project

- Solution should be complete
- Optimal solution slow → additional suboptimal algorithm
- Compare both + more choices

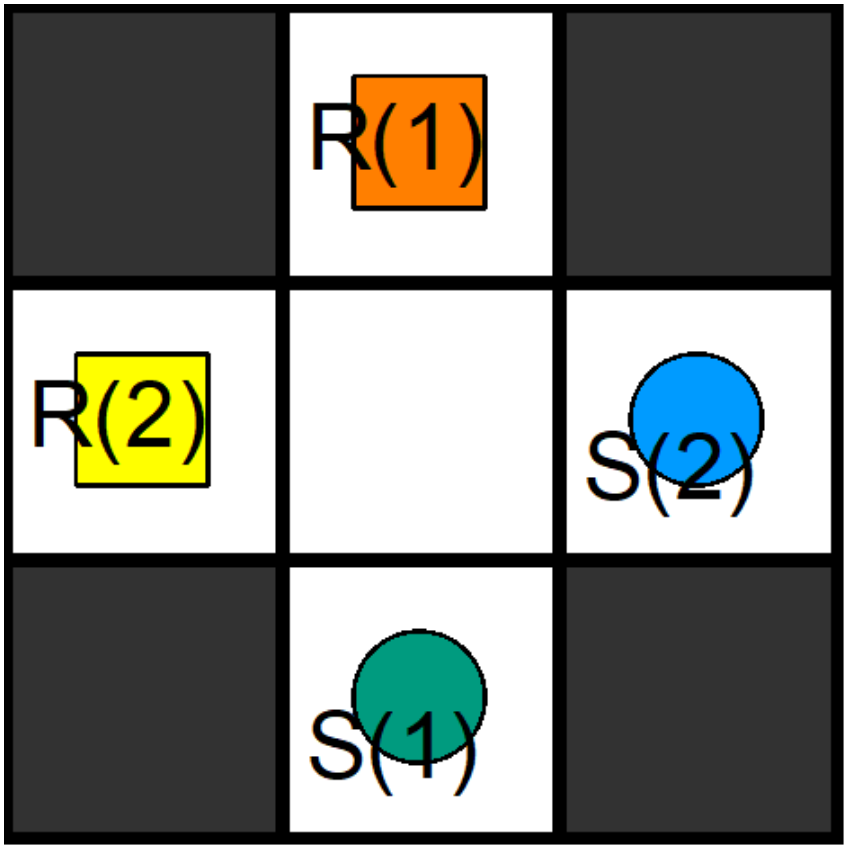


Figure: An easy instance shown in the asprilo visualizer.

Outline

CBS (Constraint-Based Search): High-Level Search

```
1: ROOT = node containing initial plan
2: Enqueue ROOT
3: while queue not empty do
4:     CURRENT = dequeue node with lowest cost
5:     if CURRENT has no conflicts then
6:         CURRENT is the solution
7:     end if
8:     FC = first conflict in CURRENT
9:     create two children for CURRENT:
10:        each child makes one of the robots avoid FC
11:    calculate plans, conflicts and costs for children (low-level search)
12:    enqueue children
13: end while
14: since queue is empty with no conflict found, initial problem is unsolvable
```

Outline

CBS (Constraint-Based Search): Low-Level Search

1: ROB = robot restricted by the new constraint
2: calculate new shortest path for ROB that satisfies given constraints
3: **if** there is no possible path **then**
4: no solution, disable child
5: **else**
6: substitute the old path of ROB with the new path in the plan
7: calculate all conflicts for the new plan
8: return the new plan and conflicts
9: **end if**

