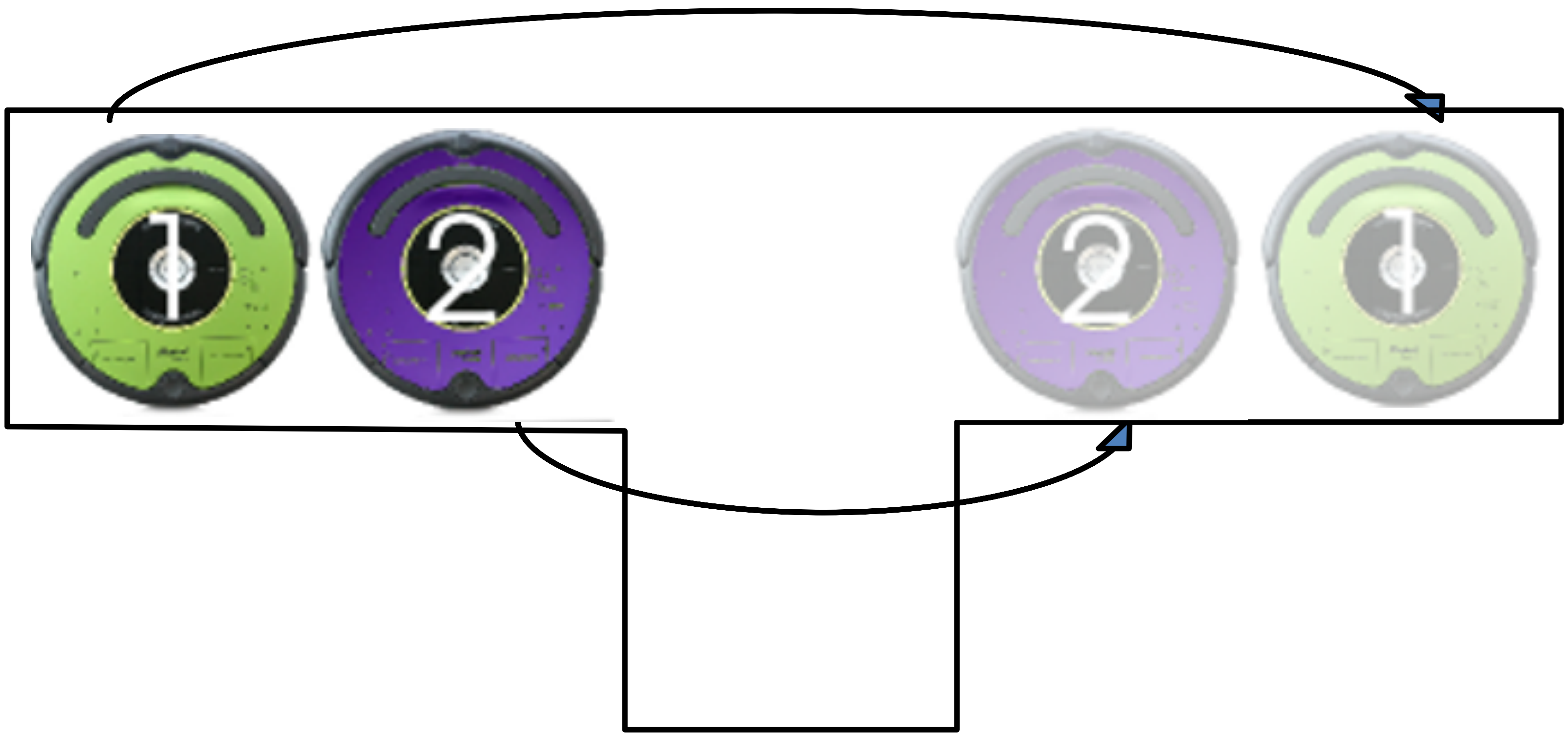
Multi-Agent Path Finding (MAPF)

- Multi-robot path finding
  - Given: a number of robots (each with a start and goal location) and a known environment
  - Task: find collision-free paths for the robots from their start to their goal locations that minimize some objective

Multi-Agent Path Finding (MAPF)

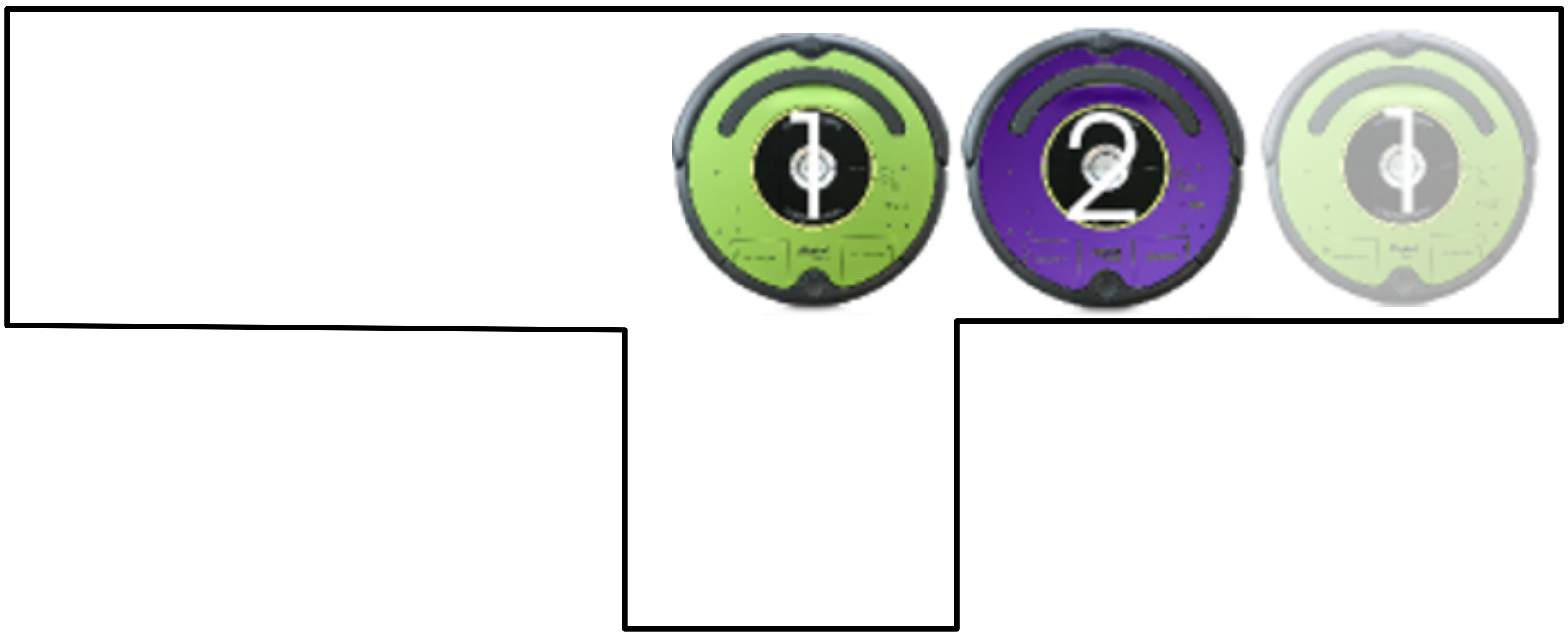


4-neighbor grid

Multi-Agent Path Finding (MAPF)

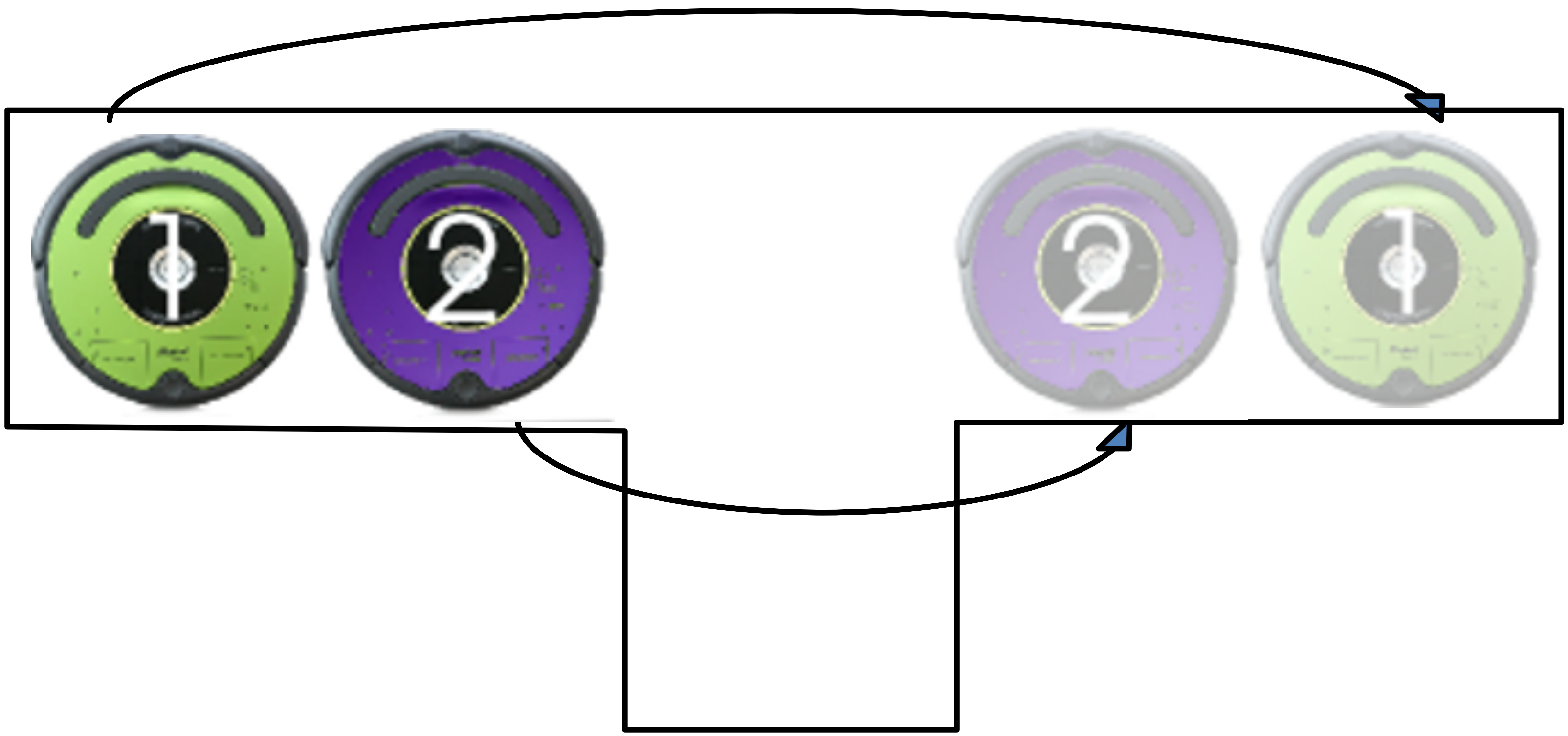
4-neighbor grid

Multi-Agent Path Finding (MAPF)



