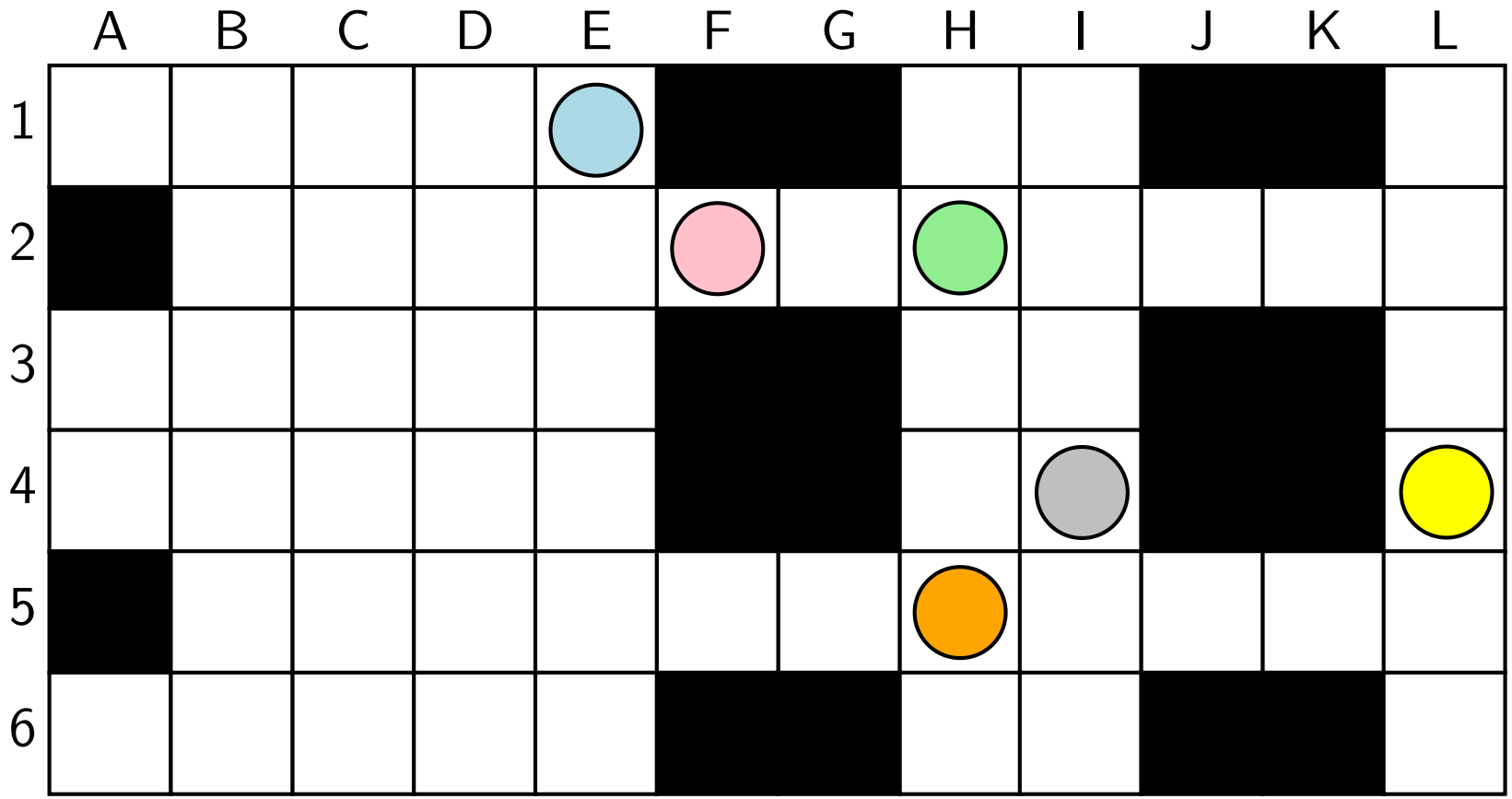
Multi-Agent Path Finding

Bojie Shen
Monash University

https://bojie-shen.com/

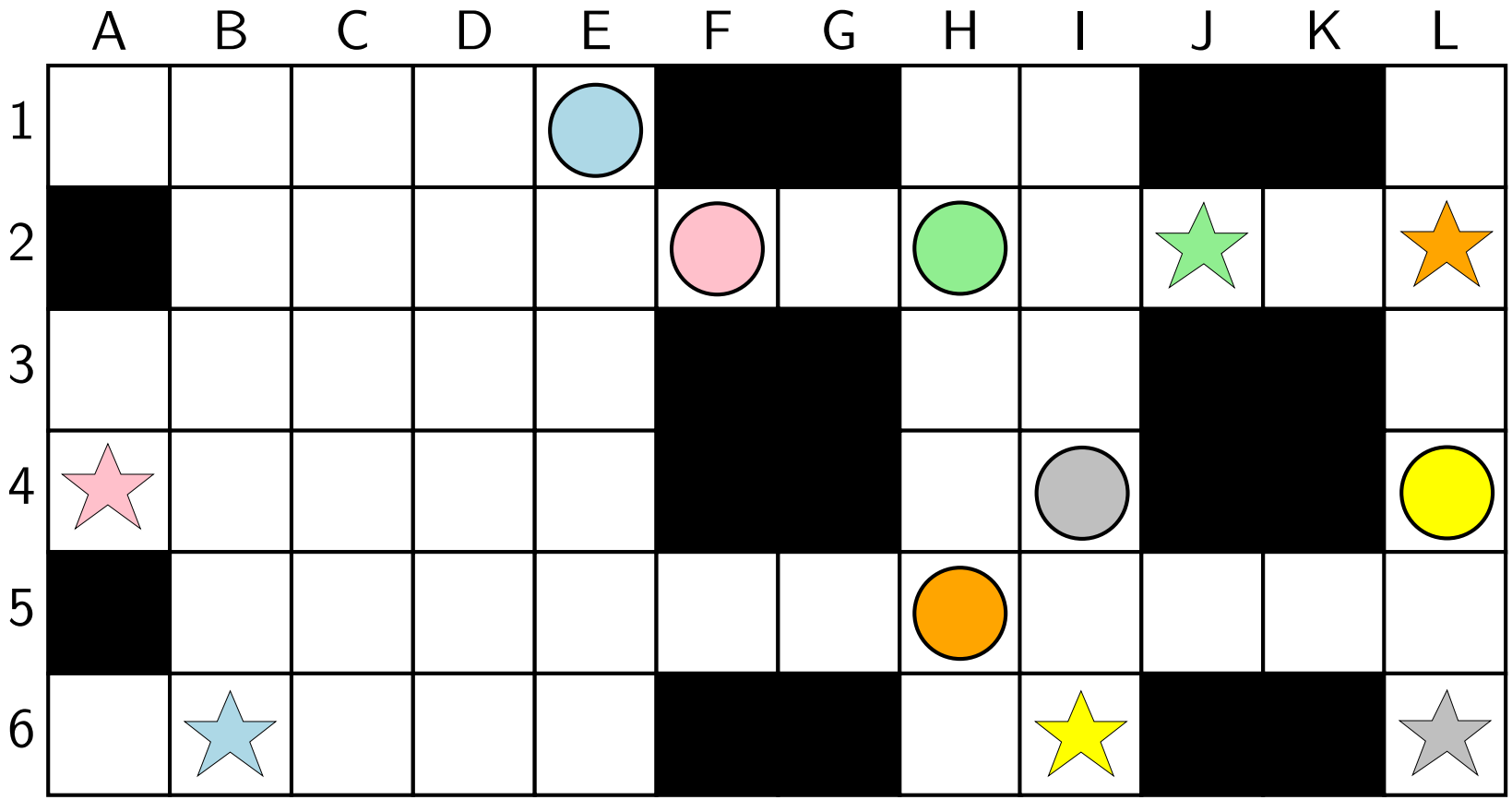
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



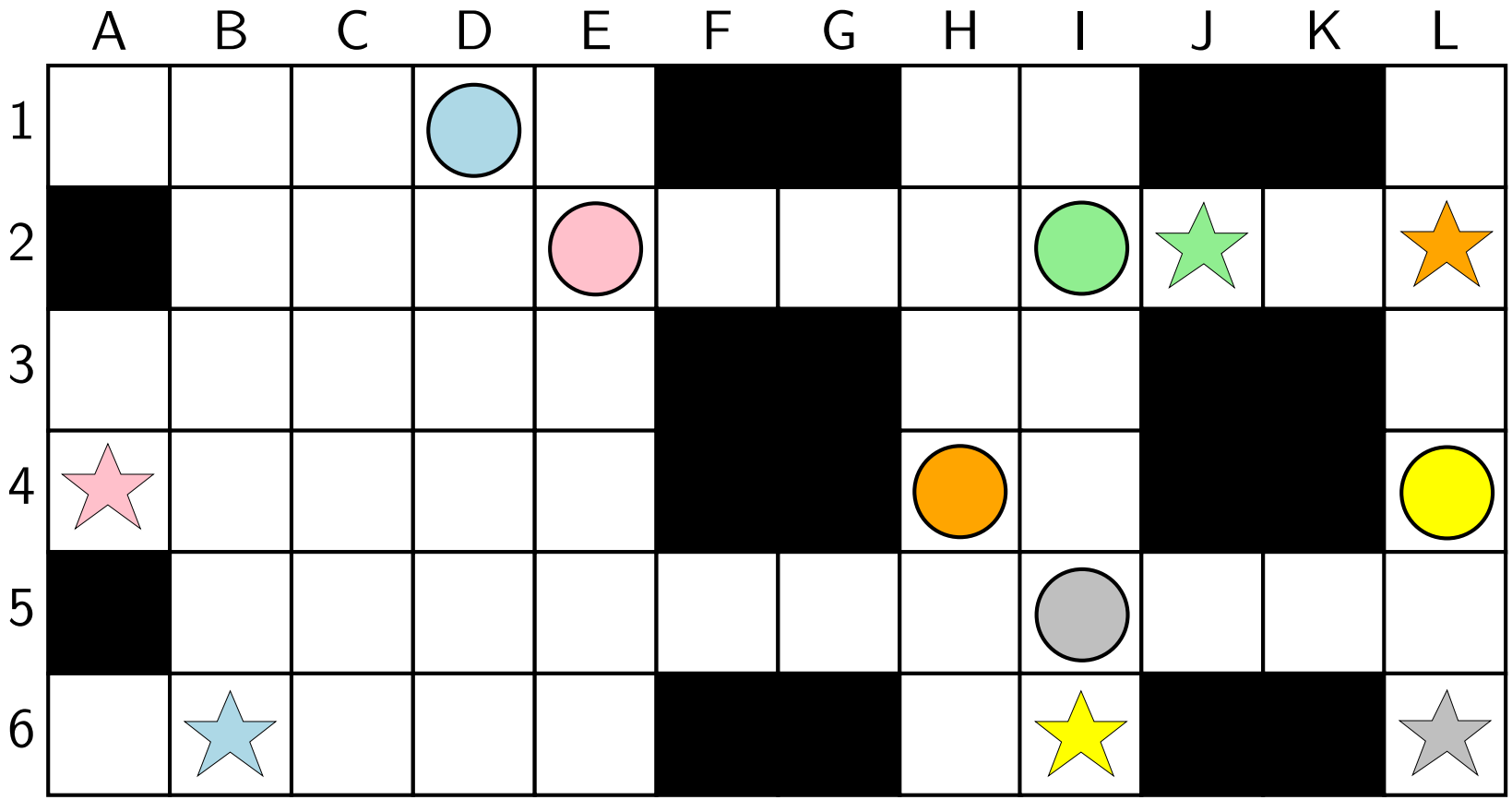
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



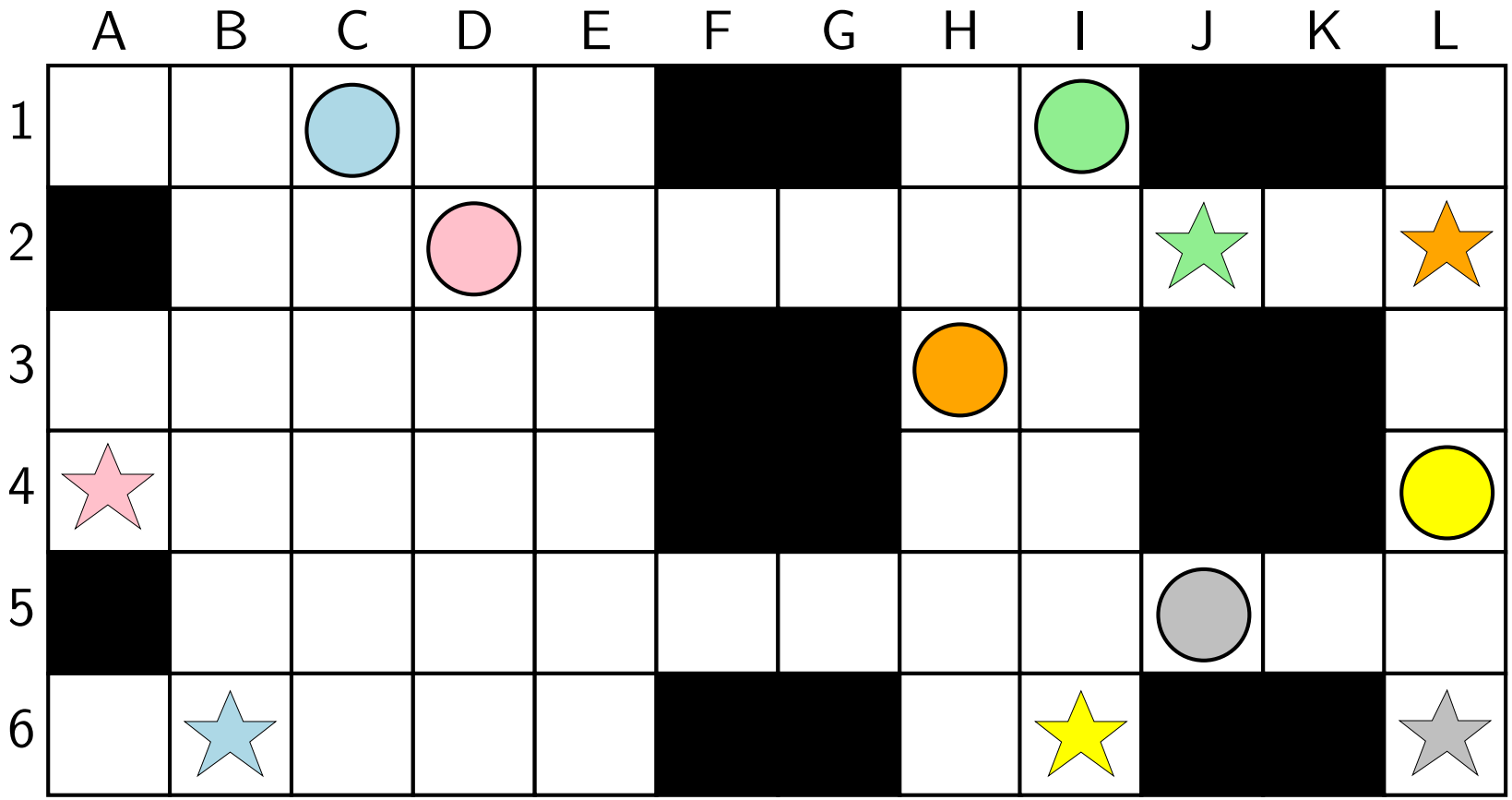
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