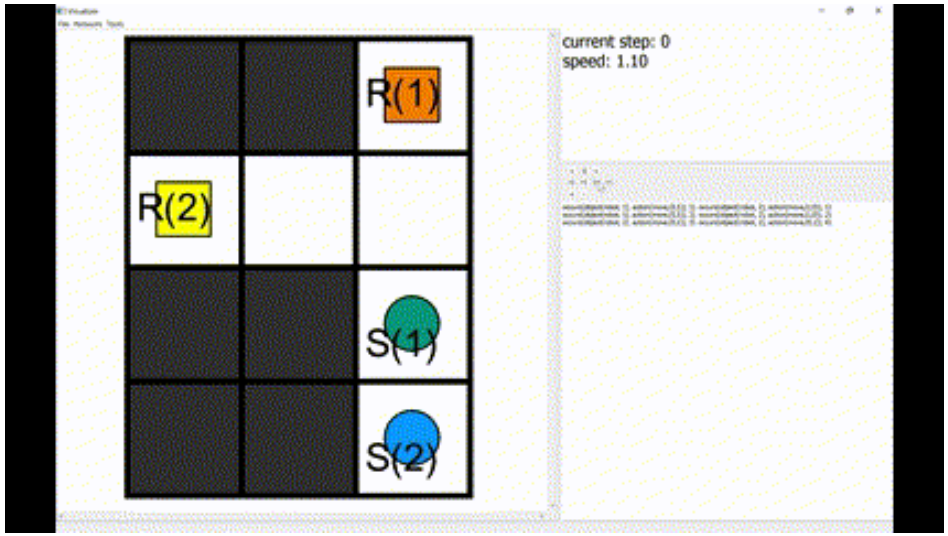
Outline

Cost Functions

1. Makespan
2. Sum of movement-times (SOMT)
3. Sum of lastmoves (SOL)
4. Number of conflicts (NOC)

Cost Functions and Greedy CBS

- Non-greedy CBS uses SOL (sum of lastmoves) cost function
- Greedy CBS $\rightarrow$ NOC (number of conflicts)



Cost Functions and Greedy CBS

- SOL not equal to makespan
- Makespan here: 4, Makespan above: 4
- SOL here: $1 + 4 = 5$, SOL above: $4 + 4 = 8$

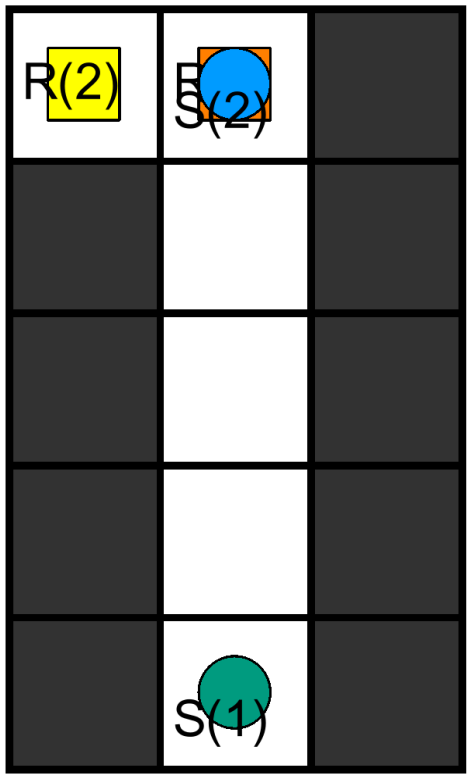


Figure: Makespan $\neq$ SOL

Outline

How did we gather our data?

- What did we measure?
  - *time:* runtime
  - *#nodes:* amount of nodes traversed in the search tree
  - *pathlength:* depth of solution in search tree
  - *horizon:* makespan of solution
  - *#moves:* sum of moves of all robots
  - *init_conflicts:* initial amount of conflicts in instance

How did we gather our data?

- Test instances
  - Generated automatically
  - Square shaped, all nodes enabled
  - Robots cannot be generated on their shelf
  - Timeout 5 min per instance
  - Benchmarking by running (greedy) CBS on folder (data set) of instances

How did we gather our data?

Design of data sets

1. Different sizes and robot-densities
   - Sizes: (5x5, 6x6, 7x7, 8x8)
   - Densities: (20%, 25%, 30%, 35%, 40%)
   - For each size/density combination ten examples
2. Incrementally increasing number of robots
   - Size set as 7x7
   - Robots ranging from 2 to 19 ($\sim$ 40% density)
   - Greedy CBS not challenged $\rightarrow$ up to 35 robots ($\sim$ 70% density)

How does the data look?