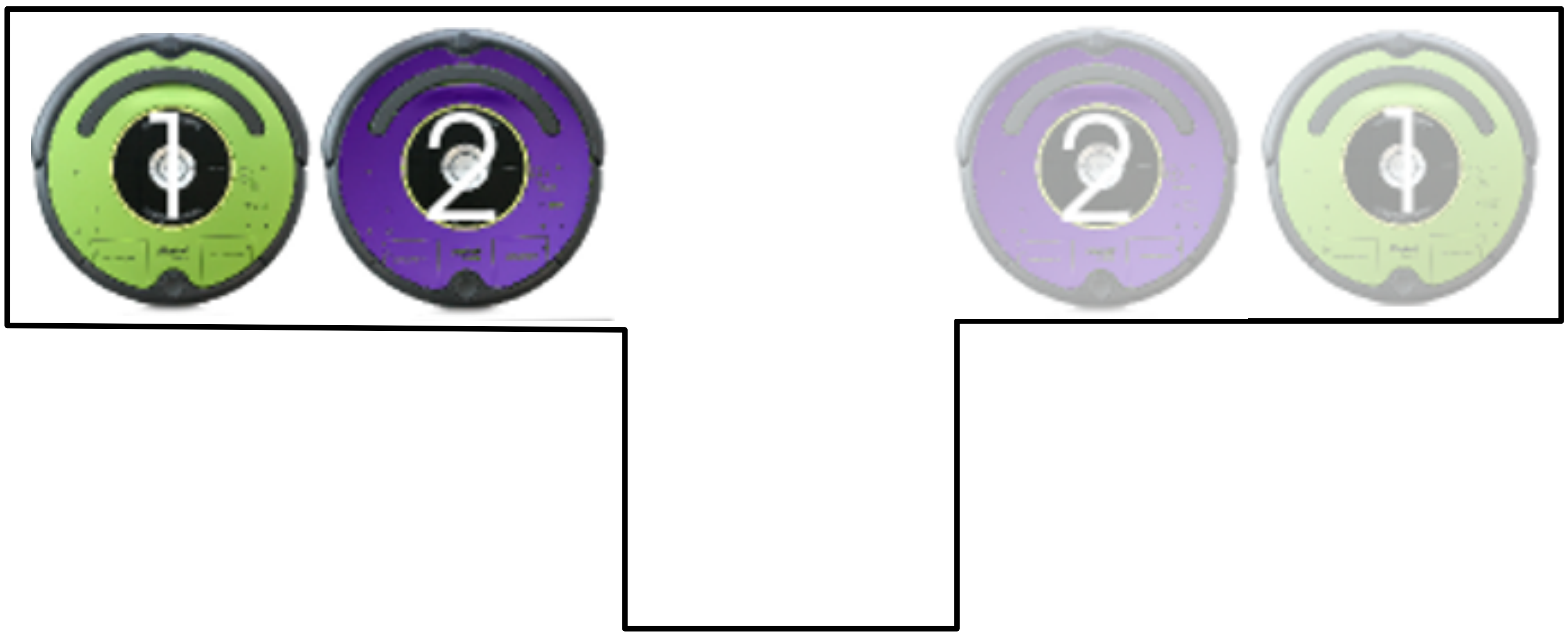
4-neighbor grid

Multi-Agent Path Finding (MAPF)



4-neighbor grid

Multi-Agent Path Finding (MAPF)



4-neighbor grid

Multi-Agent Path Finding (MAPF)

4-neighbor grid

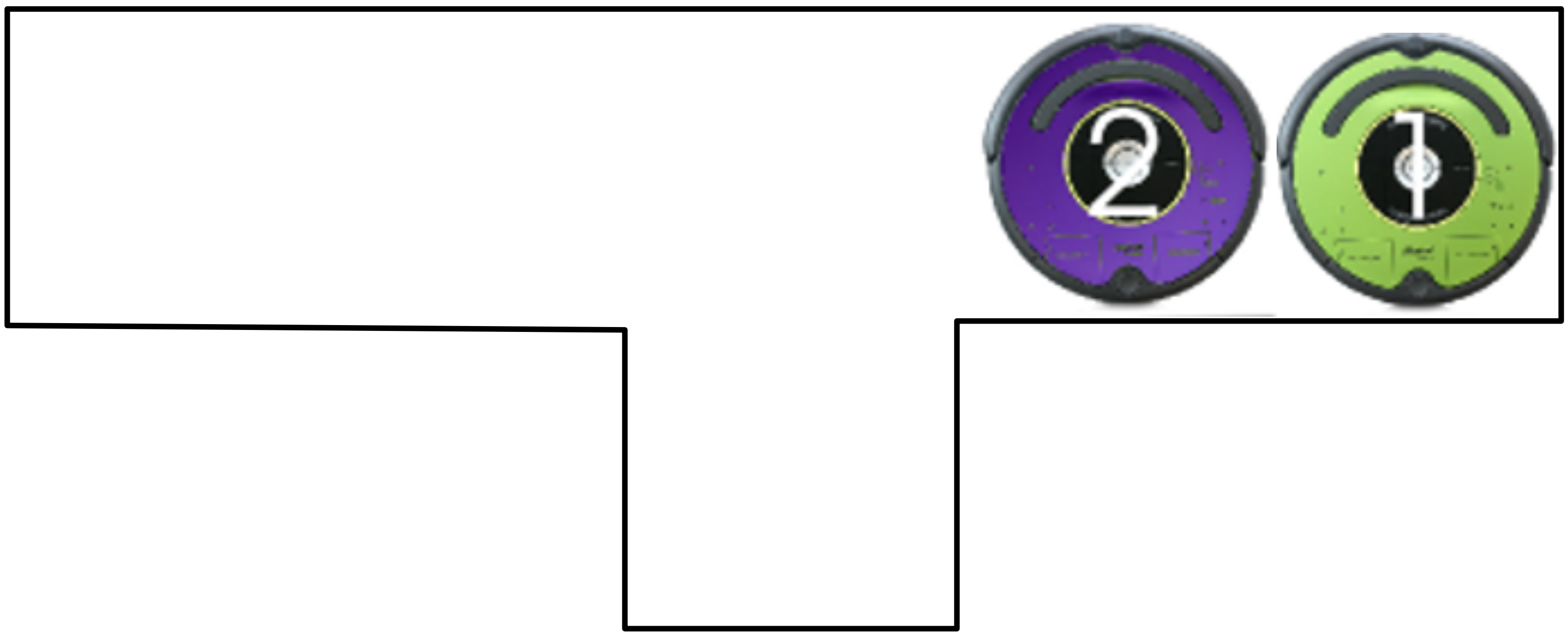
Multi-Agent Path Finding (MAPF)

4-neighbor grid

Multi-Agent Path Finding (MAPF)

4-neighbor grid

Multi-Agent Path Finding (MAPF)

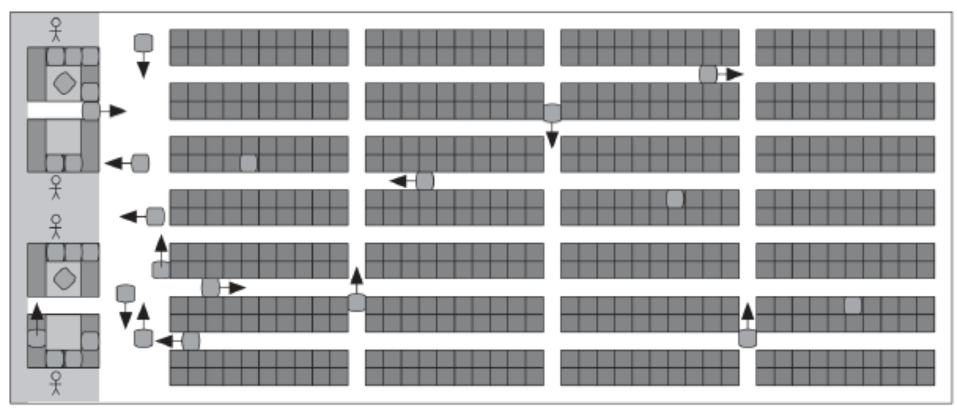


- Optimization problem with the objective to minimize task-completion time (called makespan) or the sum of travel times (called flowtime)

4-neighbor grid

Multi-Agent Path Finding (MAPF)

- Application: Amazon fulfillment centers



4-neighbor grid

[work by Kiva Systems/Amazon Robotics, not me]