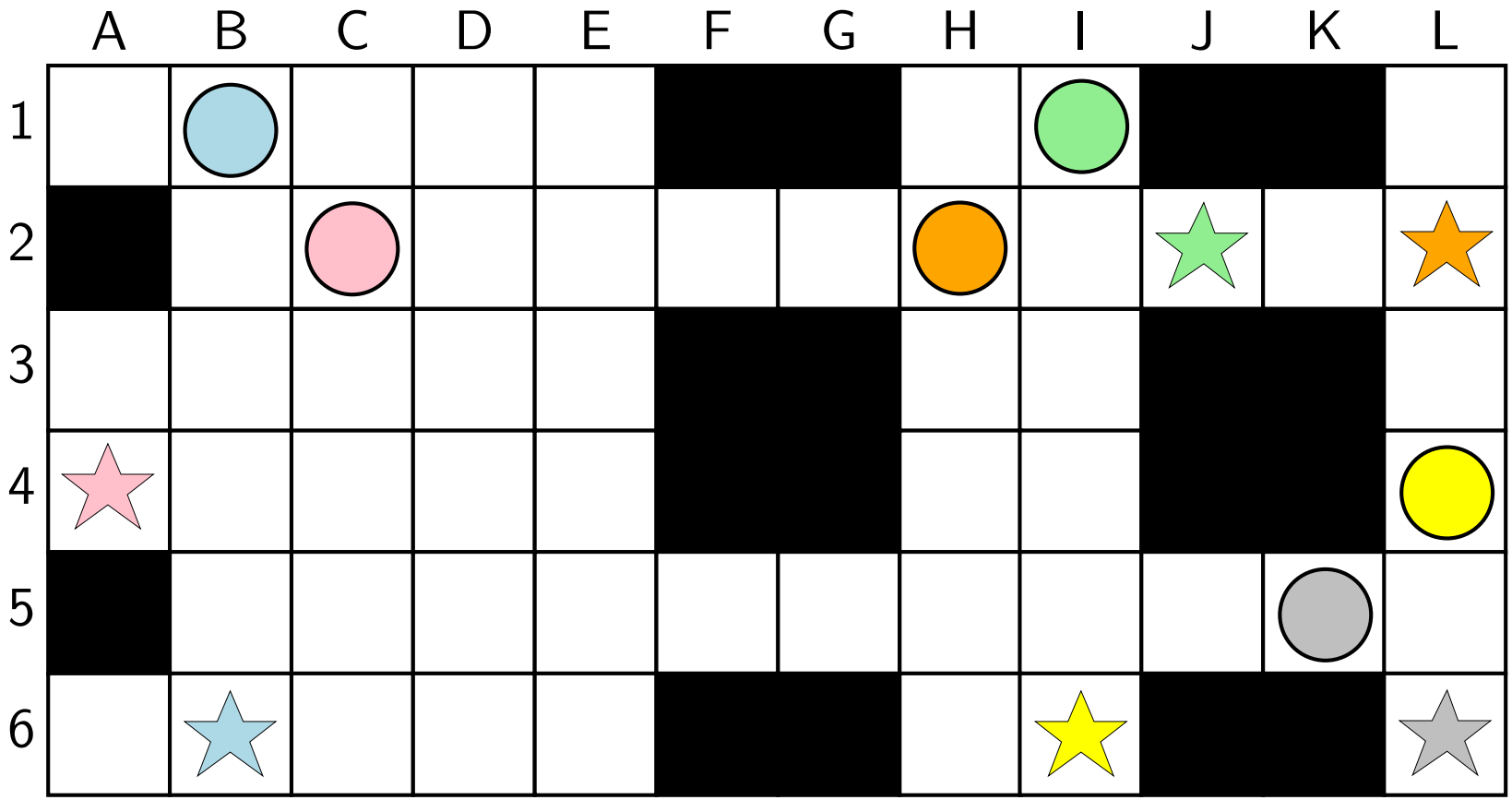
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



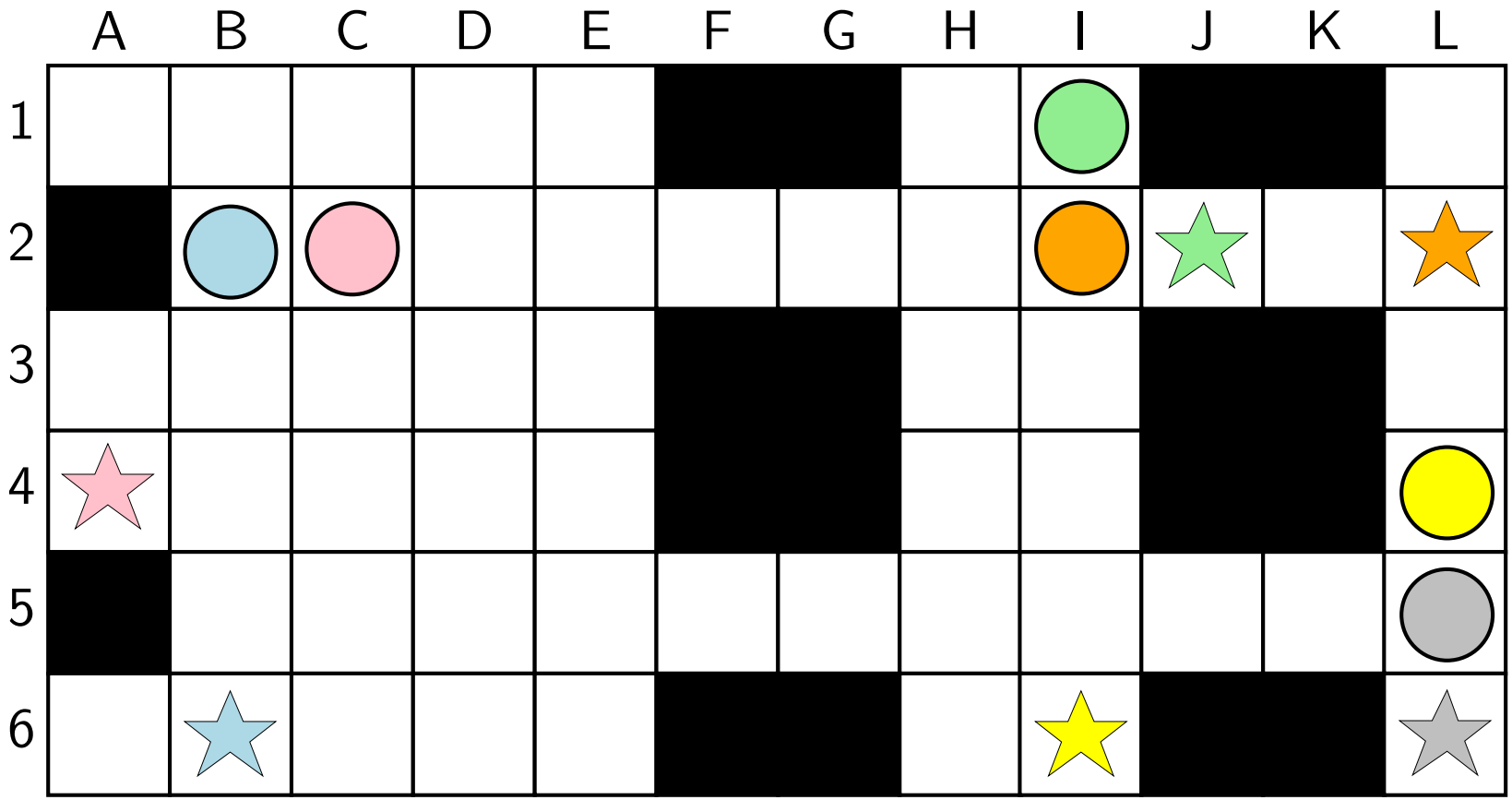
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



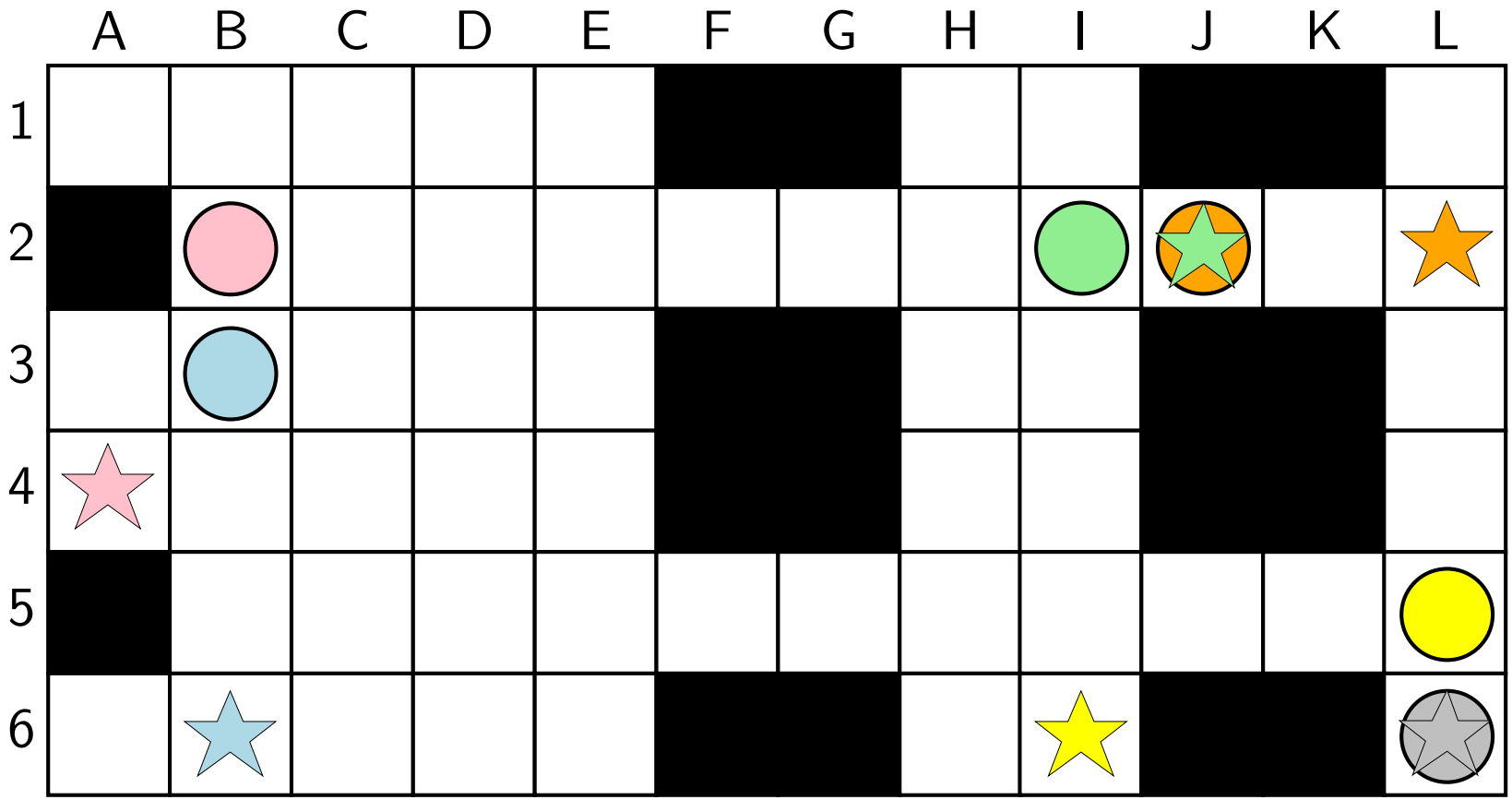
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



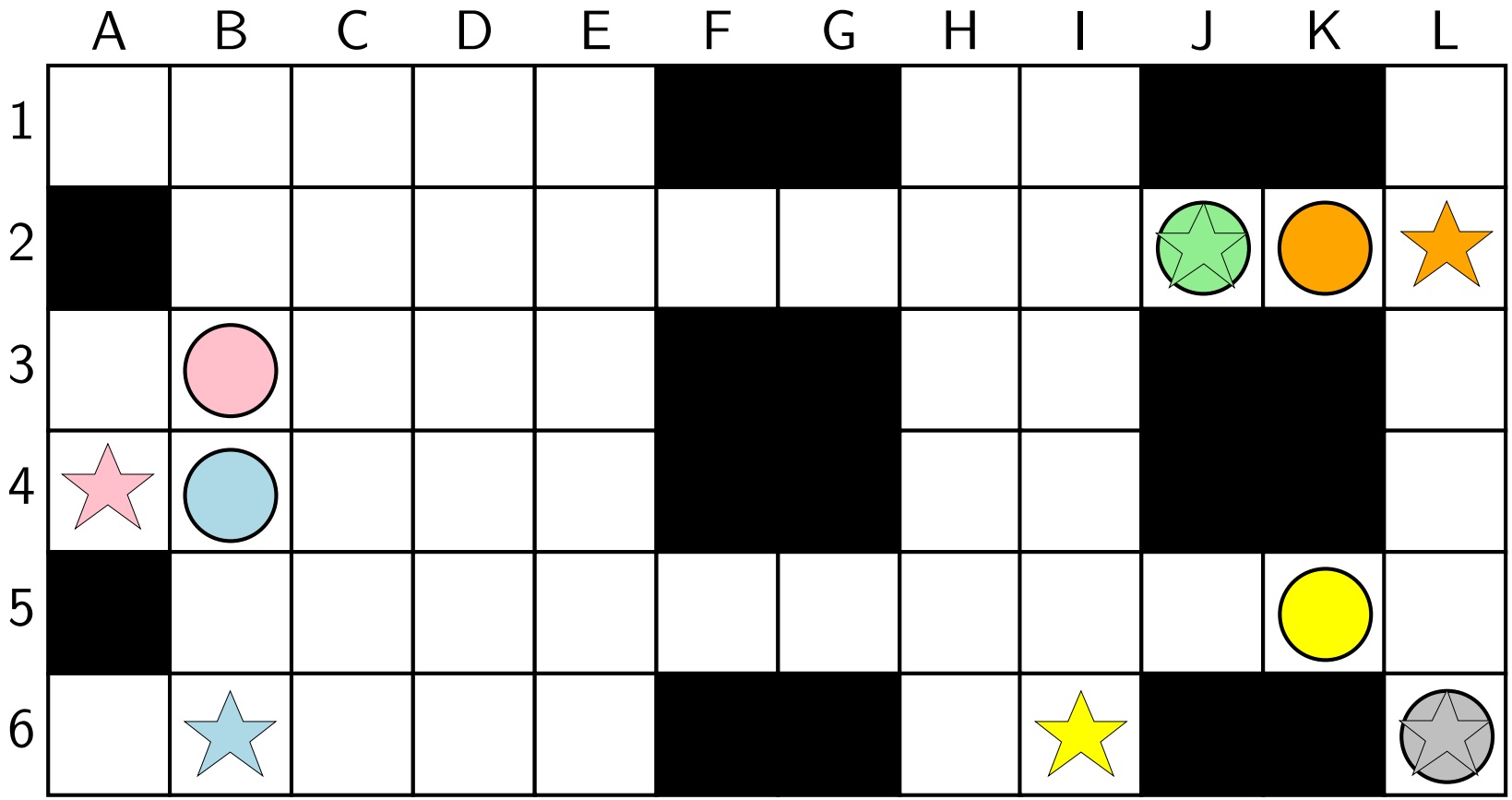
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