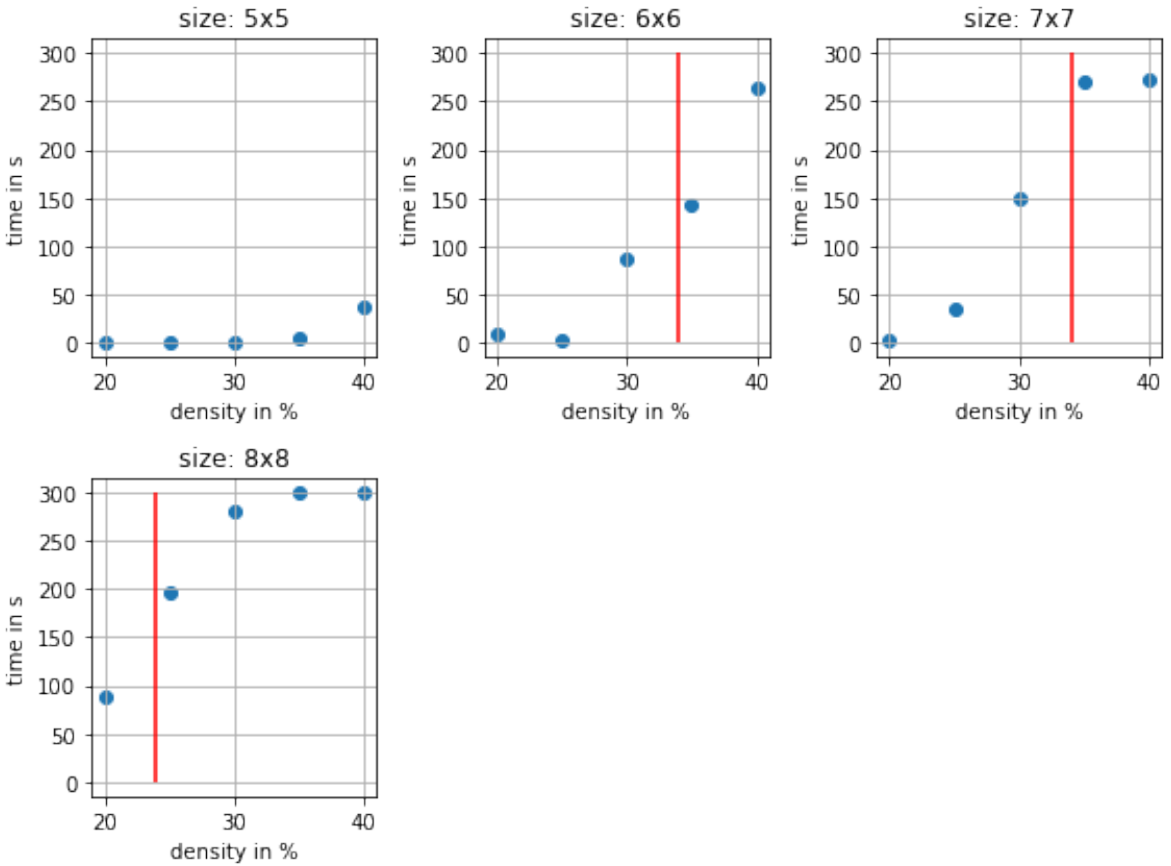


Figure: Solving speeds for densities of differently sized instances with non-greedy CBS

How does the data look?