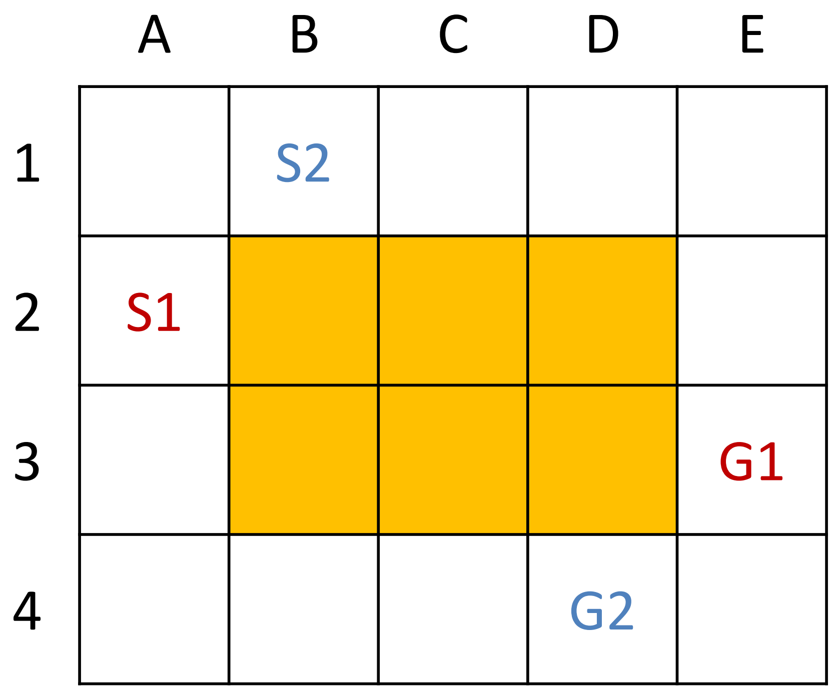
Multi-Agent Path Finding (MAPF)

- Optimal MAPF algorithms
  - Theorem [Yu and LaValle]: MAPF is NP-hard to solve optimally for makespan or flowtime minimization

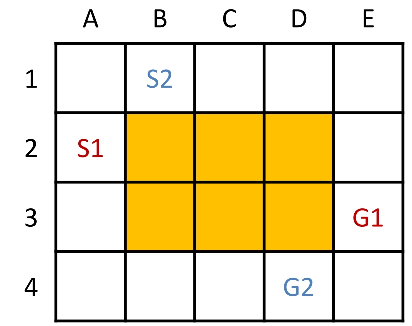
[www.random-ideas.net]

- Bounded-suboptimal MAPF algorithms
  - Theorem: MAPF is NP-hard to approximate within any factor less than 4/3 for makespan minimization on graphs in general

Multi-Agent Path Finding (MAPF)

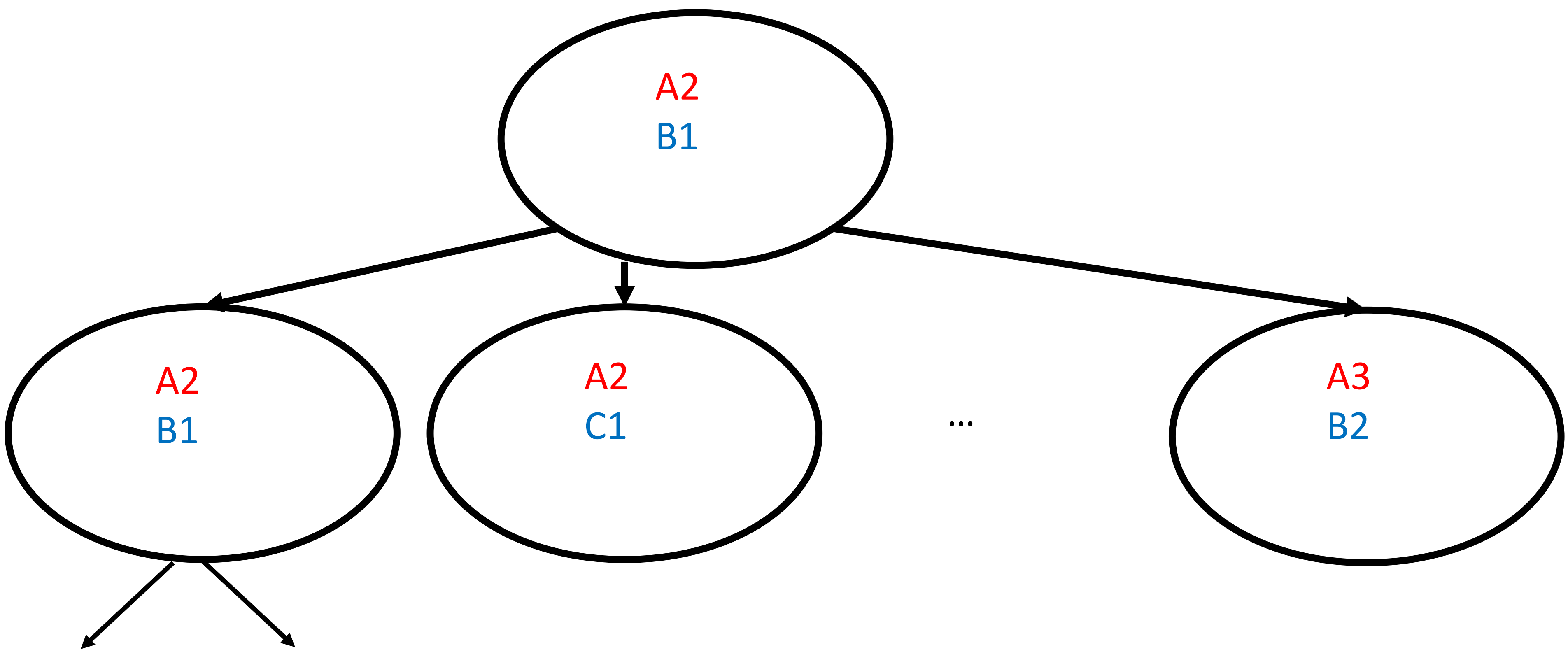


4-neighbor grid

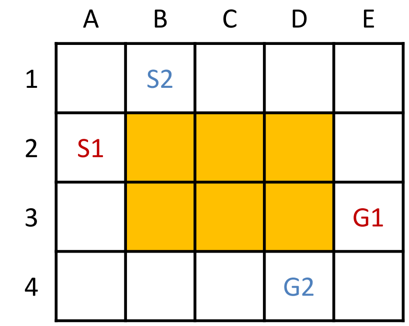


A*-Based Search

- A*-based search: Optimal (or bounded-suboptimal) MAPF solver

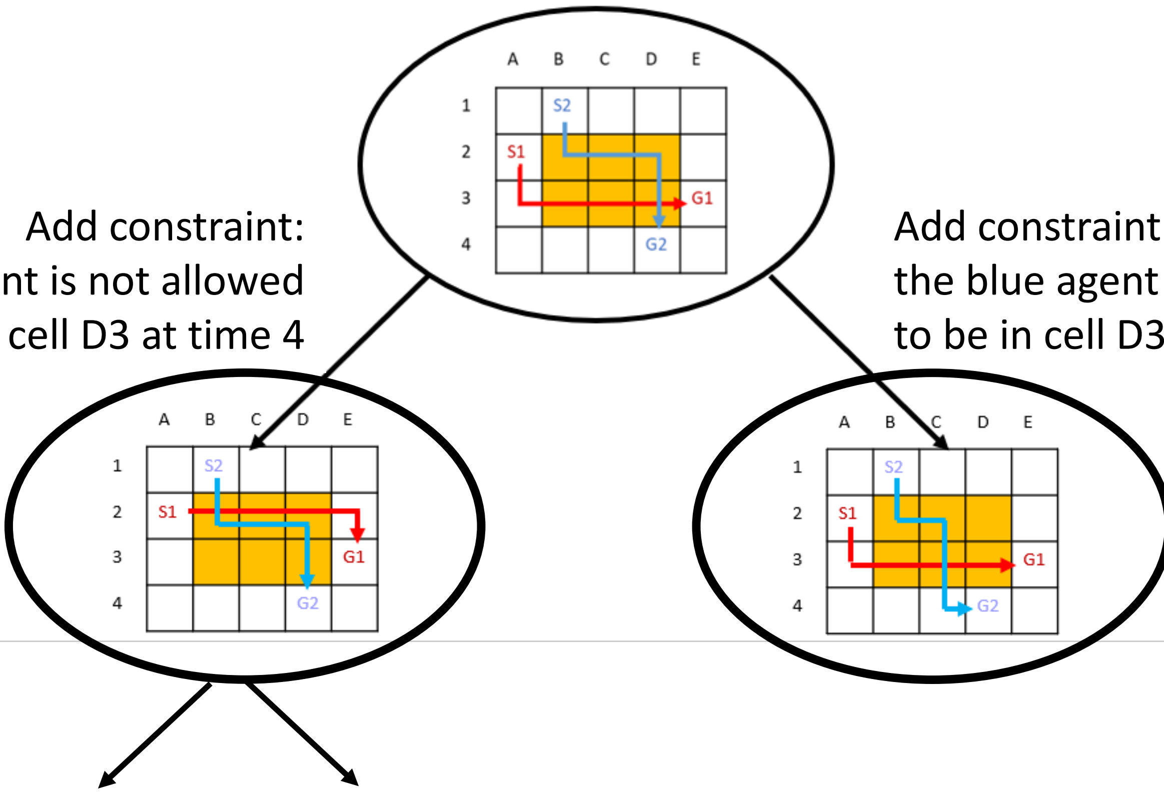


Conflict-Based Search



- Conflict-based search [Sharon, Stern, Felner and Sturtevant]: Optimal (or bounded-suboptimal) MAPF solver that plans for each agent independently

Add constraint:
the red agent is not allowed
to be in cell D3 at time 4

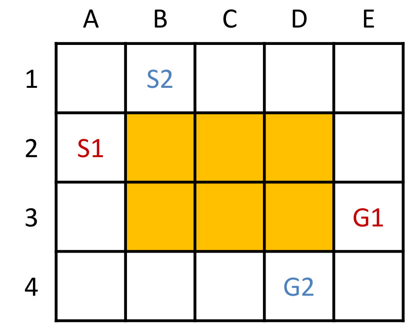


Add constraint:
the blue agent is not allowed
to be in cell D3 at time 4

4-neighbor grid

[work by Ben-Gurion University of the Negev, not me]