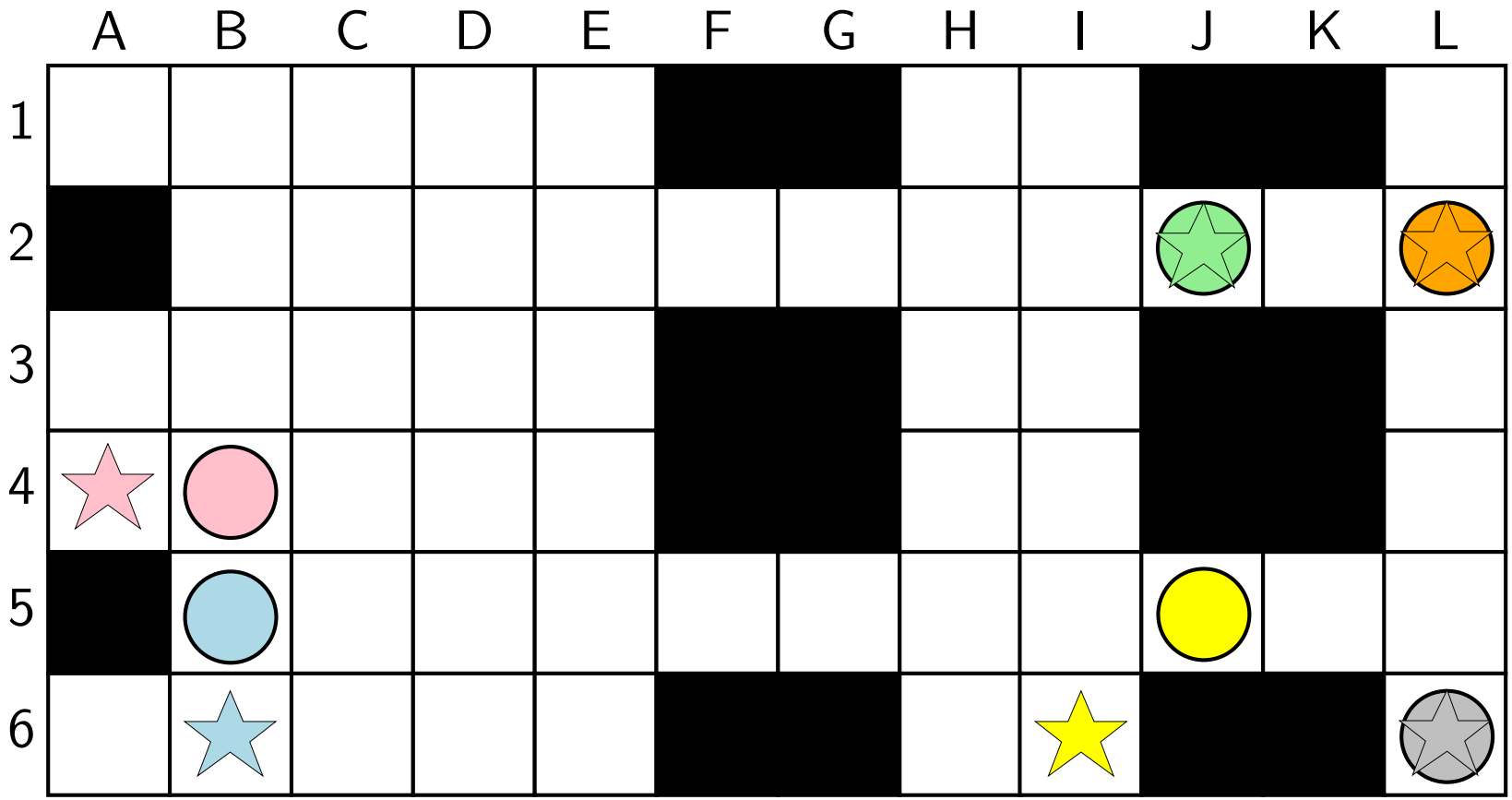
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



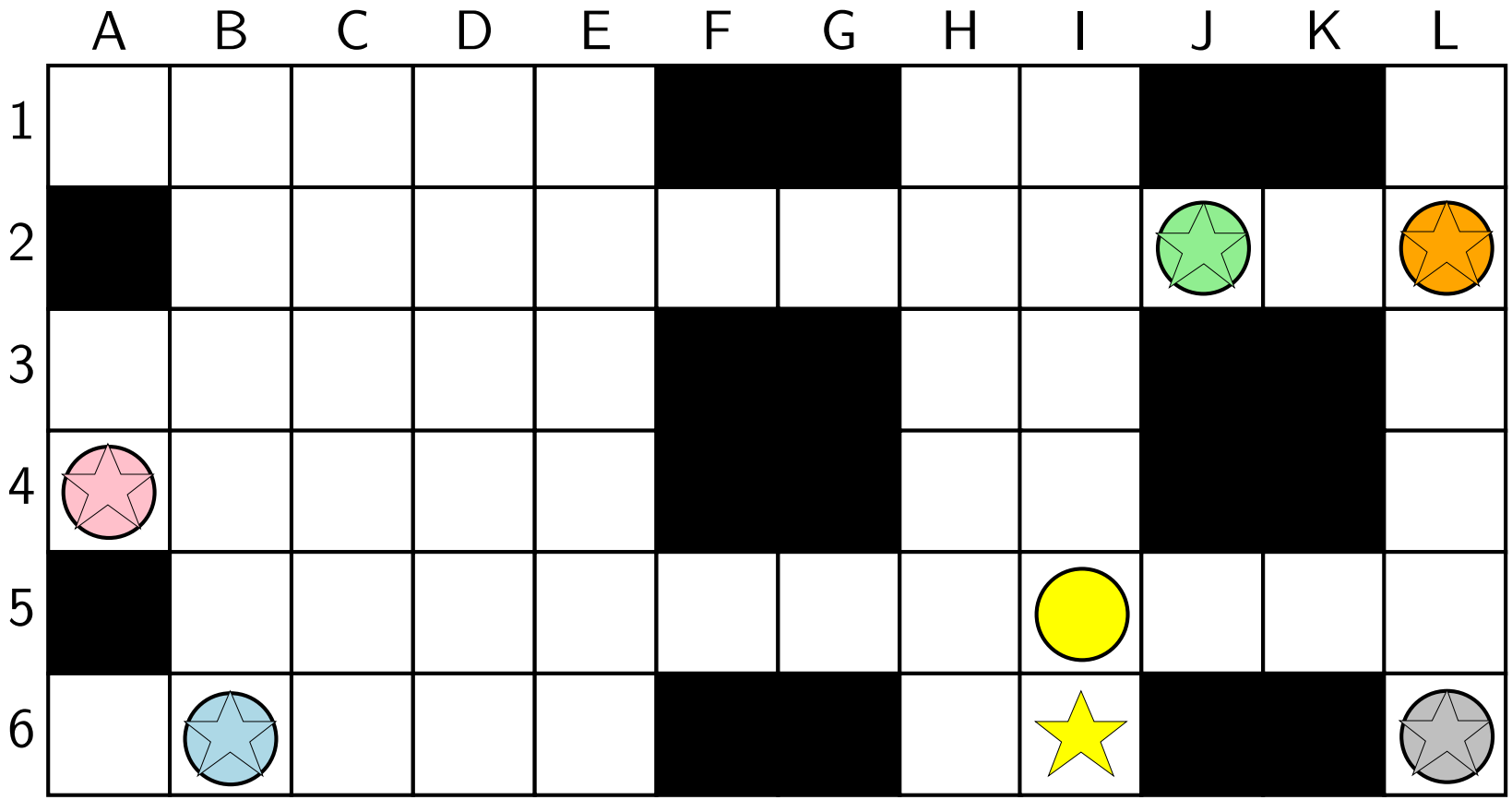
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



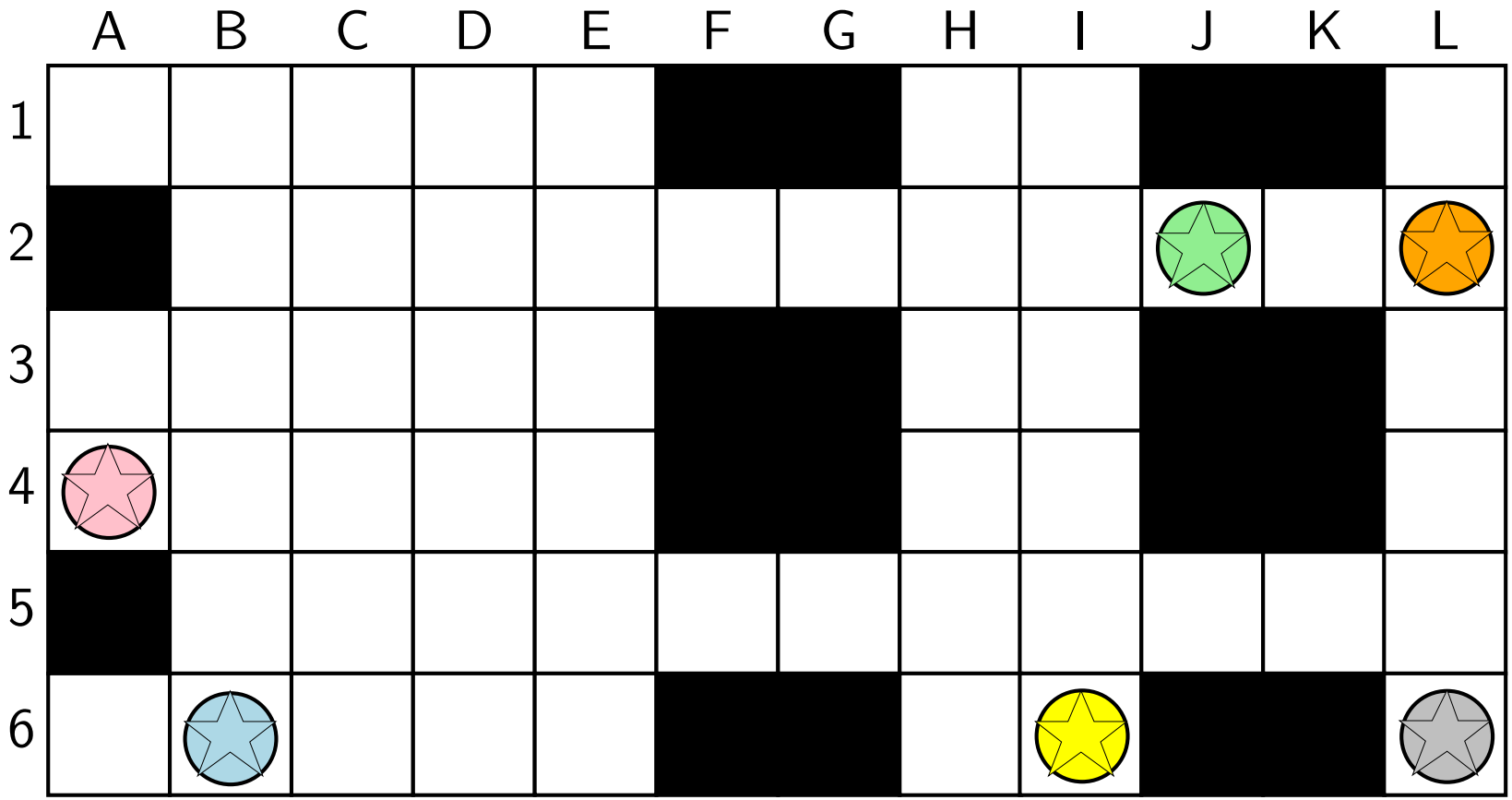
Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



Recap

The Multi-agent Path Finding problem (MAPF) asks for a collision-free plan that moves a team of agents.



Classical MAPF

Setup

- Each agent begins at a specific start position
- Each agent finishes at a specific target position
- All agents move on a 4-connected grid
- All actions incur the same unit cost
- Waiting is allowed

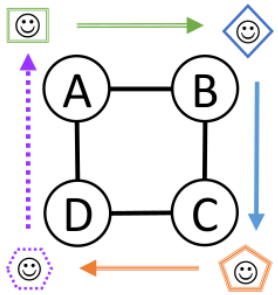
Objectives

- **Sum of Costs**: minimise the total cost of all assigned paths.
- **Makespan**: minimise the arrival time of the last agent.

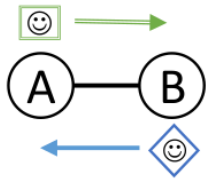
Classical MAPF

Constraints

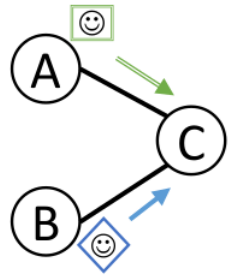
- Two agents cannot occupy a grid cell at the same time
- Two agents cannot swap positions at the same time
- Agents **can** move in "lockstep"
- Agents **arrive** their destination if they can remain there collision-free



Lockstep movement



Edge conflict



Vertex conflict

MAPF Applications

Robocup Rescue

MAPF Applications

StarCraft II

MAPF Applications

Amazon Warehouse

MAPF in Practice

Applications of MAPF often have many additional features. But the core coordination problem is always there.

	MAPF Problem	**Amazon Problem**
Environment	Grid	Grid
Workload	One shot	Lifelong
Kinematics	No	Yes
Objective	Sum-of-Costs	¯_(ツ)_/¯
Solutions	Optimal	Best available

MAPF approaches

Centralised

The agents are planned together by a single omniscient controller.