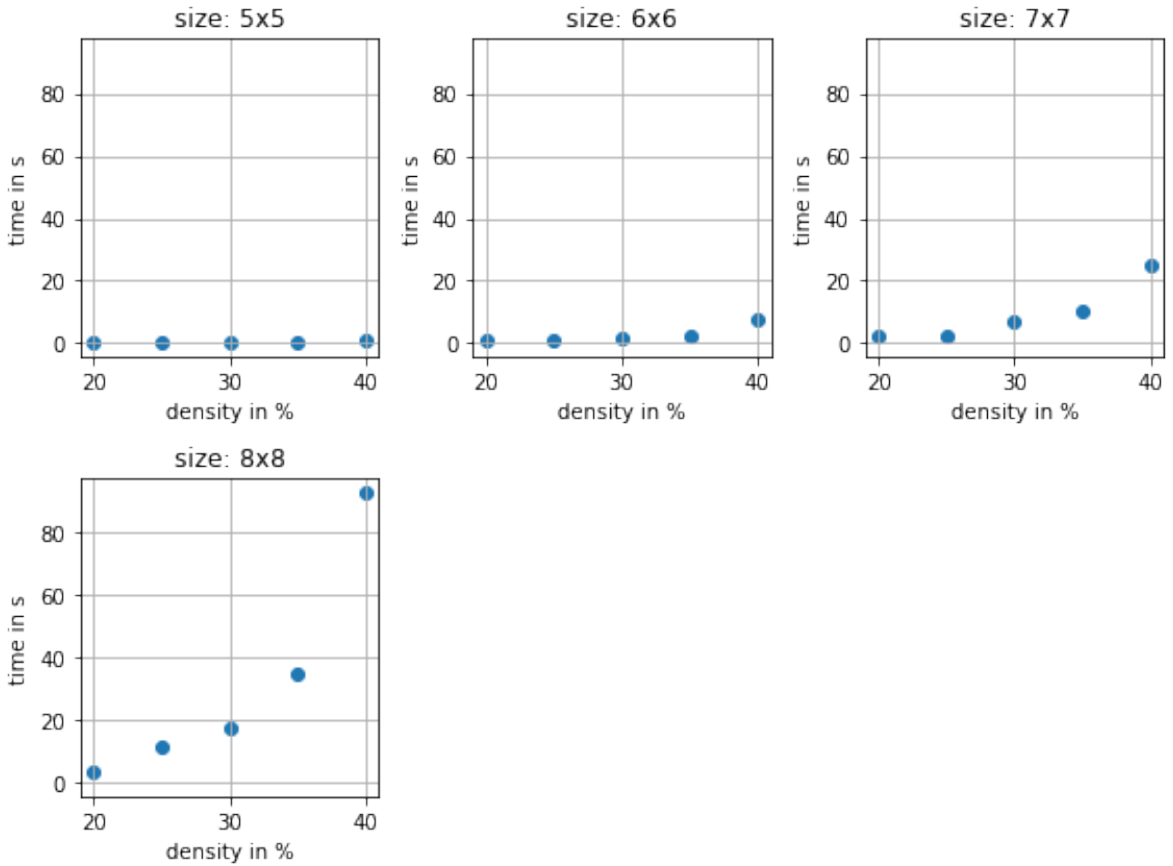


Figure: Solving speeds for densities of differently sized instances with greedy CBS