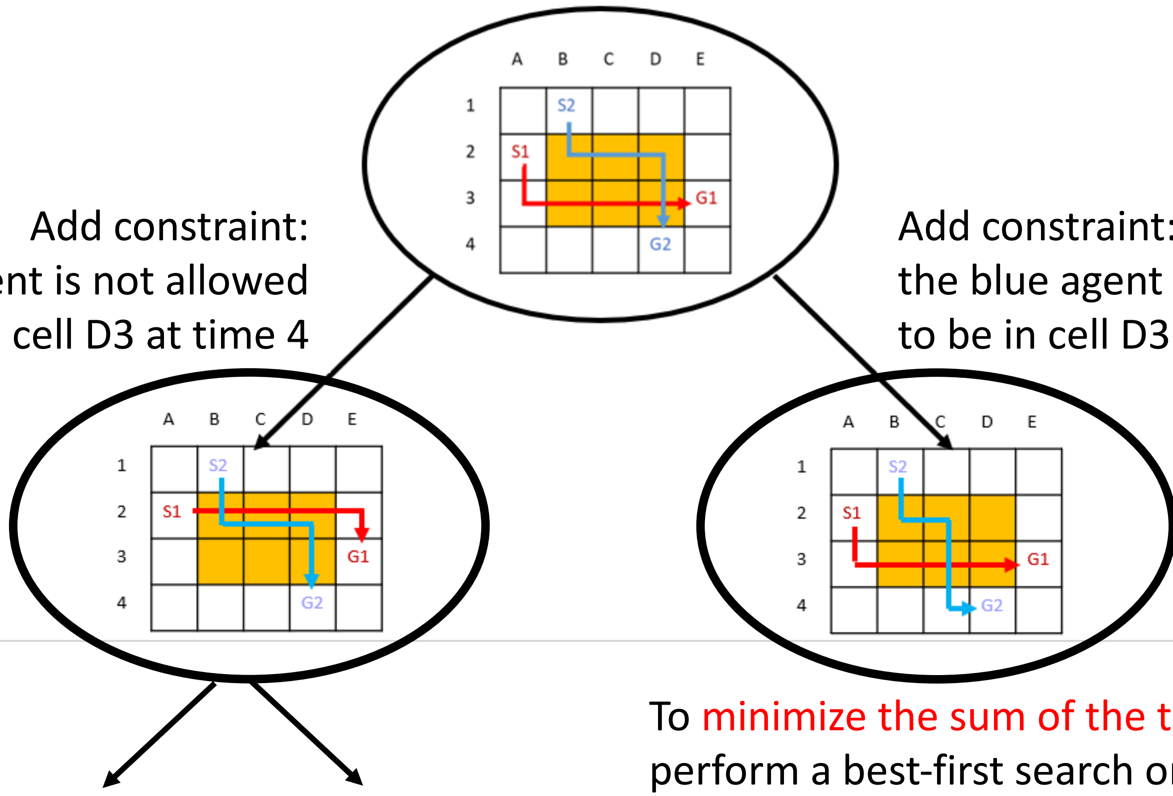
Conflict-Based Search



- Conflict-based search [Sharon, Stern, Felner and Sturtevant]: Optimal (or bounded-suboptimal) MAPF solver that plans for each agent independently

Add constraint:
the red agent is not allowed
to be in cell D3 at time 4

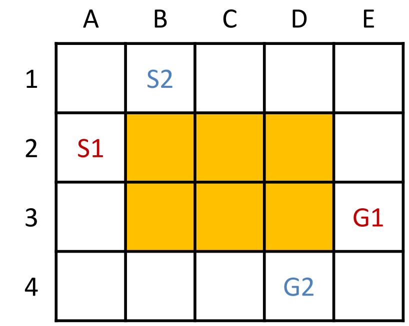
Add constraint:
the blue agent is not allowed
to be in cell D3 at time 4



4-neighbor grid

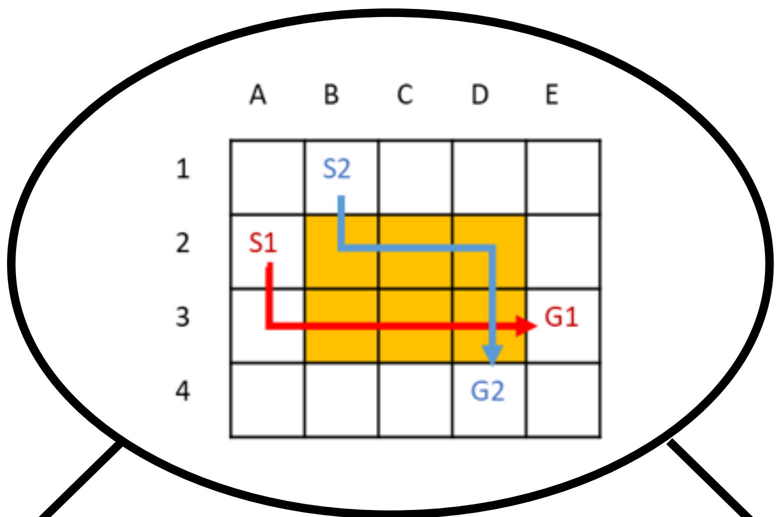
To minimize the sum of the travel times of all agents perform a best-first search on this tree with

- g = sum of travel times of all agents
- h = 0



Improvement 1

- Use more informed (= non-zero) h-values



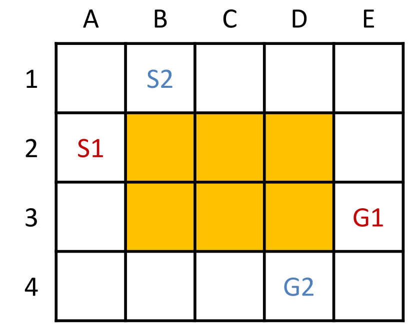
Add constraint:
the red agent is not allowed
to be in cell D3 at time 4

Add constraint:
the blue agent is not allowed
to be in cell D3 at time 4

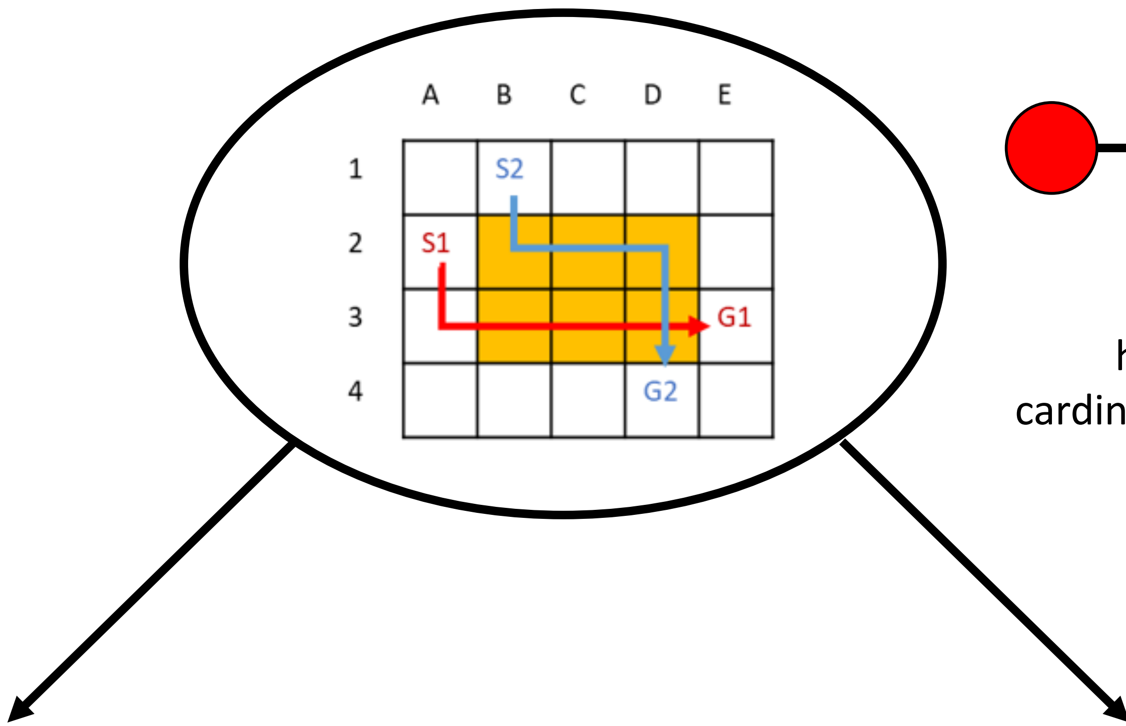
The sum of travel times of any collision-free solution is at least 11.

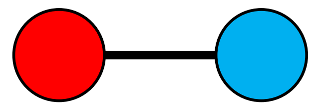
4-neighbor grid

Improvement 1



- Use more informed (= non-zero) h-values





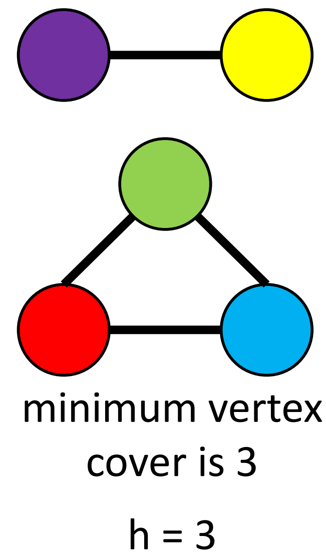
h = 1
cardinal conflict

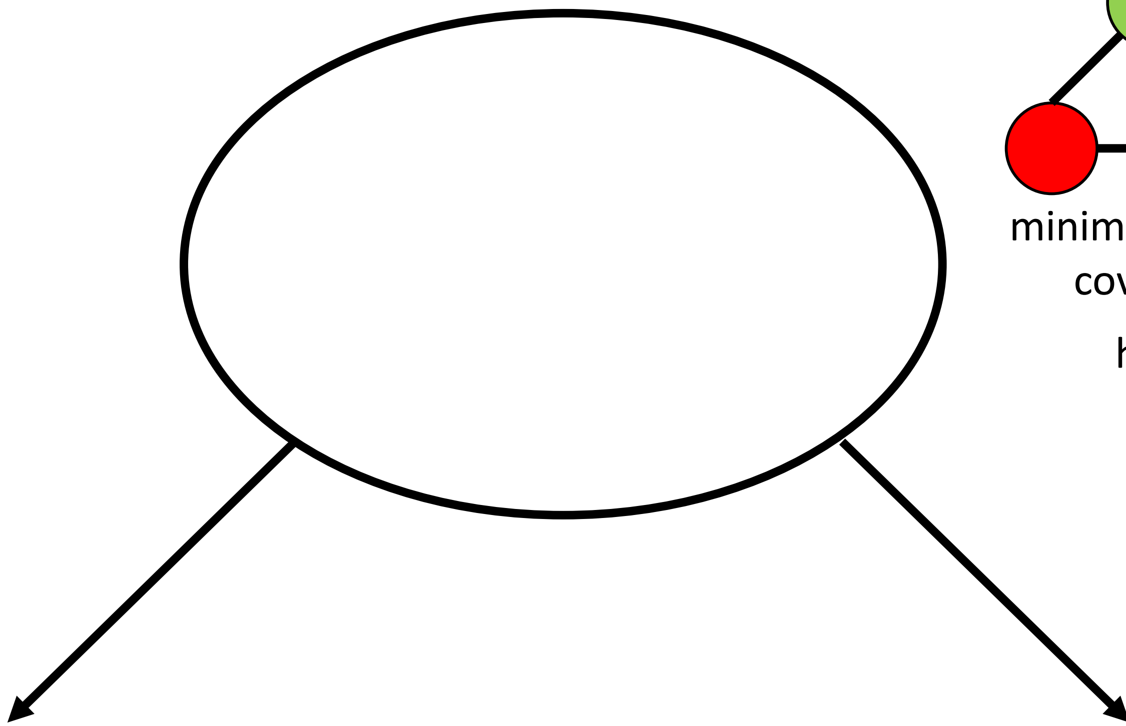
The sum of travel times of any collision-free solution is at least 11.