- Coupled: agents are planned **jointly**.
- Decoupled: agents are planned **independently**, subject to constraints.

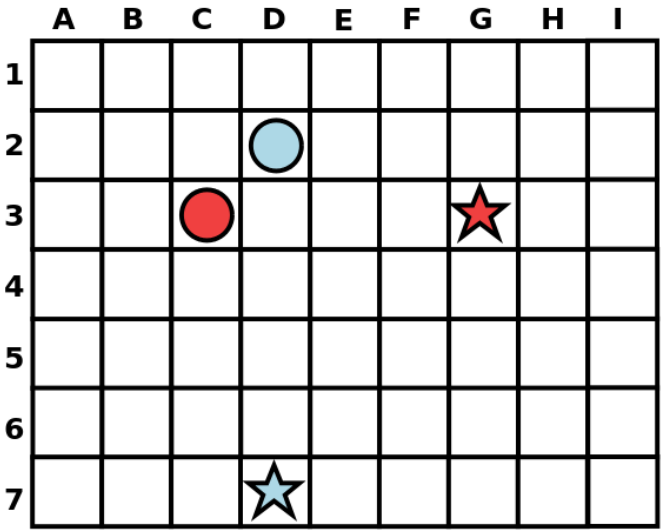
Distributed

The agents are planned independently, without global knowledge.

- Each agent has its own model of the task environment
- Each agent plans for itself and starts executing
- Conflicts are detected during execution and agents replan
- (Optional) Local information sharing and local negotiation

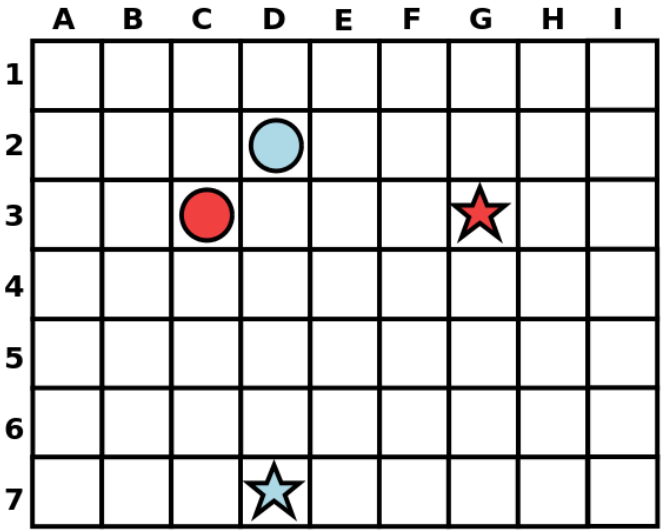
Centralised A*

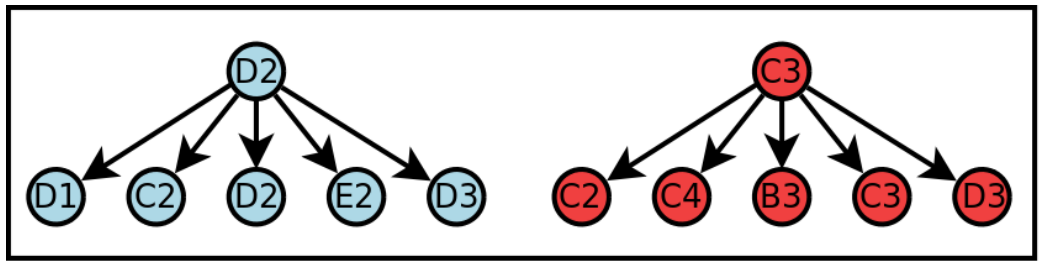
Idea: Joint-space planning. We treat treat the group of agents, A, as one large agent with many degrees-of-freedom.



Centralised A*

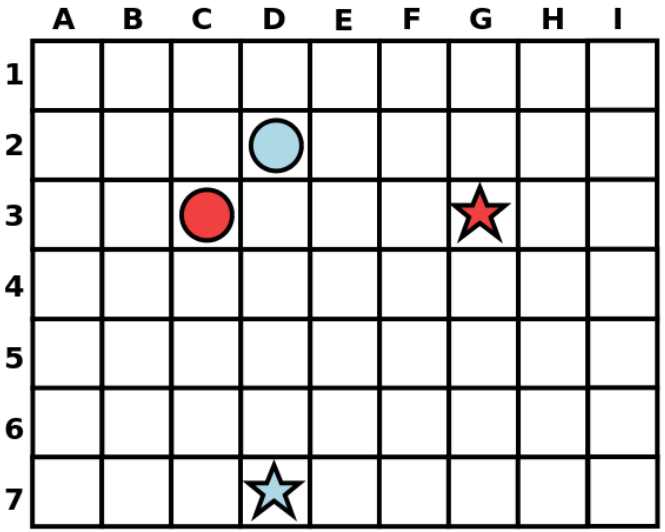
Idea: Joint-space planning. We treat treat the group of agents, A, as one large agent with many degrees-of-freedom.

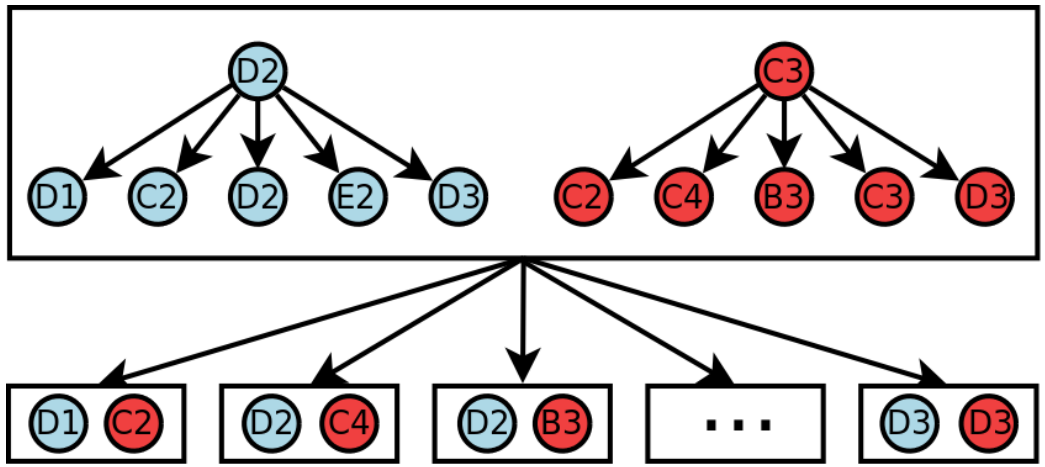




Centralised A*

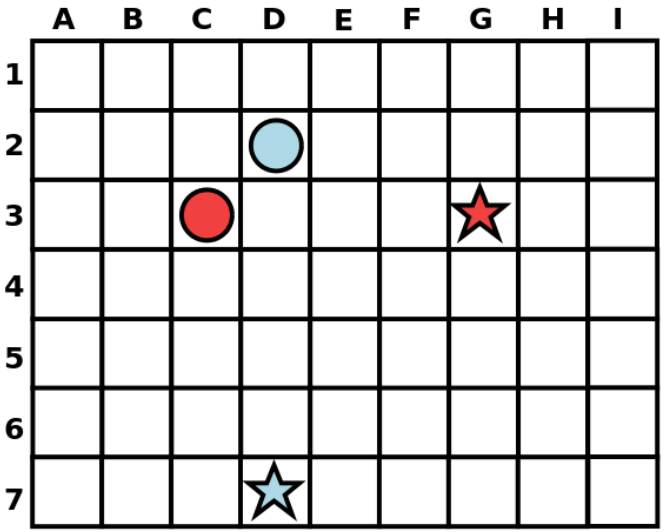
Idea: Joint-space planning. We treat treat the group of agents, A, as one large agent with many degrees-of-freedom.

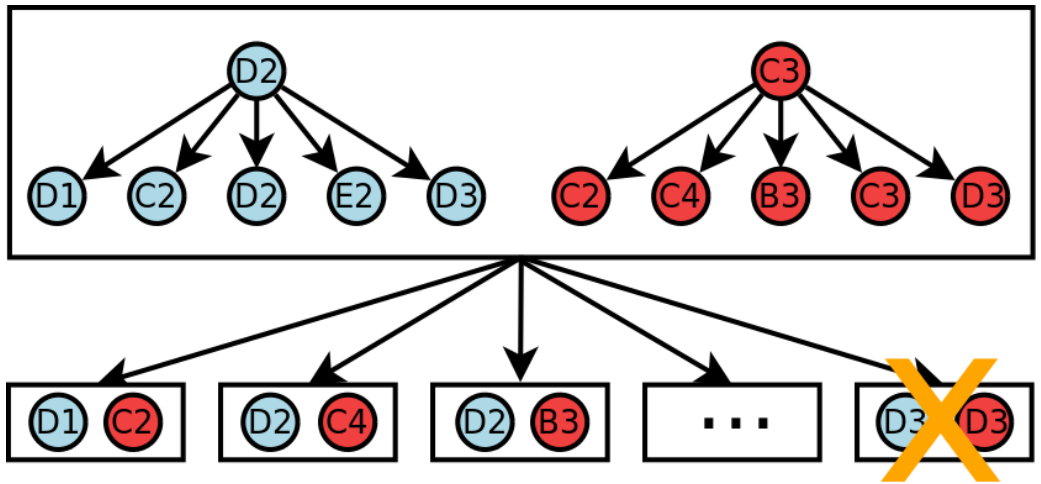




Centralised A*

Idea: Joint-space planning. We treat treat the group of agents, A, as one large agent with many degrees-of-freedom.





Analysis of Centralised A*

Joint-space planning is complete and optimal. But it doesn't scale and is often a non-starter.

Analysis of Centralised A*

Joint-space planning is complete and optimal. But it doesn't scale and is often a non-starter.

Branching factor is $b^{|A|}$. On 4-C gridmaps we have $b = 5$.

- 2 agents $= 5^2$ successors ($= 25$; acceptable)

Analysis of Centralised A*

Joint-space planning is complete and optimal. But it doesn't scale and is often a non-starter.

Branching factor is $b^{|A|}$. On 4-C gridmaps we have $b = 5$.

- 2 agents $= 5^2$ successors ($= 25$; acceptable)
- 3 agents $= 5^3$ successors ($=125$; challenging)

Analysis of Centralised A*

Joint-space planning is complete and optimal. But it doesn't scale and is often a non-starter.

Branching factor is $b^{|A|}$. On 4-C gridmaps we have $b = 5$.

- 2 agents $= 5^2$ successors ($= 25$; acceptable)
- 3 agents $= 5^3$ successors ($=125$; challenging)
- 5 agents $= 5^5$ successors ($=3125$; unreasonable)

Analysis of Centralised A*

Joint-space planning is complete and optimal. But it doesn't scale and is often a non-starter.

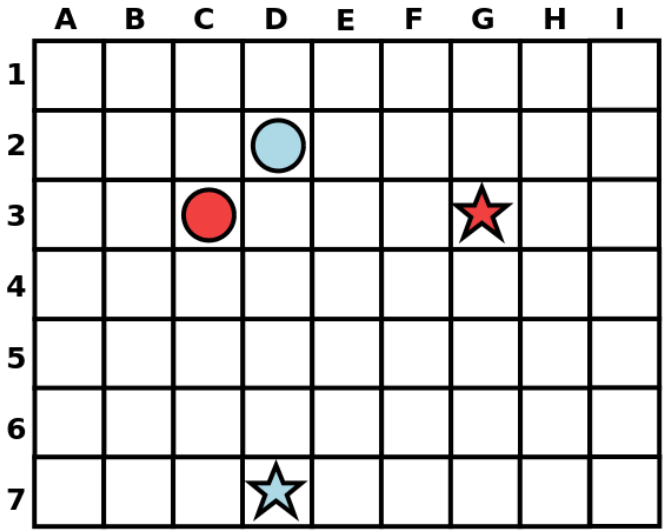
Branching factor is $b^{|A|}$. On 4-C gridmaps we have $b = 5$.

- 2 agents $= 5^2$ successors ($= 25$; acceptable)
- 3 agents $= 5^3$ successors ($=125$; challenging)
- 5 agents $= 5^5$ successors ($=3125$; unreasonable)
- 10 agents $= 5^{10}$ ($=$ **lolno**)

CA* performance can be improved using *partial expansion*. This strategy expands a node multiple times, each time generating only the most promising successors from those that remain. This is a very useful idea and generally applicable. For details see **(Felner, A. et al. 2012. Partial-Expansion A* with Selective Node Generation. In Proceedings of AAAI (pp. 180-181))**

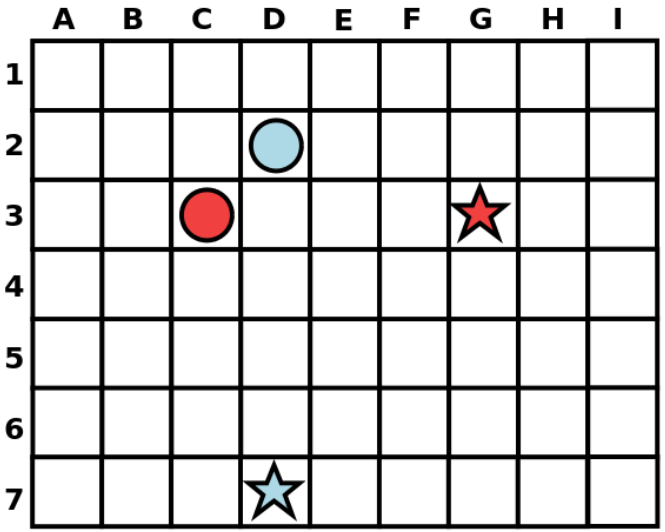
Operator Decomposition

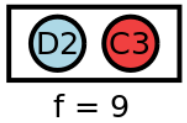
Idea: Agents select compatible moves one at a time, in some fixed order. After $|A|$ individual moves, the search advances one timestep.



Operator Decomposition

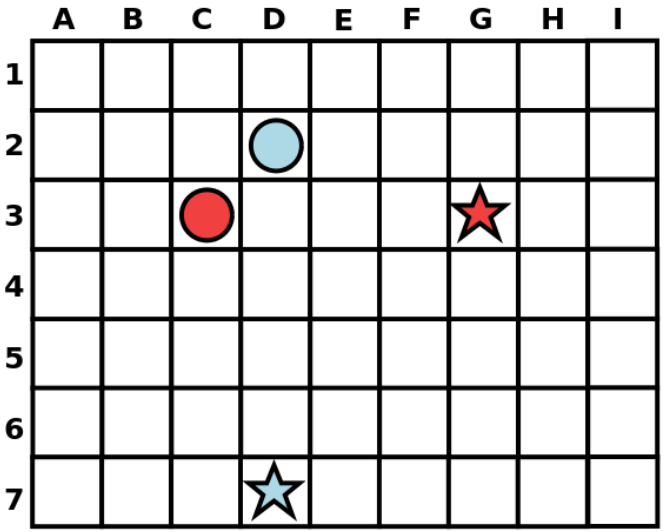
Idea: Agents select compatible moves one at a time, in some fixed order. After $|A|$ individual moves, the search advances one timestep.