Speed greedy vs. non-greedy CBS

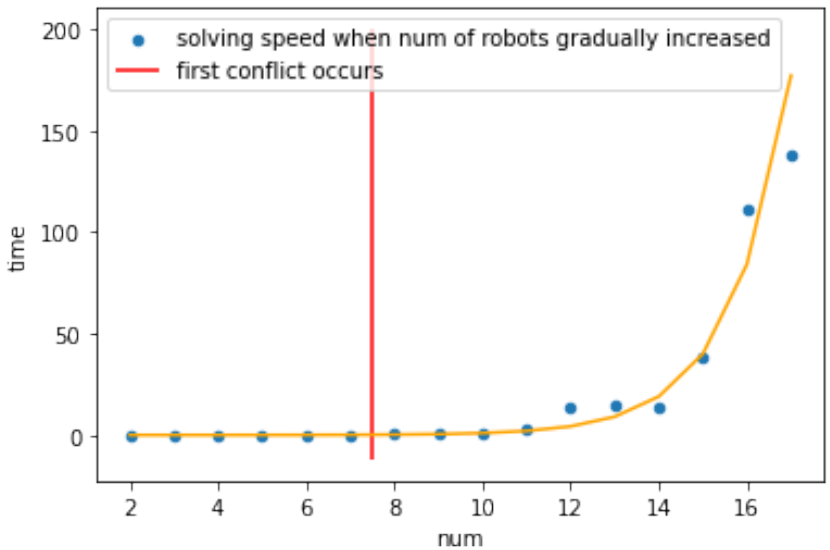


Figure: Runtime of CBS on incrementally increasing robot number

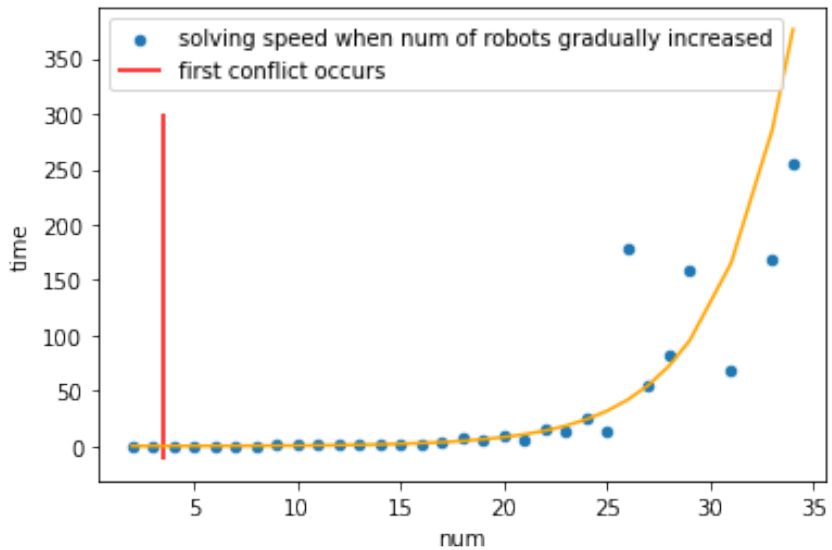


Figure: Runtime of greedy CBS on incrementally increasing robot number

Optimality greedy vs. non-greedy CBS

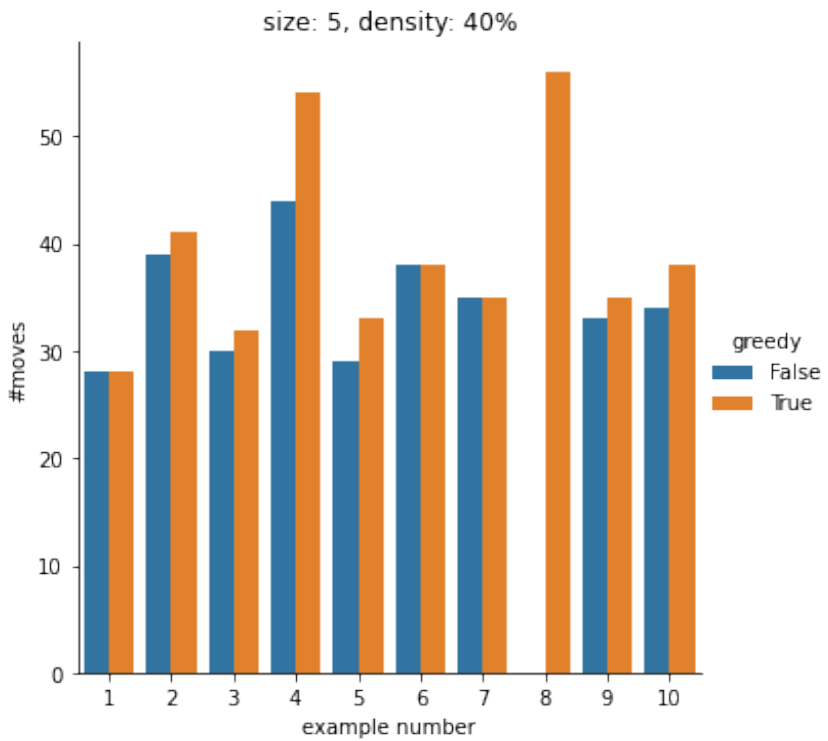


Figure: Move-sum on 5x5 instance with 40% robot-density

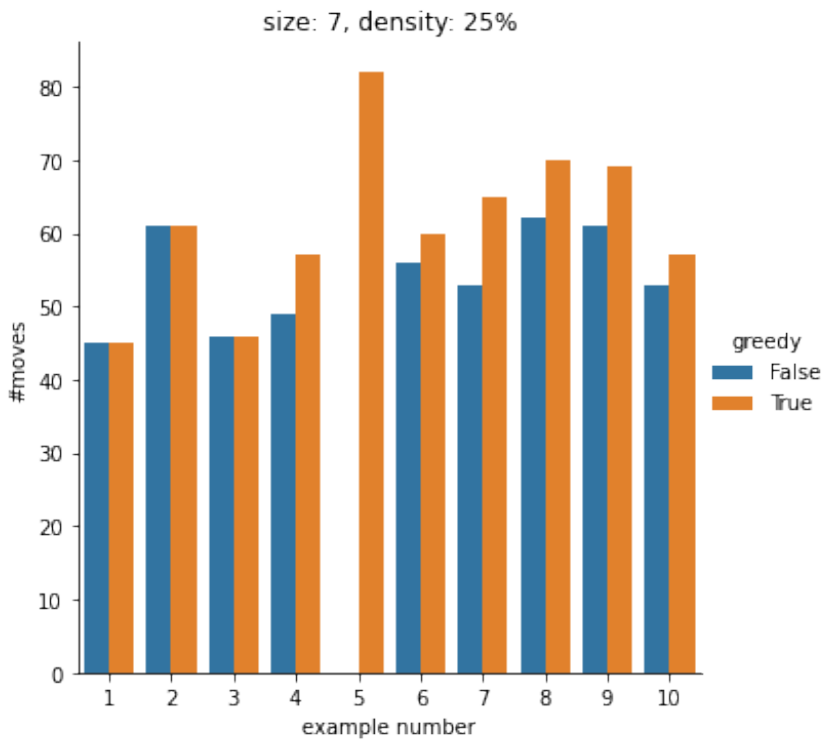


Figure: Move-sum on 7x7 instance with 25% robot-density

Outline

Comparing with other groups

Groups	Instances provided	Benchmark provided	Runs on their own instances	Runs on other instances
Alexander Benjamin Glätzer, Mert Akil	✓	✓	✓	no
Jan Behrens	✓	no	-	-
Philomena Moek, Denis Andreev	✓	✓	✓	✓
Andrés Córdova, Aleksandra Khatova	✓	no	-	-
Nico Sauerbrei, Paul Raatschen	✓	✓	✓	✓
Amade Nemes, Ryan Kepler Murphy	no	no	-	-
Steven Pan	✓	✓	✓	partially

Results (1)

	file	time	horizon	#moves	program	soc
0	AndreevMoek/14robs.lp	122.600684	9.0	67.0	AndreevMoek CBS	NaN
1	AndreevMoek/14robs.lp	6.133282	9.0	71.0	AndreevMoek greedyCBS	NaN