4-neighbor grid

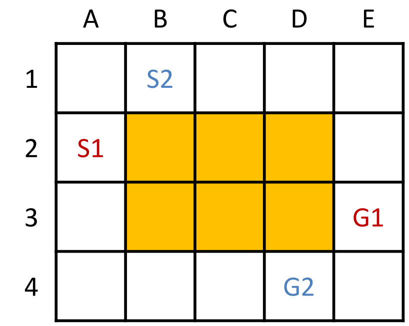
Improvement 1

- Use more informed (= non-zero) h-values



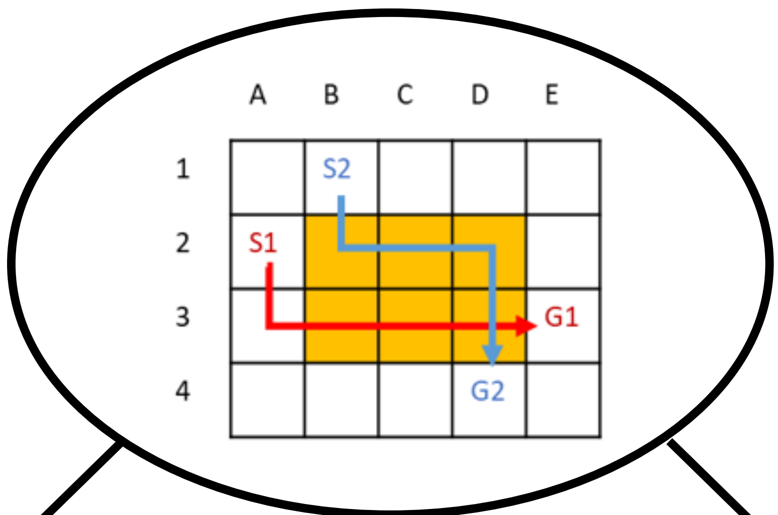


4-neighbor grid



Improvement 2

- Symmetry breaking of rectangle conflicts



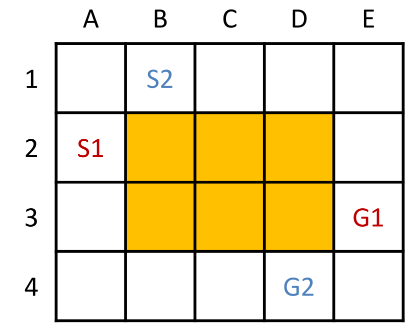
Add constraint:
the red agent is not allowed
to be in cell D3 at time 4

Add constraint:
the blue agent is not allowed
to be in cell D3 at time 4

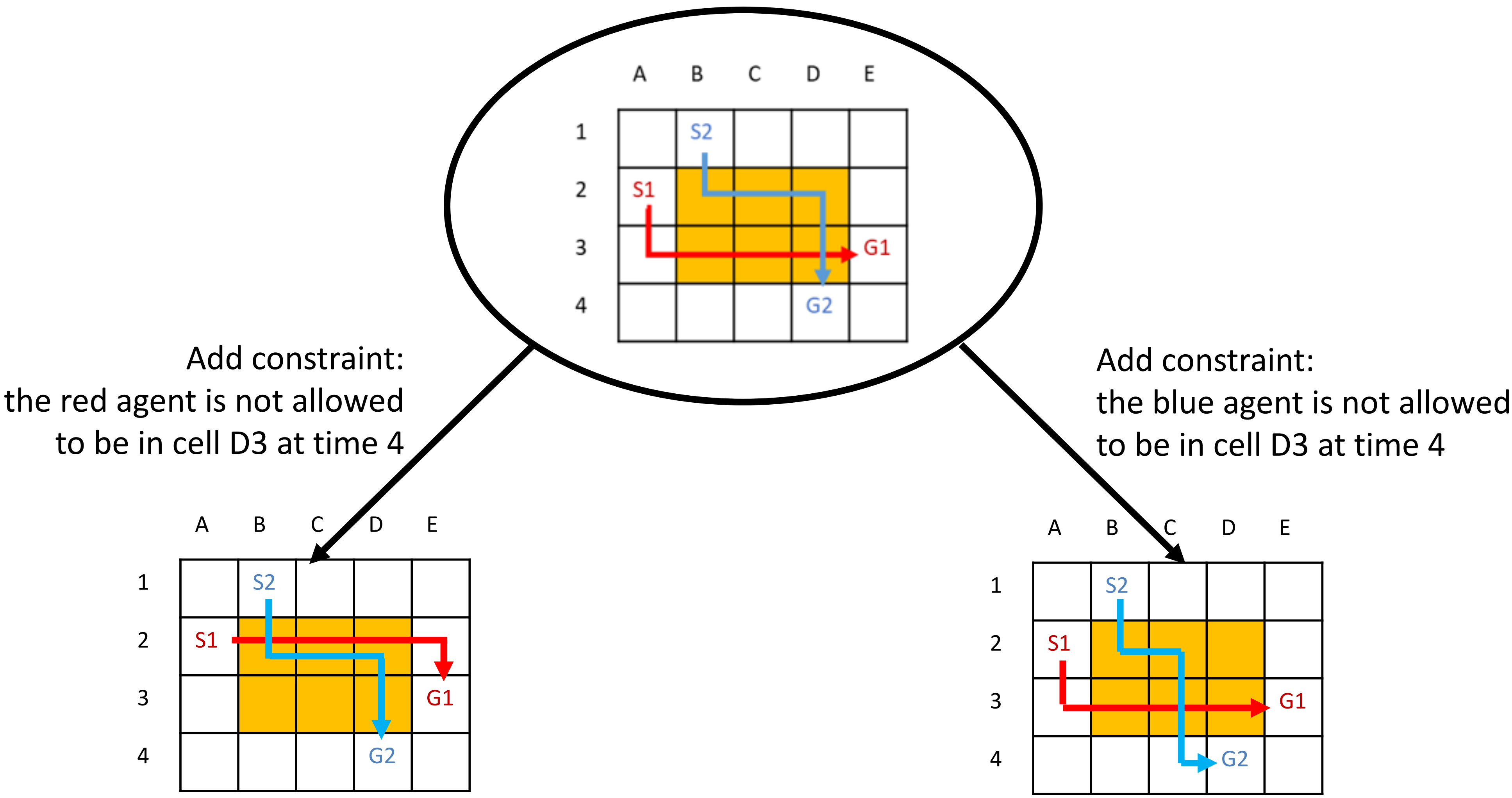
The sum of travel times of any collision-free solution is at least 11 but conflict-based search does not detect it right away.

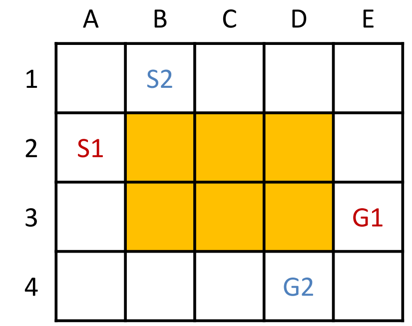
4-neighbor grid

Improvement 2



- Symmetry breaking of rectangle conflicts





Improvement 2

- Symmetry breaking of rectangle conflicts

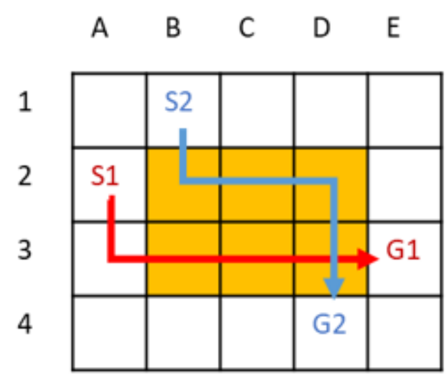
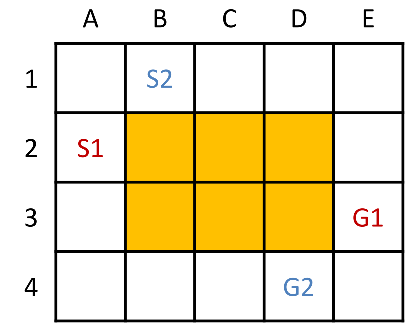


Table 1: Number of expanded CT nodes by CBSH on instances that 2 agents involve in cardinal rectangle conflicts. The first column and first row are the width and length of the rectangular area.