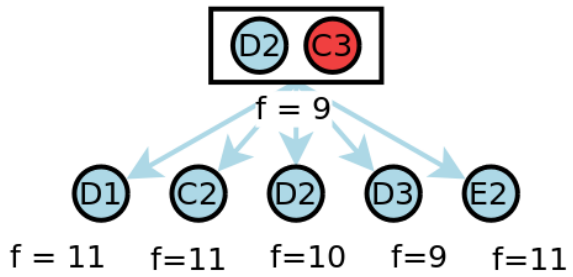




Operator Decomposition

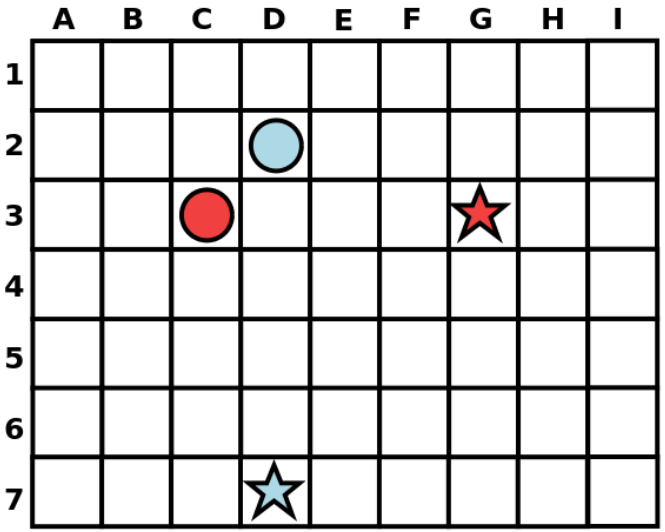
Idea: Agents select compatible moves one at a time, in some fixed order. After $|A|$ individual moves, the search advances one timestep.

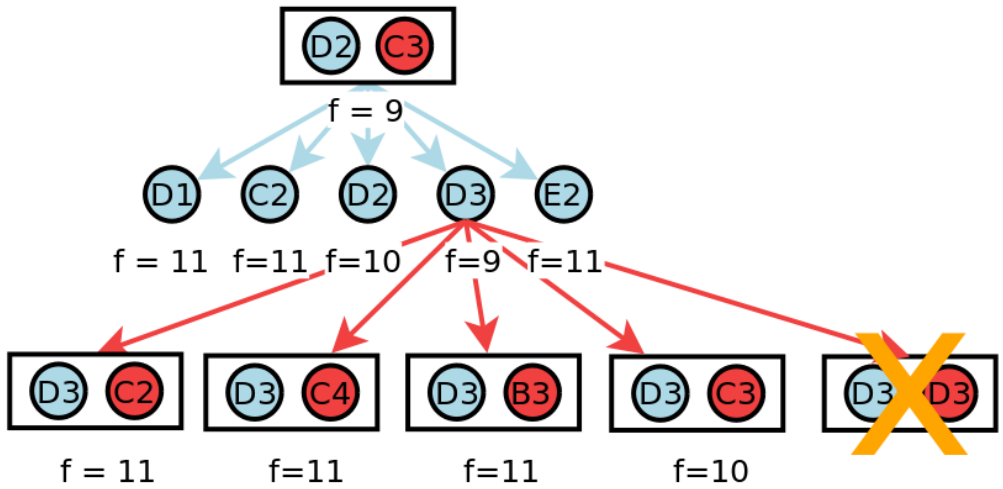




Operator Decomposition

Idea: Agents select compatible moves one at a time, in some fixed order. After $|A|$ individual moves, the search advances one timestep.





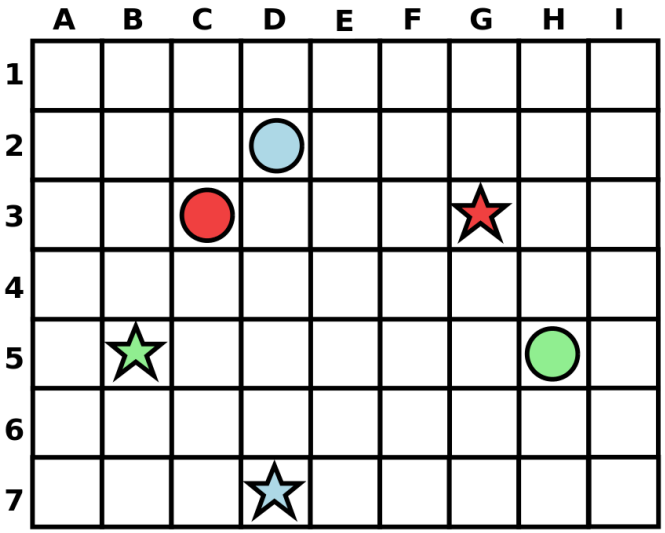
Analysis of OD

Operator Decomposition is complete and optimal.

- Branching factor reduced to b (=5 on 4-C gridmaps)
- But solutions are deeper in the tree (from d with CA* to $d \times |A|$)
- Works well for problems with small numbers of agents

Independence Detection (OD+ID)

Idea: many agents don't interact with other agents or only just a few. Try to identify such groups and solve them independently.



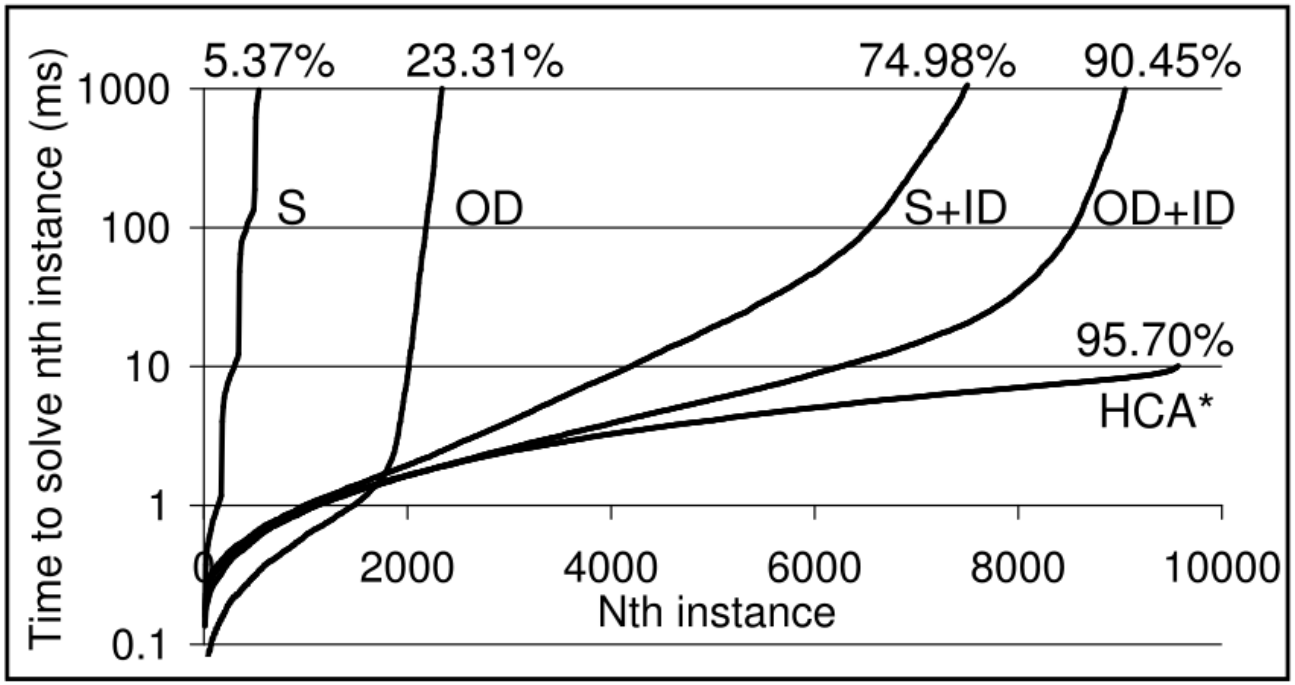
Algorithm 1 Simple Independence Detection

```
1: assign each agent to a singleton group
2: plan a path for each group
3: repeat
4:     simulate execution of all paths until a conflict occurs
5:     merge two conflicting groups into a single group
6:     cooperatively plan new group
7: until no conflicts occur
8: solution ← paths of all groups combined
9: return solution
```

OD+ID is easy to understand and, even 10 years since its introduction, still reasonably effective. It first appears in: **(Standley, T.S., 2010, July. Finding Optimal Solutions to Cooperative Pathfinding Problems. In AAAI (Vol. 1, pp. 28-29))**.

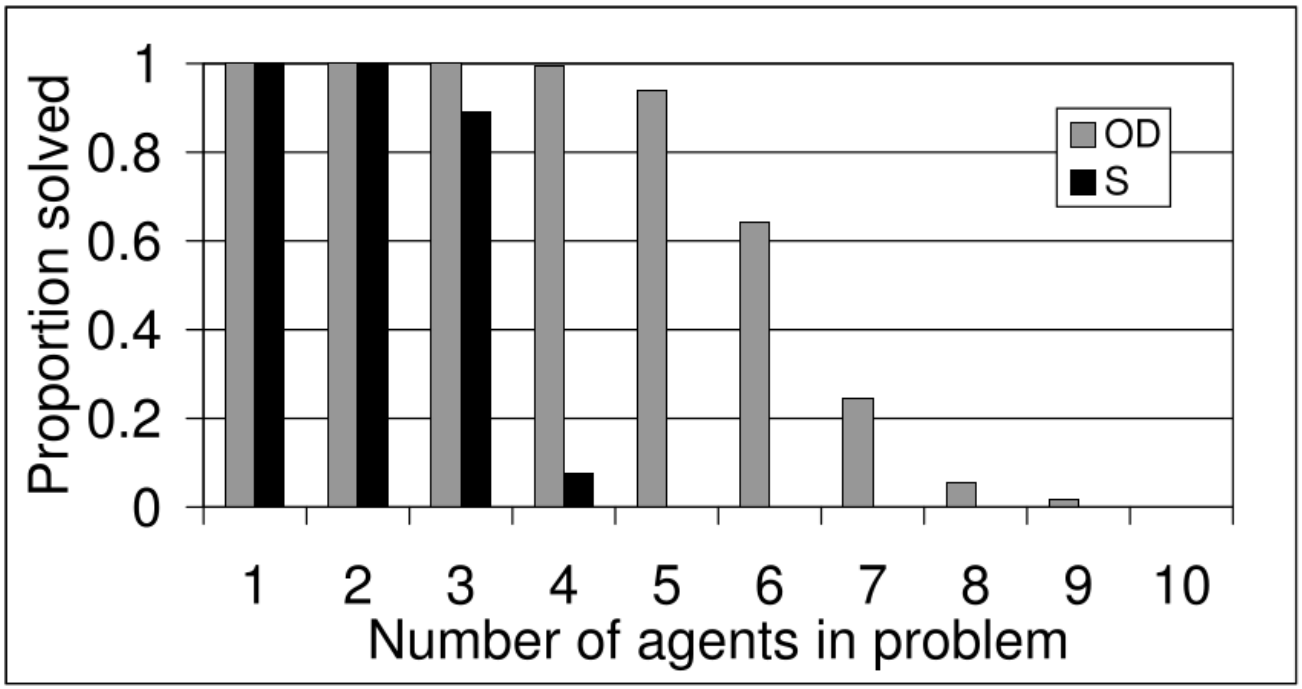
Effectiveness of OD+ID

Experiments on 32×32 grids with 20% random obstacles.
Random problems with up to 60 agents.

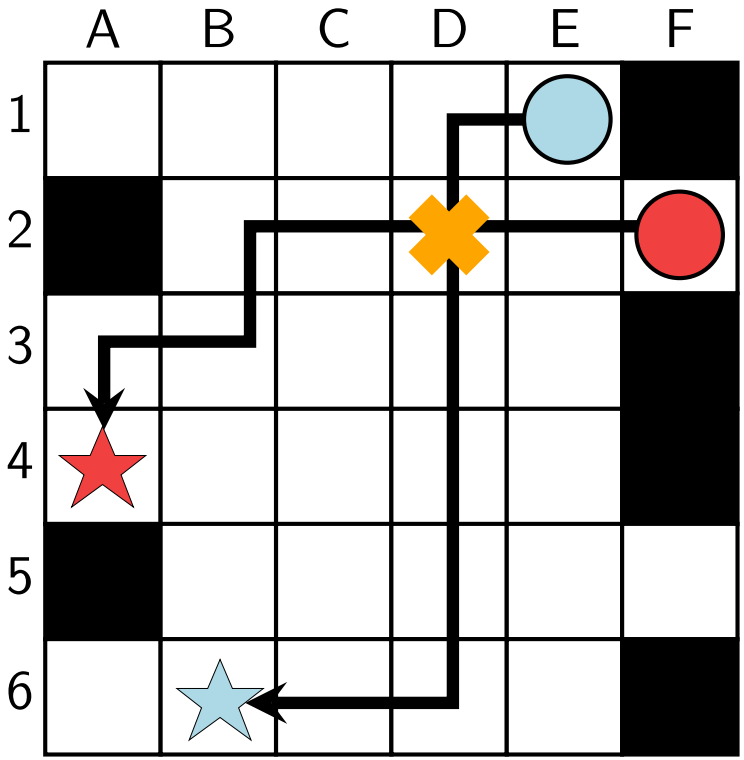


Effectiveness of OD+ID

Experiments on 32×32 grids with 20% random obstacles.
Random problems with up to 60 agents.