2	AndreevMoek/14robs.lp	3.247514	9.0	63.0	RaatschenSauerbrei CBS	65.0
3	AndreevMoek/14robs.lp	1.043049	15.0	77.0	RaatschenSauerbrei IS	210.0
4	AndreevMoek/14robs.lp	0.445551	10.0	71.0	RaatschenSauerbrei PP	73.0
5	AndreevMoek/ex8.lp	117.700398	10.0	94.0	AndreevMoek CBS	NaN
6	AndreevMoek/ex8.lp	8.839043	10.0	102.0	AndreevMoek greedyCBS	NaN
7	AndreevMoek/ex8.lp	6.359438	10.0	94.0	RaatschenSauerbrei CBS	94.0
8	AndreevMoek/ex8.lp	0.584755	10.0	96.0	RaatschenSauerbrei PP	98.0
9	Behrens/mix.lp	19.002301	11.0	182.0	AndreevMoek greedyCBS	NaN
10	Behrens/small_room.lp	8.696061	12.0	33.0	AndreevMoek CBS	NaN
11	Behrens/small_room.lp	3.617692	16.0	41.0	AndreevMoek greedyCBS	NaN
12	Behrens/small_room.lp	1.421336	11.0	29.0	RaatschenSauerbrei CBS	30.0
13	Behrens/small_room.lp	0.209429	12.0	29.0	RaatschenSauerbrei IS	36.0
14	Behrens/small_room.lp	0.175655	11.0	29.0	RaatschenSauerbrei PP	30.0
15	GlätzerAkil/mini_maze.lp	0.432578	14.0	29.0	AndreevMoek CBS	NaN
16	GlätzerAkil/mini_maze.lp	0.615183	22.0	52.0	AndreevMoek greedyCBS	NaN
17	GlätzerAkil/mini_maze.lp	0.582606	NaN	NaN	Pan	60.0
18	GlätzerAkil/mini_maze.lp	0.391481	NaN	NaN	Pan	48.0