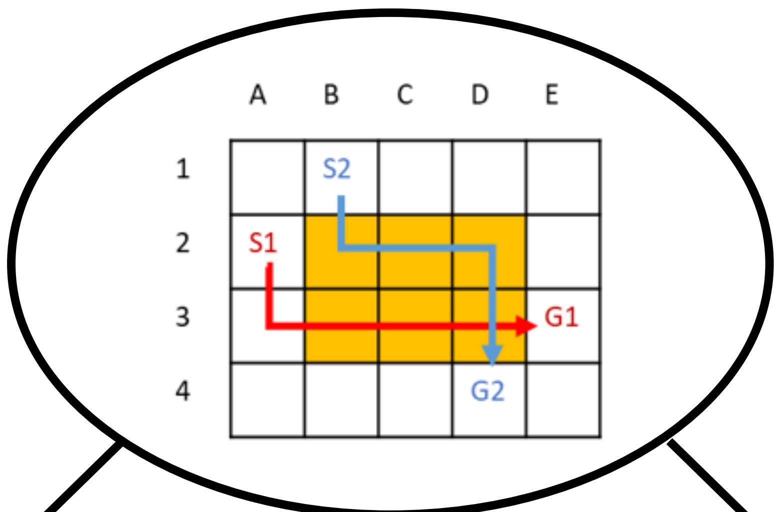
	1	2	3	4	5	6	7	8	9
1	1	1	2	3	4	5	6	7	8
2		3	7	14	26	46	79	133	221
3			22	53	116	239	472	904	1,692
4				142	392	1,016	2,651	6,828	17,747
5					1,015	2,971	8,525	23,733	65,236
6						7,447	24,275	78,002	254,173
7							62,429	222,524	795,197
8								573,004	>1,518,151

4-neighbor grid

Improvement 2

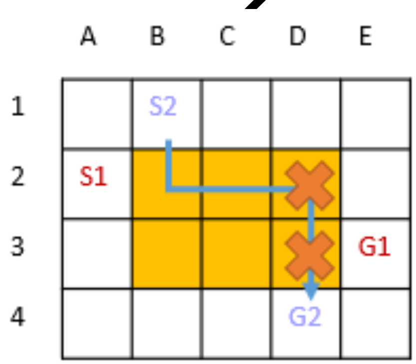


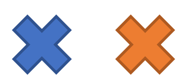
- Symmetry breaking of rectangle conflicts



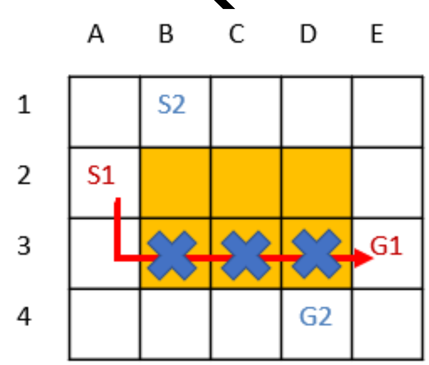
Add constraint:
the red agent is not allowed
to be in cell D3 at time 4
or in cell D2 at time 3

Add constraint:
the blue agent is not allowed
to be in cell D3 at time 4,
in cell C3 at time 3
or in cell B2 at time 2