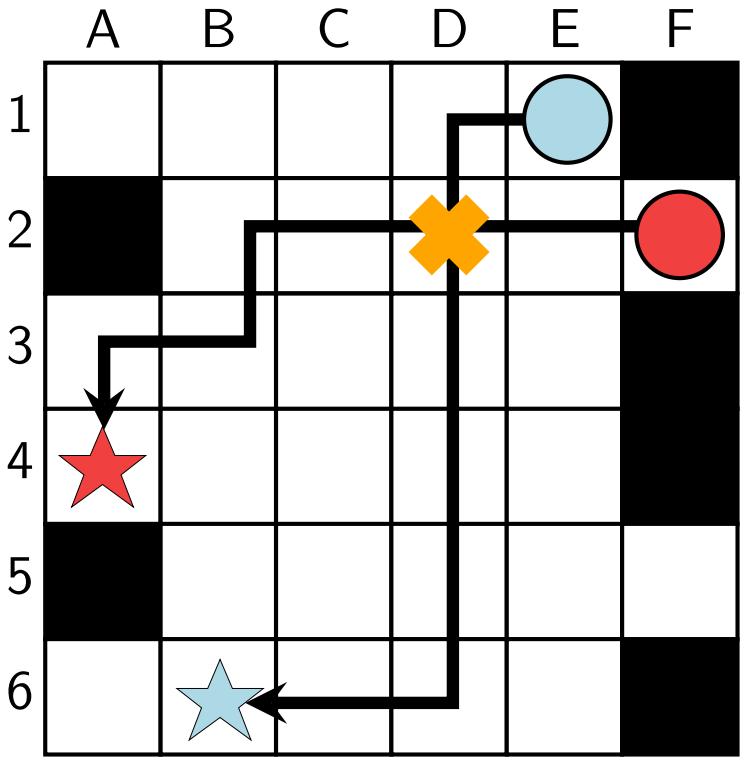


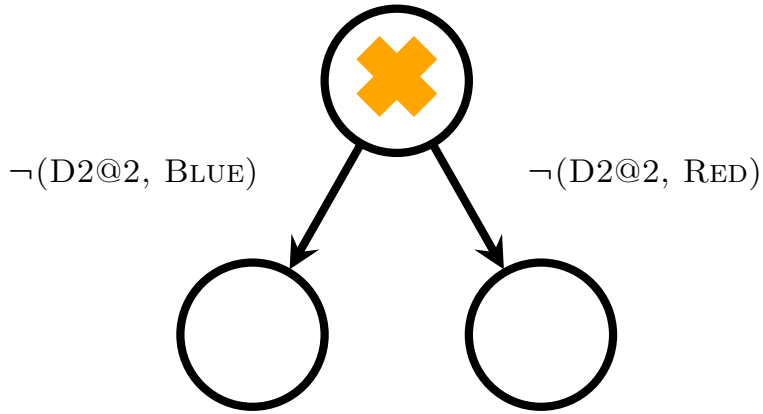
Conflict-based Search [Sharon et al, AIJ 2015]



CBS is an optimal branch-and-replan search algorithm.

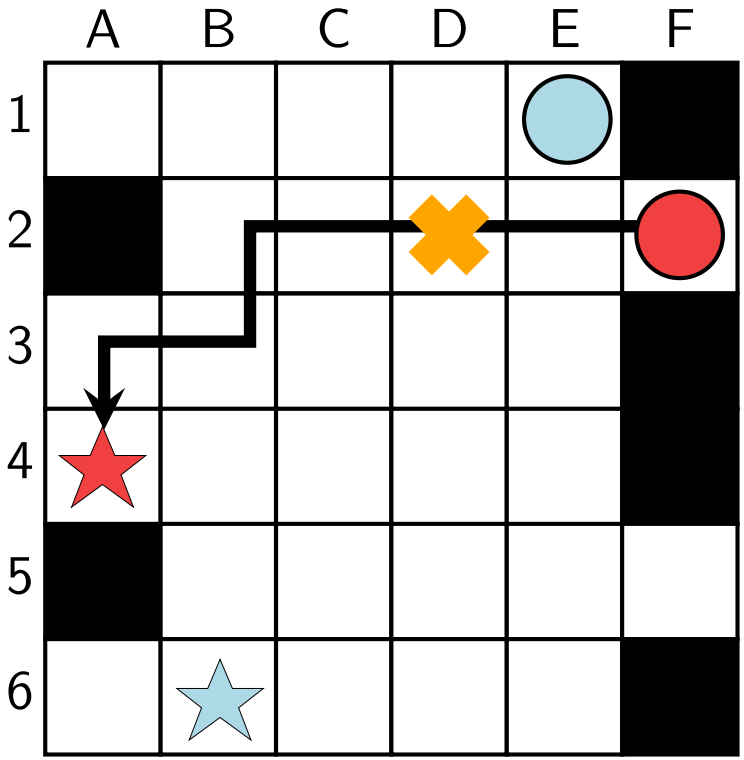
Conflict-based Search [Sharon et al, AIJ 2015]

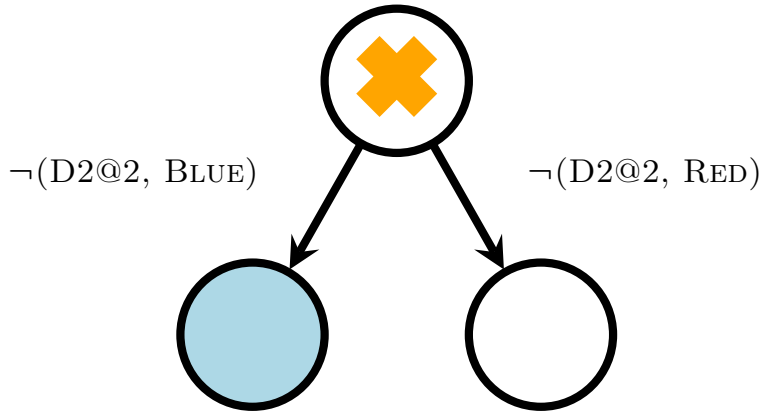




CBS is an optimal branch-and-replan search algorithm.

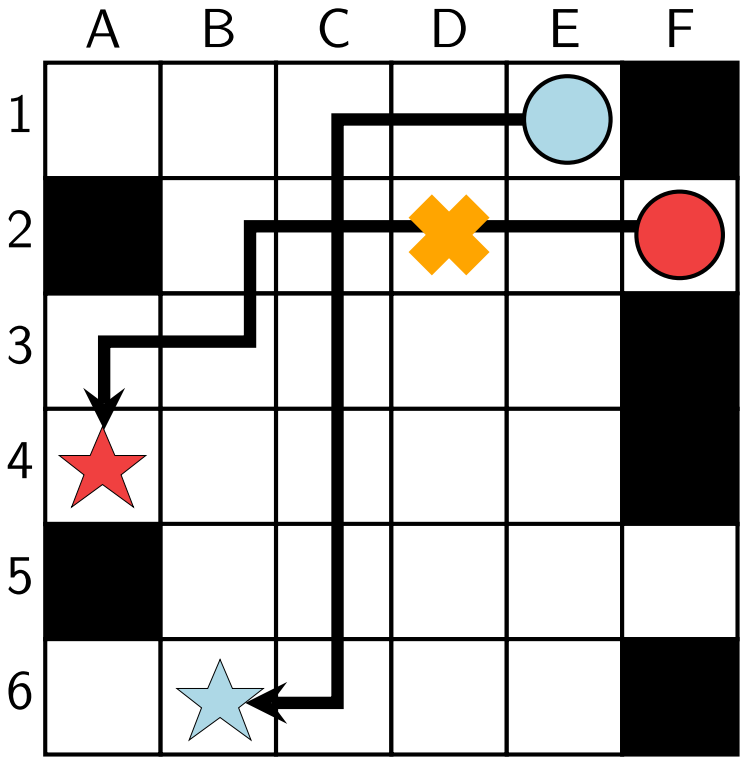
Conflict-based Search [Sharon et al, AIJ 2015]

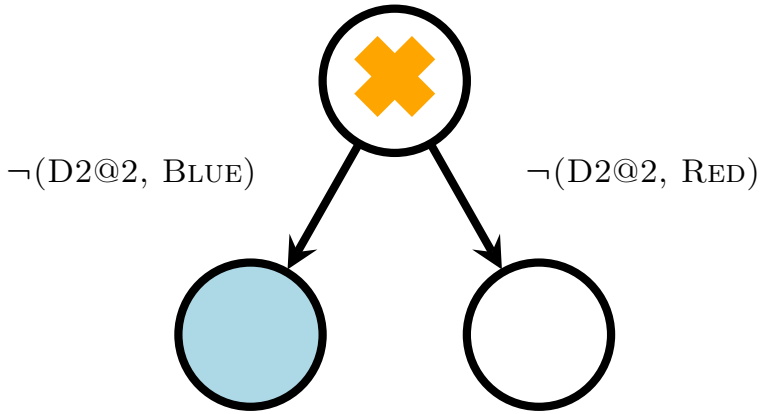




CBS is an optimal branch-and-replan search algorithm.

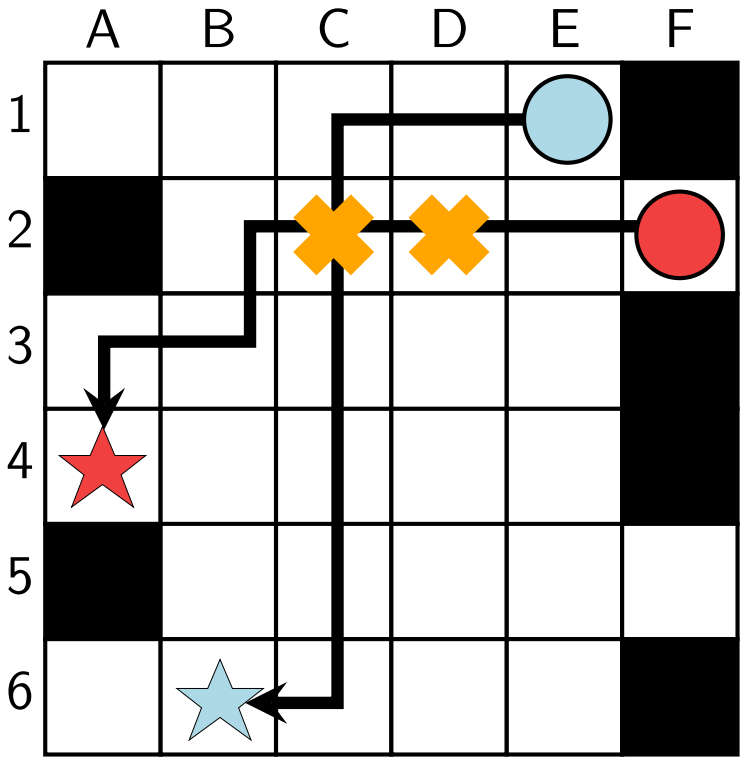
Conflict-based Search [Sharon et al, AIJ 2015]

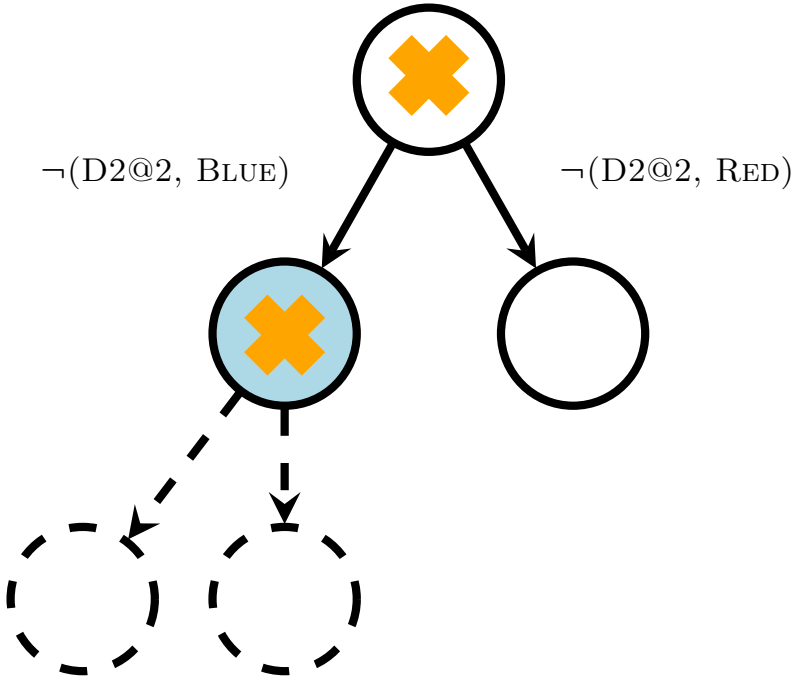




CBS is an optimal branch-and-replan search algorithm.

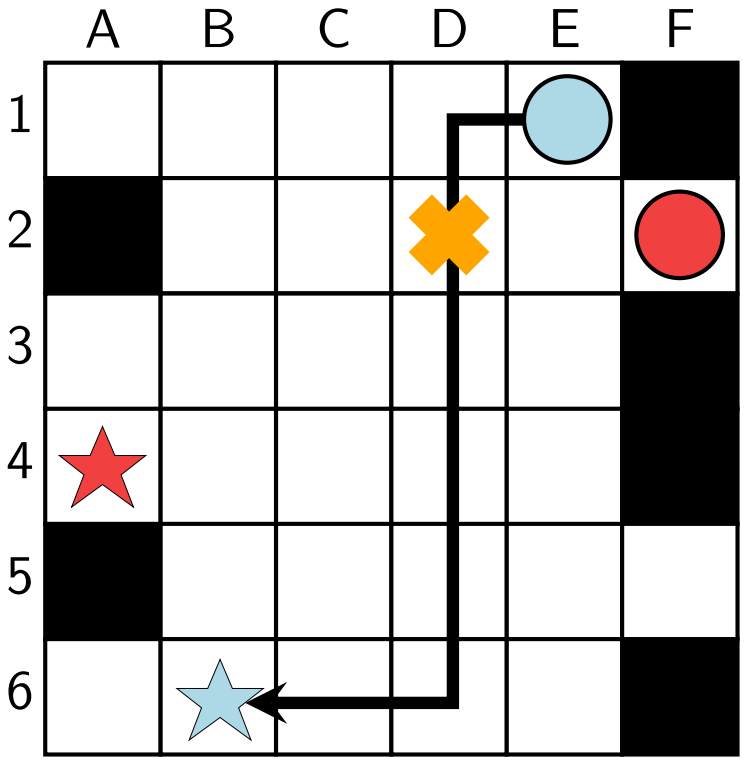
Conflict-based Search [Sharon et al, AIJ 2015]

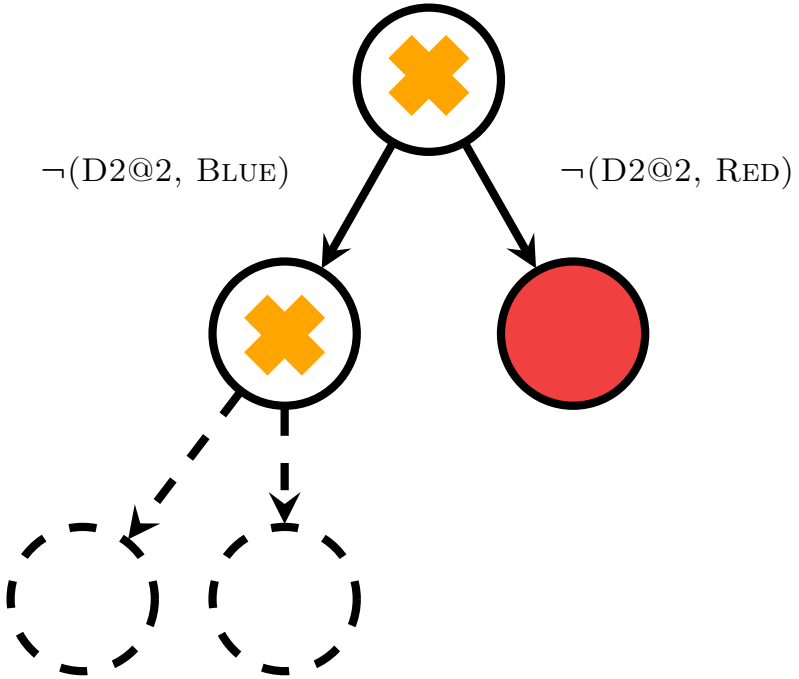




CBS is an optimal branch-and-replan search algorithm.