Results (2)

	file	time	horizon	#moves	program	soc
19	GlätzerAkil/mini_maze.lp	0.750859	15.0	56.0	RaatschenSauerbrei CBS	58.0
20	GlätzerAkil/mini_maze.lp	0.459576	15.0	56.0	RaatschenSauerbrei PP	58.0
21	GlätzerAkil/warehouse.lp	0.049554	17.0	60.0	AndreevMoek CBS	NaN
22	GlätzerAkil/warehouse.lp	0.049498	17.0	60.0	AndreevMoek greedyCBS	NaN
23	GlätzerAkil/warehouse.lp	0.756851	17.0	60.0	RaatschenSauerbrei CBS	60.0
24	KhatovaCordova/24_merged_plans.lp	105.608239	15.0	247.0	AndreevMoek greedyCBS	NaN
25	KhatovaCordova/24_merged_plans.lp	266.851197	16.0	235.0	RaatschenSauerbrei CBS	236.0
26	KhatovaCordova/24_merged_plans.lp	2.663130	15.0	239.0	RaatschenSauerbrei PP	263.0
27	KhatovaCordova/30_merged_plans.lp	43.134053	19.0	309.0	AndreevMoek greedyCBS	NaN
28	KhatovaCordova/30_merged_plans.lp	14.133932	19.0	303.0	RaatschenSauerbrei CBS	303.0
29	KhatovaCordova/30_merged_plans.lp	3.614463	19.0	307.0	RaatschenSauerbrei PP	316.0
30	Pan/x6_y11_r16_fo1_cmw1_cx4_cy4_rom2_instance_...	14.716154	NaN	NaN	Pan	332.0
31	Pan/x6_y11_r32_fo2_cmw1_cx4_cy4_rom2_instance_...	87.124829	NaN	NaN	Pan	177.0
32	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-1.lp	3.872790	NaN	NaN	Pan	119.0
33	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-1.lp	4.185371	31.0	133.0	RaatschenSauerbrei IS	279.0
34	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-1.lp	2.329662	23.0	129.0	RaatschenSauerbrei PP	153.0
35	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-2.lp	7.305222	12.0	48.0	AndreevMoek greedyCBS	NaN
36	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-2.lp	0.426617	NaN	NaN	Pan	47.0
37	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-2.lp	19.908878	12.0	44.0	RaatschenSauerbrei CBS	45.0
38	RaatschenSauerbrei/Plan-Sauerbrei-Raatschen-2.lp	0.242980	8.0	46.0	RaatschenSauerbrei PP	49.0

Figure: Results of comparing with other groups (Part 2).

Conclusion

- Both CBS implementations slower on bigger/denser instances
- Difference greedy CBS/non-greedy CBS $\nearrow$ with size/density
- Non-greedy CBS slower than other CBS implementations
- Greedy CBS can compete

- Outlook
  - Compare to different cost function(s)
  - Compare with other groups more in depth
  - Explore different approaches (instead of m-domain)