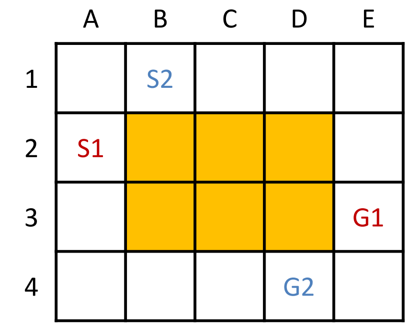




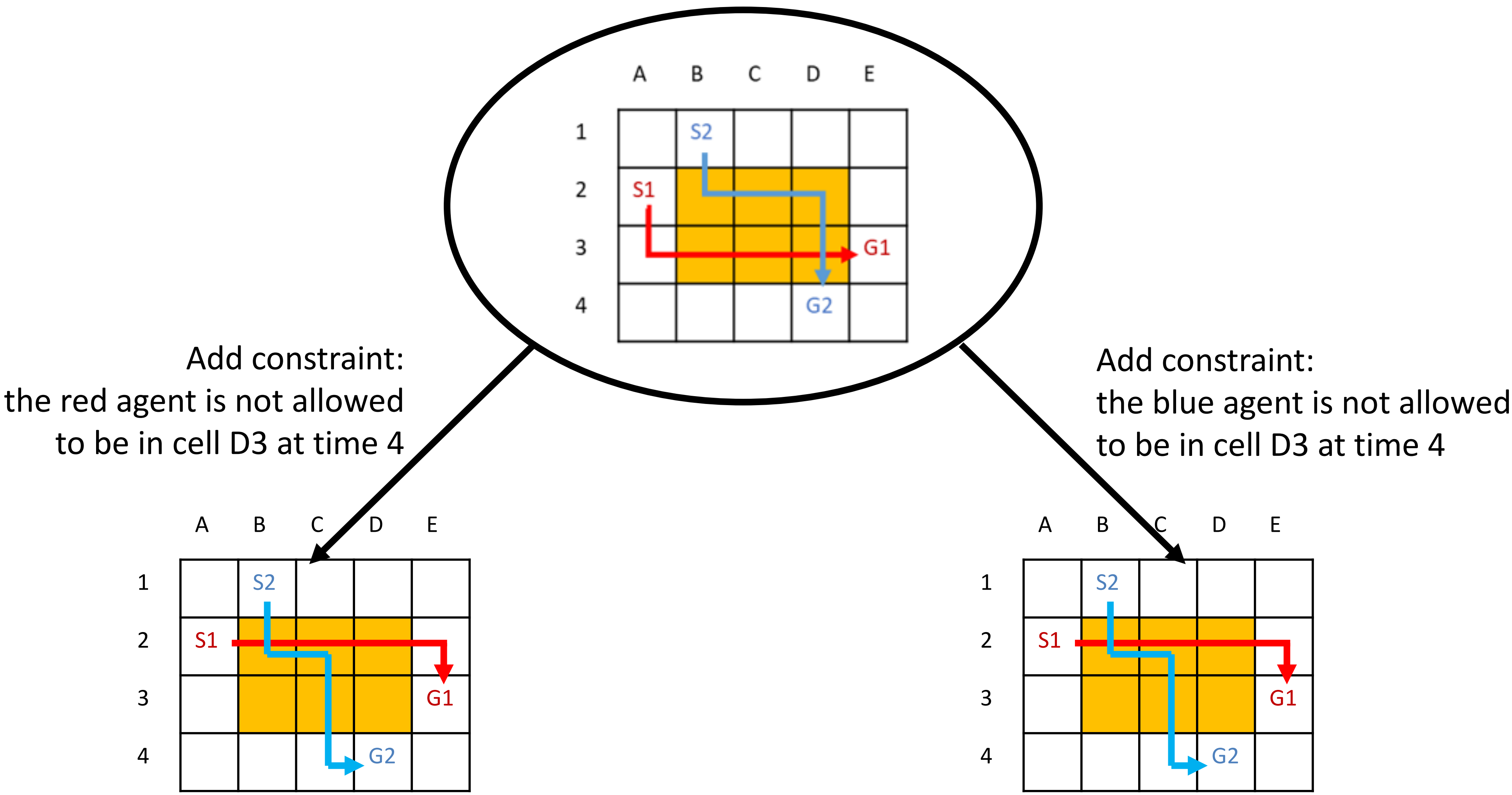
barrier constraints



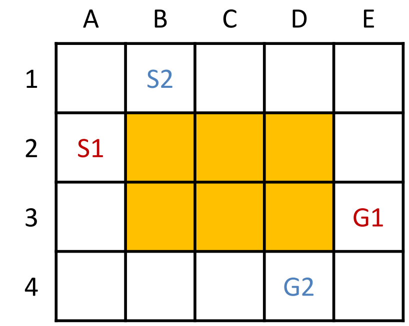
Improvement 3



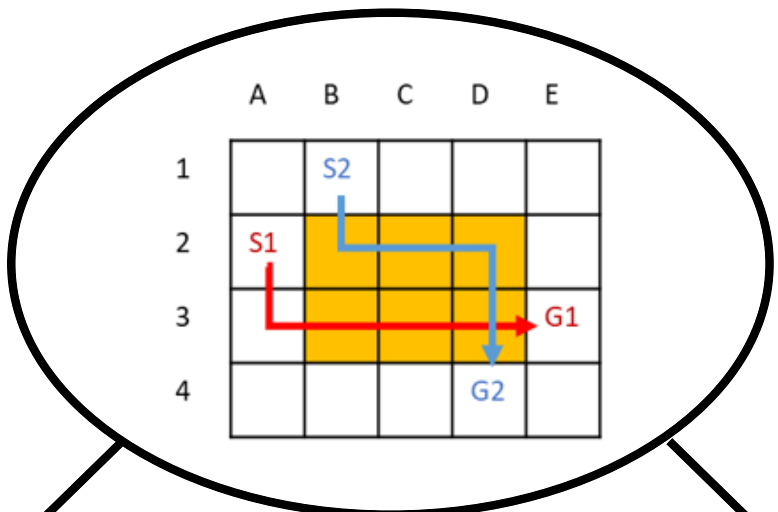
- Disjoint splitting



Improvement 3

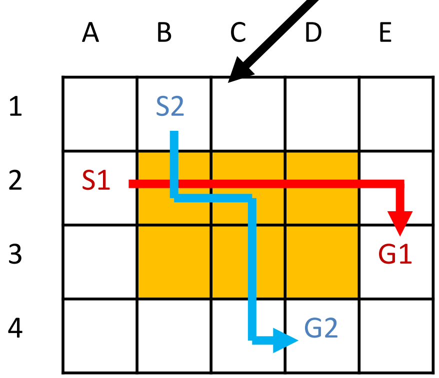


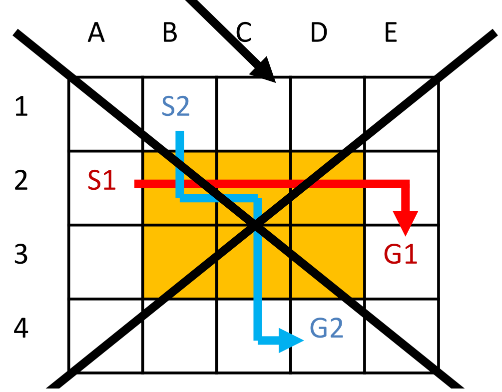
- Disjoint splitting

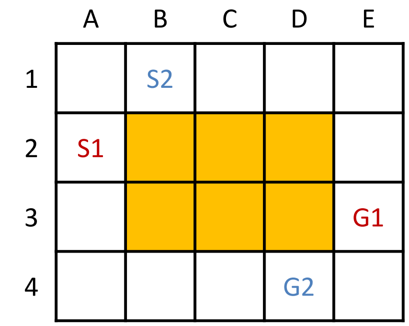


Add constraint:
the red agent is not allowed
to be in cell D3 at time 4

Add constraint:
the red agent must
be in cell D3 at time 4

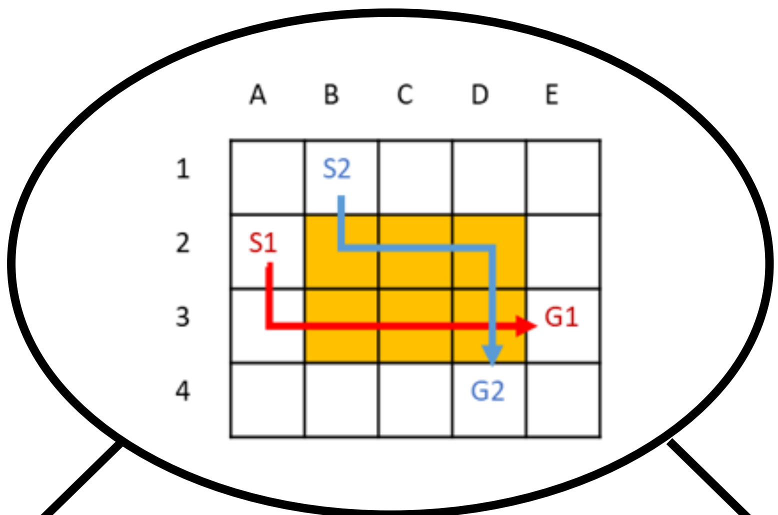






Improvement 3

- Disjoint splitting



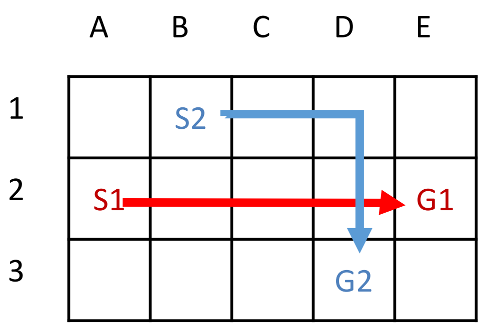
Add constraint:
the red agent is not allowed
to be in cell D3 at time 4

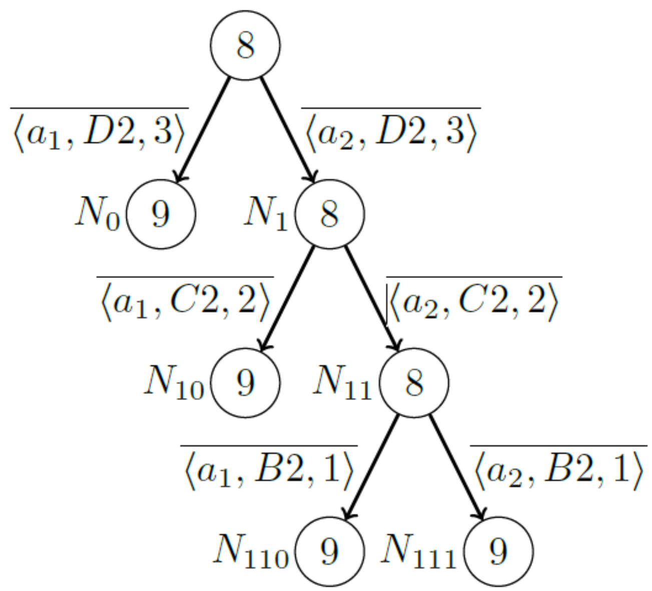
Add constraint:
the red agent must
be in cell D3 at time 4
which implies that all
the other agents are not
allowed to be in cell D3
at time 4

4-neighbor grid

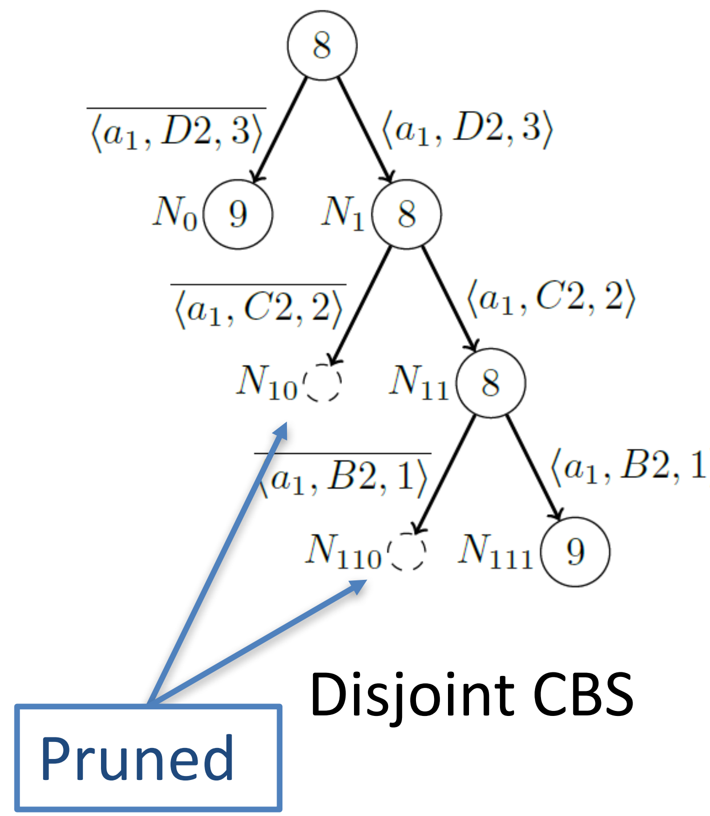
Improvement 3

- Disjoint splitting





Original CBS

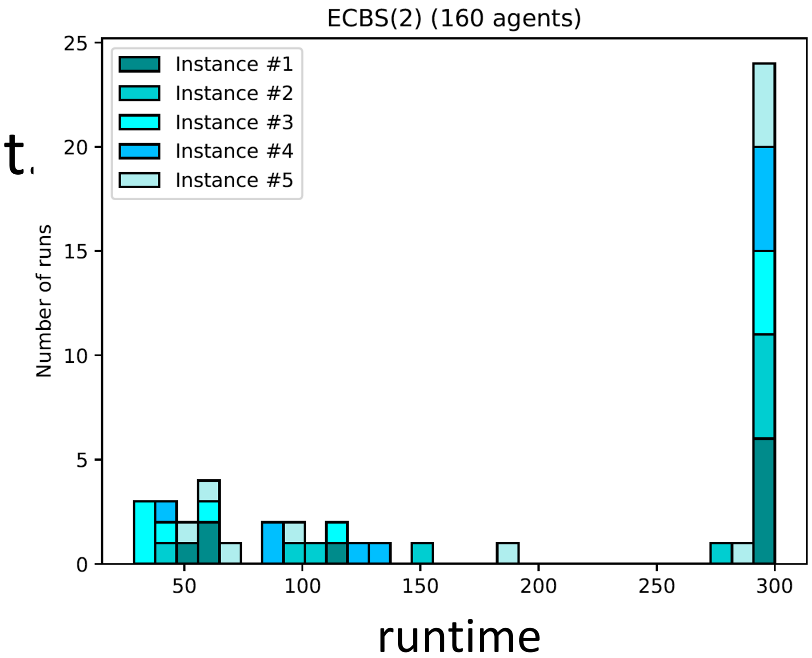


Disjoint CBS

4-neighbor grid

Improvement 4

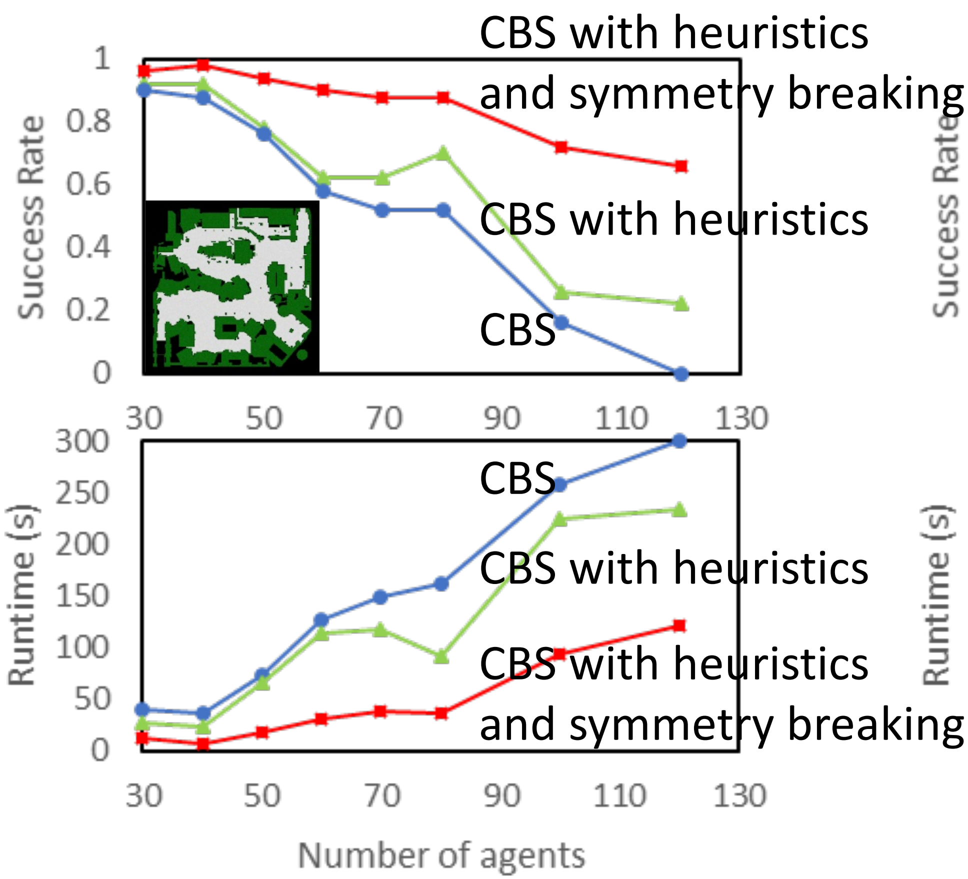
- Rapid random restarts help to solve more multi-agent path finding problems within a given runtime limit.
- Here: We randomize the ordering in which the agents plan their paths in the high-level root node.