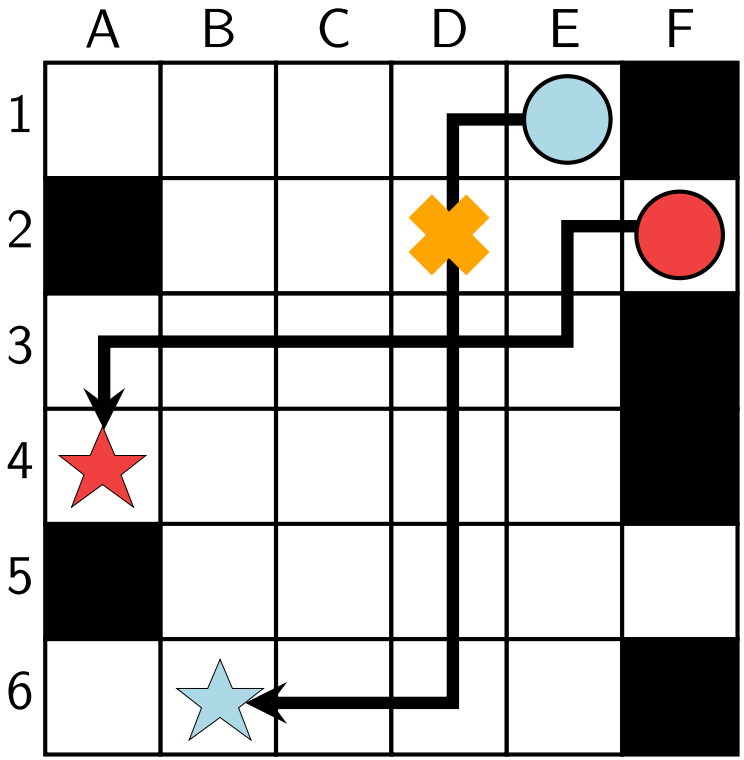
Conflict-based Search [Sharon et al, AIJ 2015]

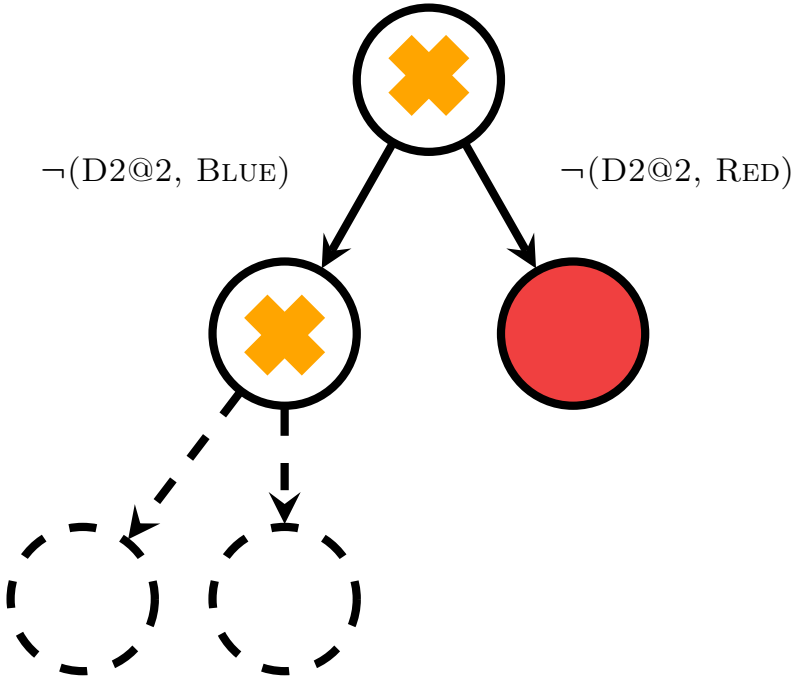




CBS is an optimal branch-and-replan search algorithm.

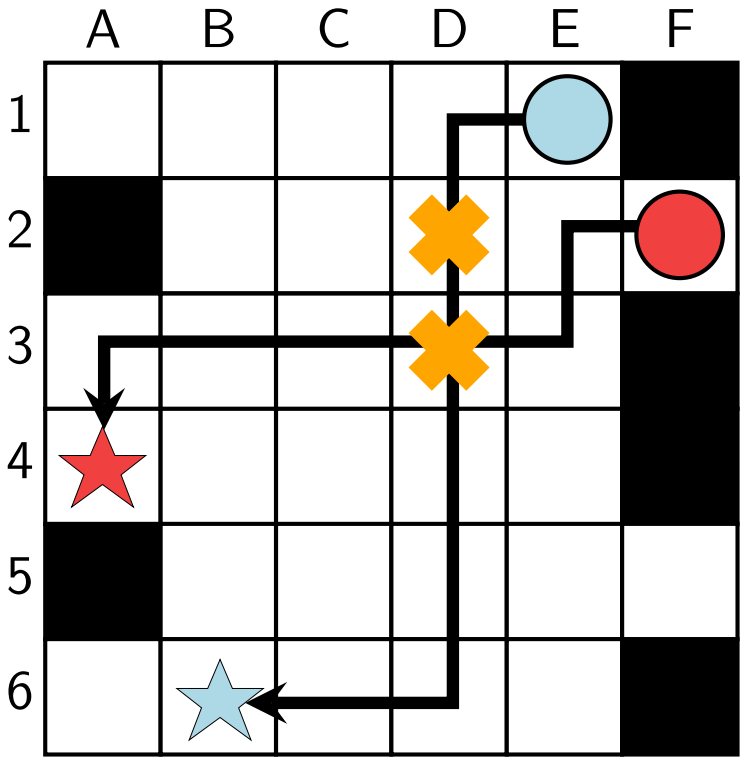
Conflict-based Search [Sharon et al, AIJ 2015]

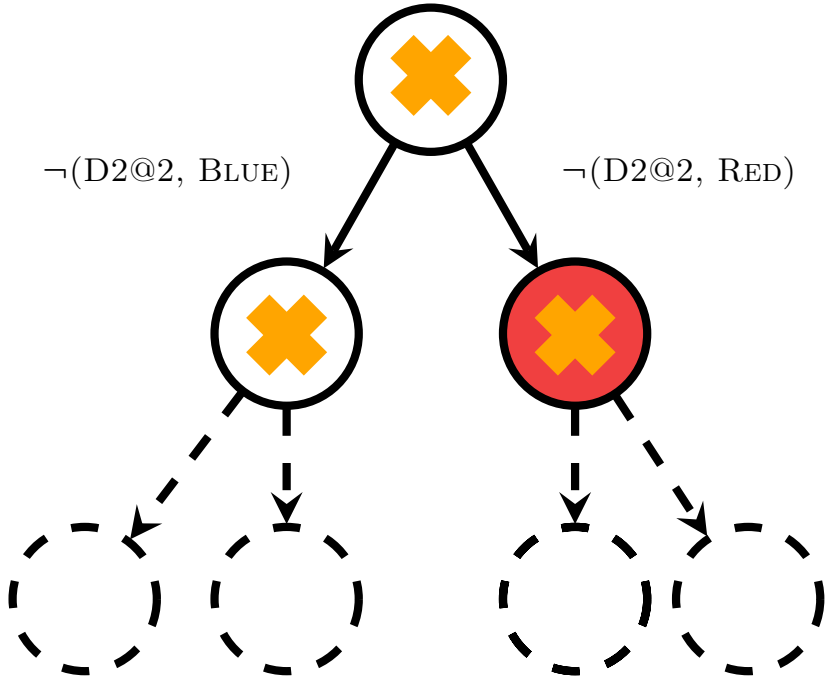




CBS is an optimal branch-and-replan search algorithm.

Conflict-based Search [Sharon et al, AIJ 2015]





CBS is an optimal branch-and-replan search algorithm.

Analysis of CBS

CBS is solution complete and optimal.

Optimality Sketch:

- For each valid solution S, there exists a node N on OPEN which **permits** S.

Analysis of CBS

CBS is solution complete and optimal.

Optimality Sketch:

- For each valid solution S, there exists a node N on OPEN which **permits** S.
  - Trivially true at the root node (no constraints)
  - Each split divides the set of valid solutions at the parent node.
  - Constraints eliminate valid solutions:
    from one subtree or the other but never from both.

Analysis of CBS

CBS is solution complete and optimal.

Optimality Sketch:

- For each valid solution S, there exists a node N on OPEN which **permits** S.
  - Trivially true at the root node (no constraints)
  - Each split divides the set of valid solutions at the parent node.
  - Constraints eliminate valid solutions:
    from one subtree or the other but never from both.
- Each node N is a minimum cost plan where the paths of the agents are consistent with the set of constraints at N

Analysis of CBS

CBS is solution complete and optimal.

Optimality Sketch:

- For each valid solution S, there exists a node N on $\textsc{Open}$ which **permits** S.
  - Trivially true at the root node (no constraints)
  - Each split divides the set of valid solutions at the parent node.
  - Constraints eliminate valid solutions:
    from one subtree or the other but never from both.
- Each node N is a minimum cost plan where the paths of the agents are consistent with the set of constraints at N
- The minimum f-value of any node $N \in \textsc{Open}$ is lower-bound on the minimum cost of any valid solution. □

Analysis of CBS

CBS is solution complete and optimal.

Completeness sketch:

- After some maximum timestep T no further collisions are possible (every agent arrives)
- The cost of every generated node is monotonically non-decreasing.
- The optimal solution cost $C \leq T$ $\square$

Complexity of MAPF

MAPF is intractable in a large number of cases. But not always!

Unlikely to find polynomial time algorithms for these problems:

- MAPF on general directed graphs
- MAPF on general undirected graphs
- MAPF on general grid graphs

MAPF complexity proofs are often very creative and they provide us with wonderful insights into the core difficulties. The following recent paper is a good start for those who want to know more: **(Nebel, B., 2020. On the Computational Complexity of Multi-Agent Pathfinding on Directed Graphs. In Proceedings of ICAPS.)**

Complexity of MAPF

MAPF is intractable in a large number of cases. But not always!

Cases which admit polynomial-time solutions:

- Bi-connected graphs
- Well-formed instances

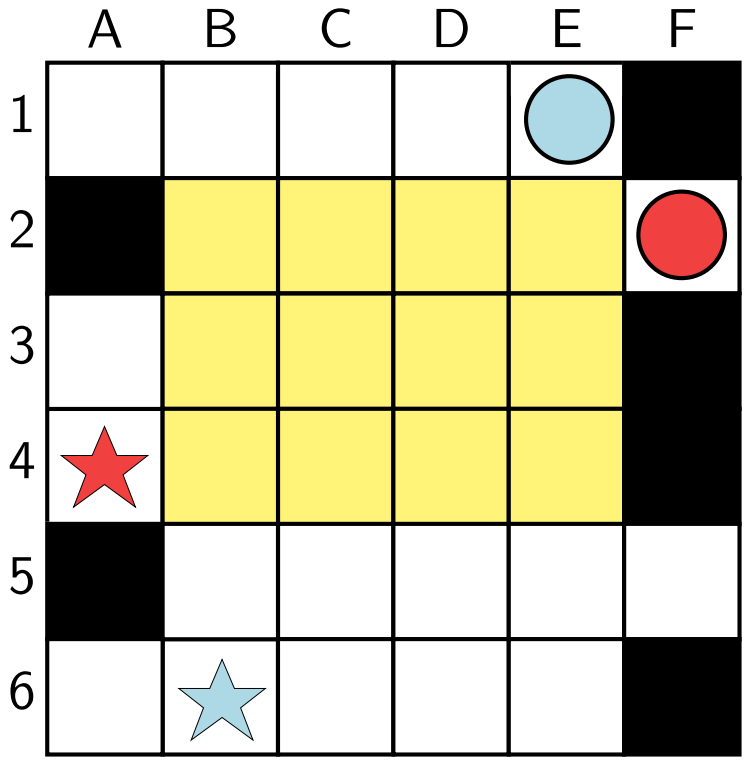
The class of well-formed MAPF problems is proving especially important for practical settings. The following paper is a good starting point for this important topic: **(Cap, M., Vokrinek, J. and Kleiner, A. 2015. Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures. In Proceedings of ICAPS)**

Multi-Agent Path Finding: Constraints

Bojie Shen
Monash University

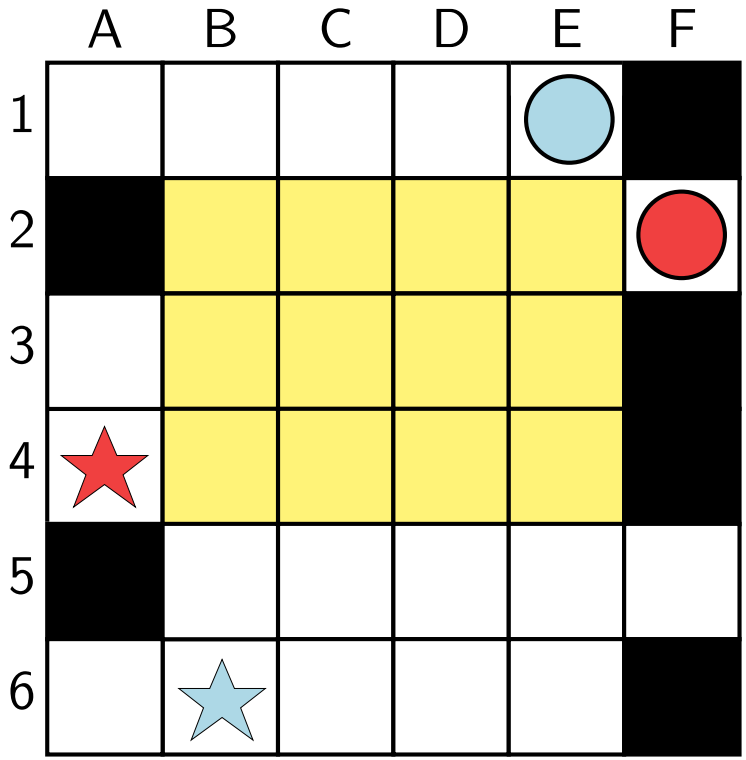
https://bojie-shen.com/

Rectangle Symmetry



Each agent has many equivalent paths through the yellow area. Solving this problem requires proving that one agent must wait.