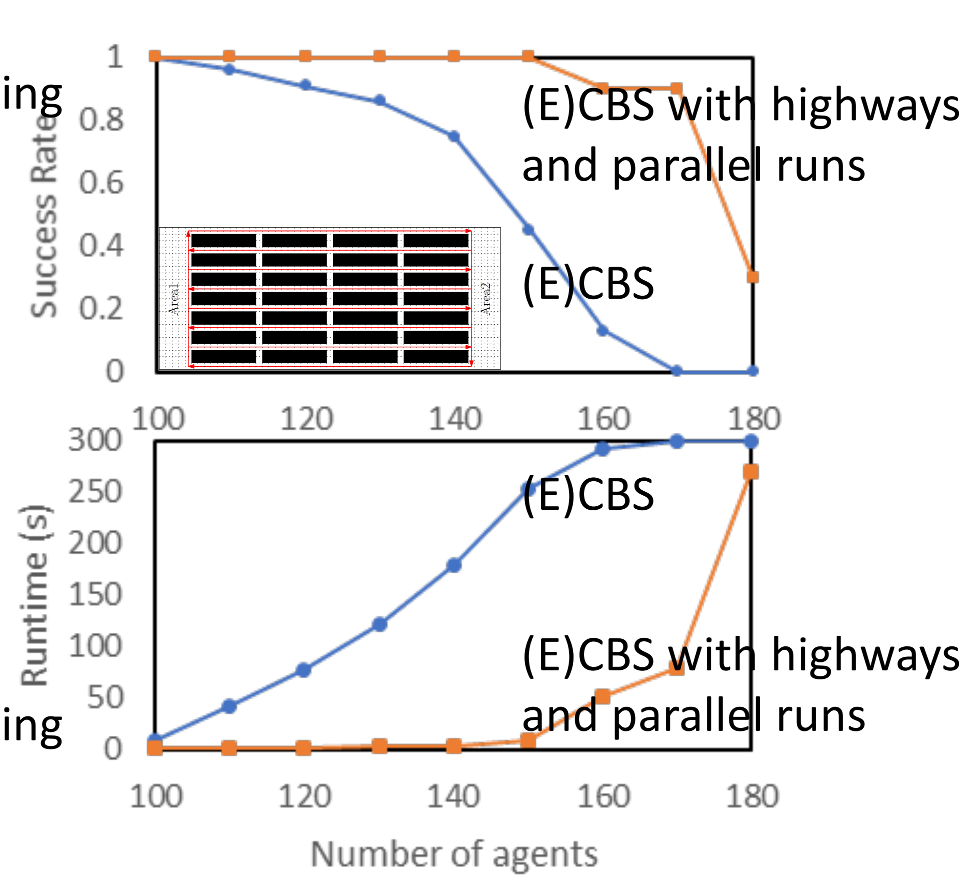
runs	time limit	38 “easy”	12 “hard”	50 total
1	300 sec	100.00%	0.00%	76.00%
3	100 sec	97.65%	96.87%	97.60%
5	60 sec	98.57%	98.81%	98.70%

Conflict-Based Search

Optimal MAPF Planning

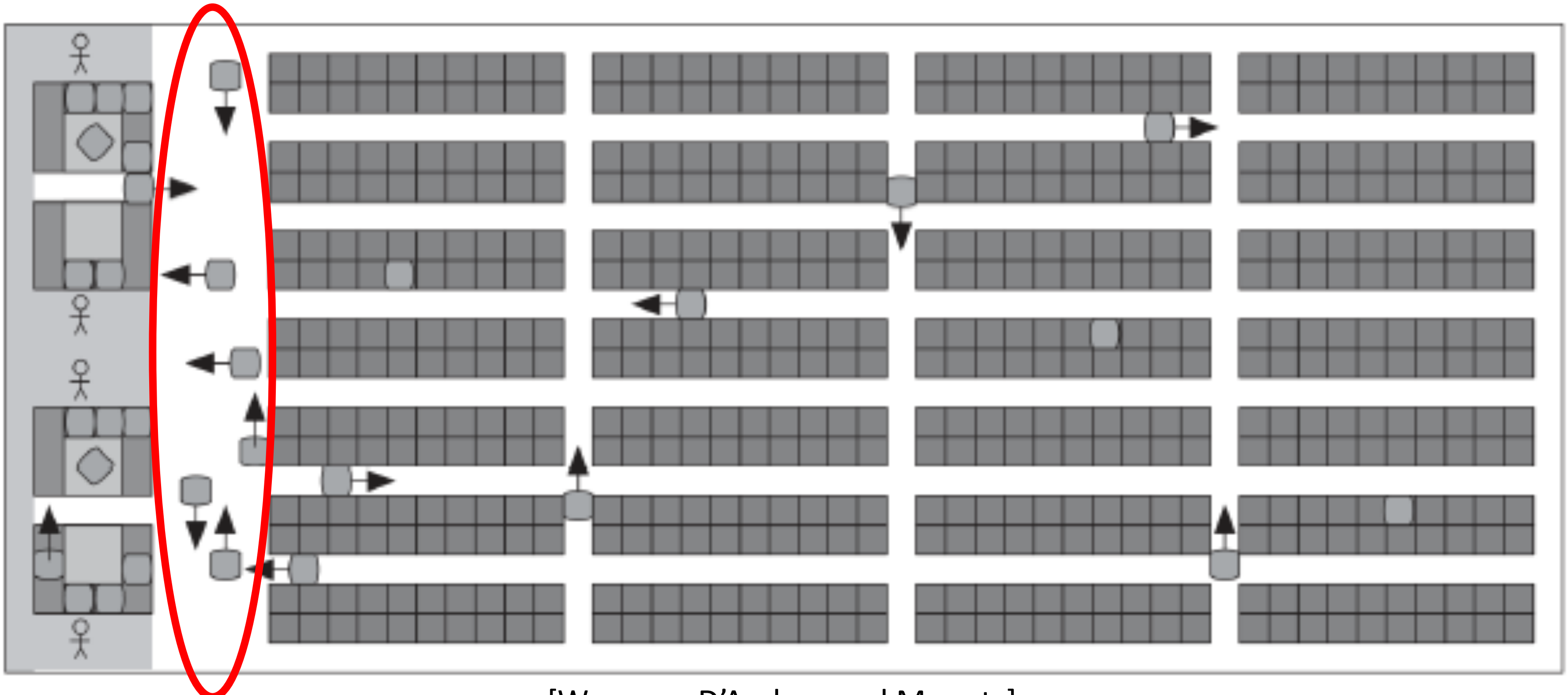


Bounded-Suboptimal MAPF Planning



4-neighbor grid

Conflict-Based Search

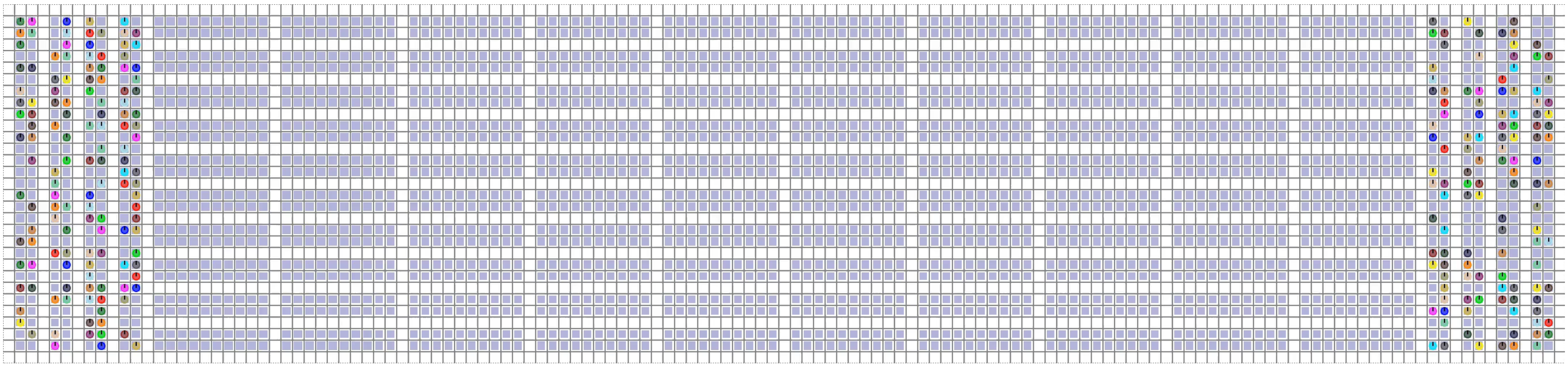


[Wurman, D'Andrea and Mountz]

4-neighbor grid

Lifelong Multi-Agent Path Finding

- Runtime on 135x31 grids
  - 250 agents and 20,000 random pickup-and-delivery tasks
  - Makespan ≈ 0.5 hour
  - Mean total planning time ≈ 10s



4-neighbor grid

More Information on MAPF

- Go to mapf.info for more information on MAPF

Acknowledgments

- This tutorial reported on joint work with a large number of collaborators (including students) from the University of Southern California and elsewhere. We would like to acknowledge their contributions
- Special thanks to K. Arras, A. Arunasalam, N. Ayanian, E. Boyarski, T. Cai, D. Chan, H. Choset, L. Cohen, K. Daniel, A. Felner, D. Harabor, C. Hernandez, W. Hoenig, S. Jahangiri, T. K. S. Kumar, M. Likhachev, H. Ma, P. Meseguer, A. Nash, L. Palmieri, G. Sharon, X. Sun, P. Stuckey, N. Sturtevant, C. Tovey, T. Uras, G. Wagner, H. Xu, W. Yeoh, S. Young and D. Zhang
- Thanks to Amazon, ARO, IBM, JPL, NSF, ONR for funding!

Acknowledgments

- Please visit idm-lab.org/projects.html for more information, pointers to the literature and our publications
- If you have any interesting ideas, please send me an email: skoenig@usc.edu