Rectangle Symmetry



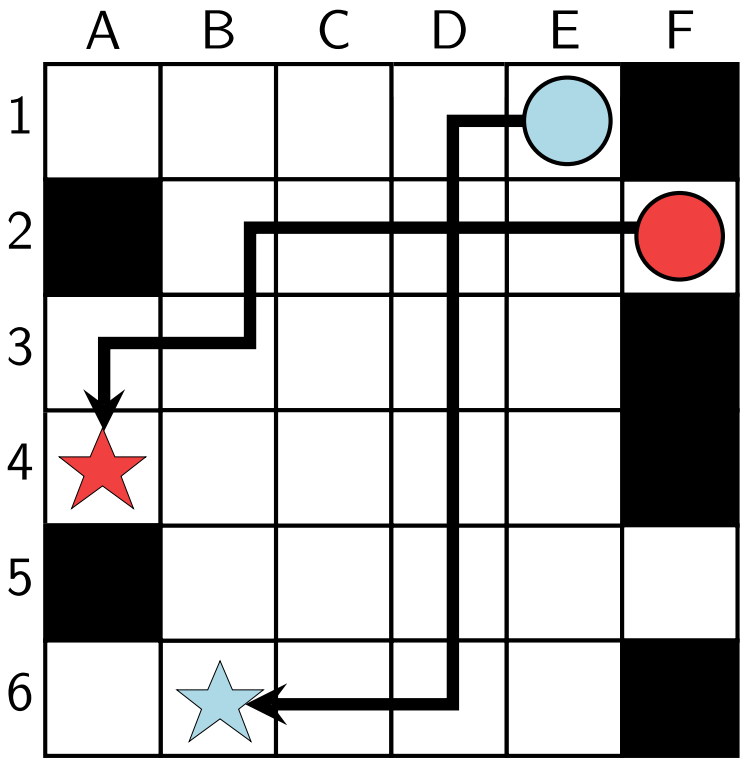
R	1	2	3	4	5	6	7	8	9
1	1	1	2	3	4	5	6	7	8
2		3	7	14	26	46	79	133	221
3			22	53	116	239	472	904	1,692
4				142	392	1,016	2,651	6,828	17,747
5					1,015	2,971	8,525	23,733	65,236
6						7,447	24,275	78,002	254,173
7							62,429	222,524	795,197
8								573,004	**>1,518,151**

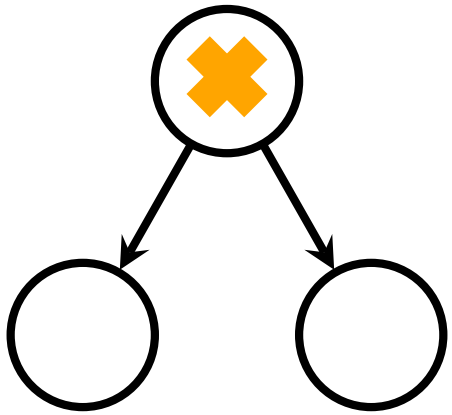
Table:
Nodes expanded by CBS to solve a 2-agent problem with a $n \times m$ rectangle.

Each agent has many equivalent paths through the yellow area. Solving this problem requires proving that one agent must wait.

Barrier Constraints

Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, Sven Koenig. **Symmetry Breaking Constraints for Grid-based Multi-Agent Path Finding**. AAAI, 2019.

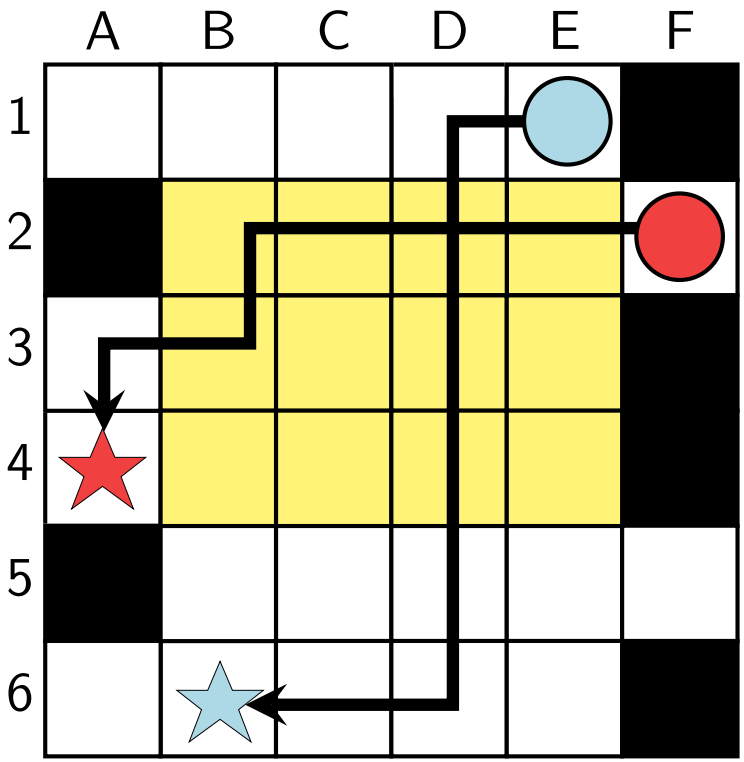


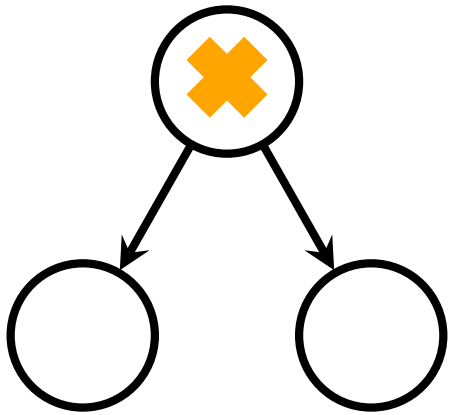


Barriers constraints are a specialised reasoning technique that *break* rectangle symmetries and help CBS prove optimality sooner.

Barrier Constraints

Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, Sven Koenig. **Symmetry Breaking Constraints for Grid-based Multi-Agent Path Finding**. AAAI, 2019.

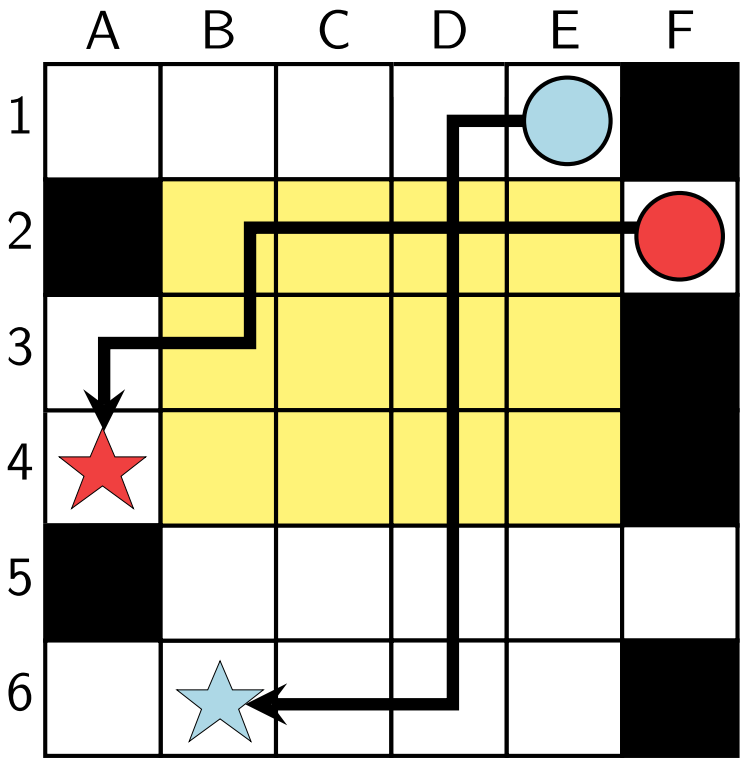


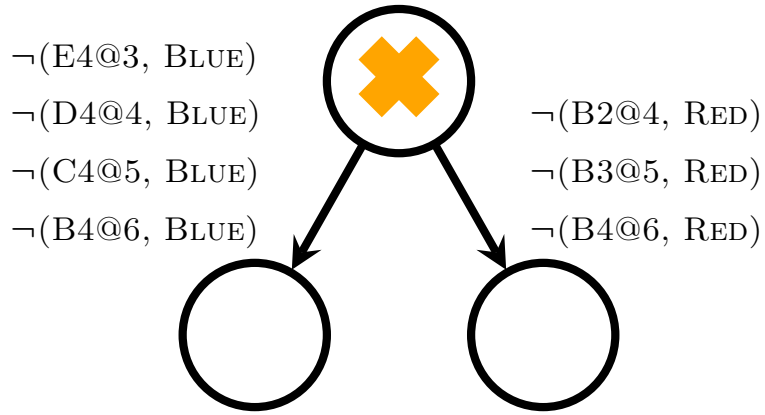


We modify CBS to look for conflicts which have rectangle symmetries.

Barrier Constraints

Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, Sven Koenig. **Symmetry Breaking Constraints for Grid-based Multi-Agent Path Finding**. AAAI, 2019.

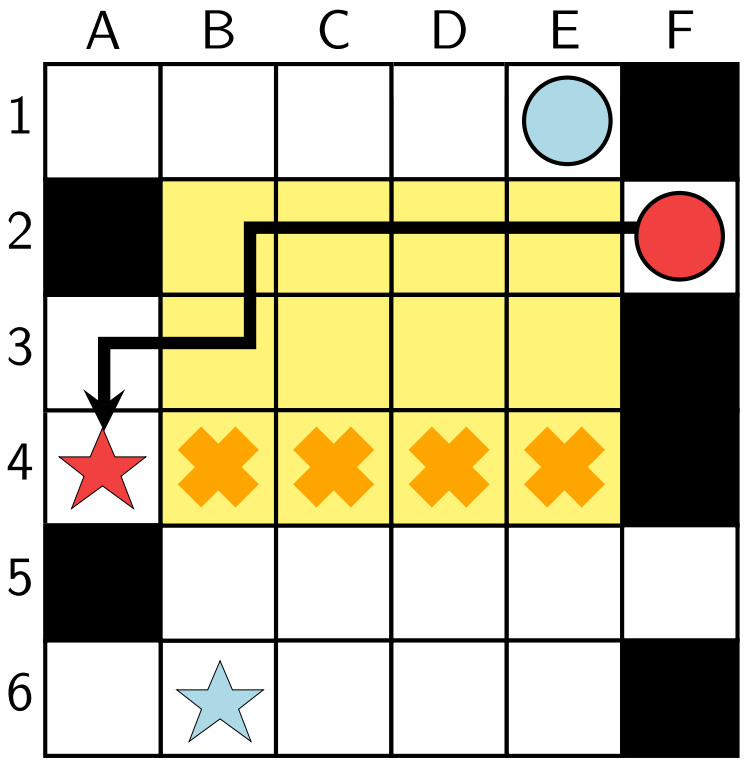


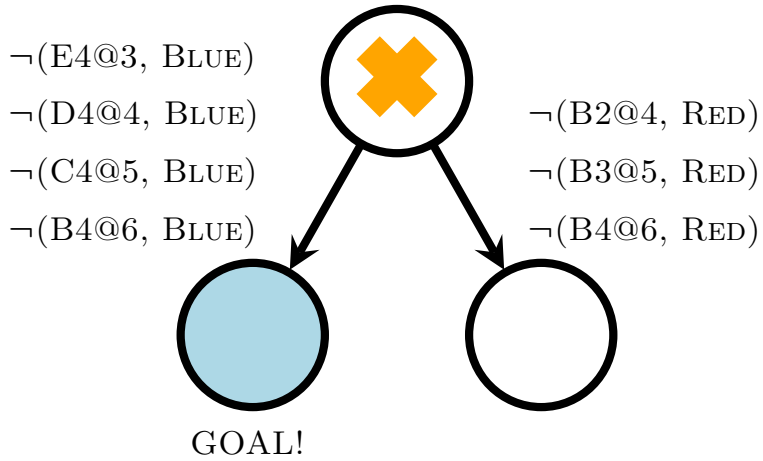


When we detect a conflict with rectangle symmetries we post a set of constraints that force agents to *wait* or find a *detour*.

Barrier Constraints

Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, Sven Koenig. **Symmetry Breaking Constraints for Grid-based Multi-Agent Path Finding**. AAAI, 2019.

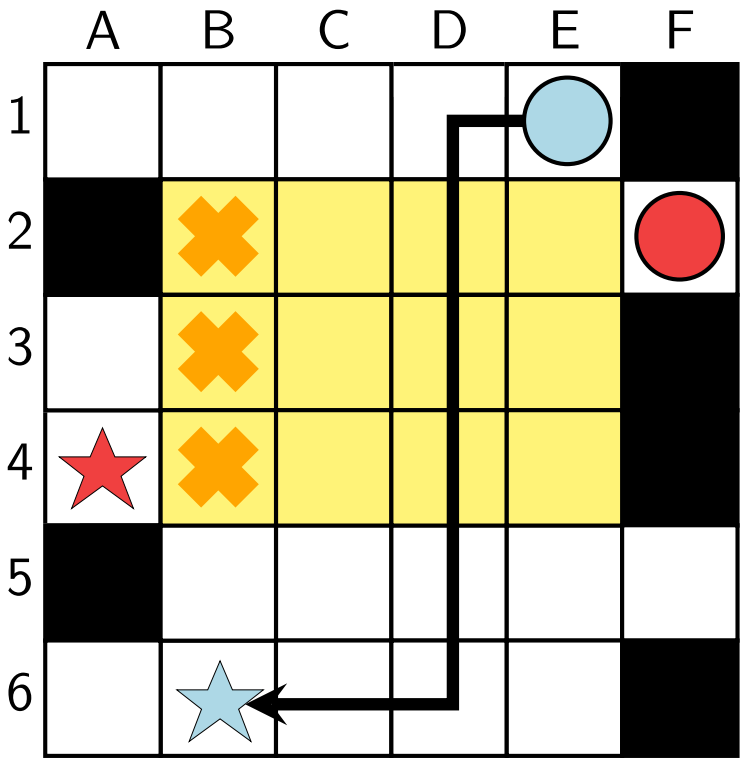


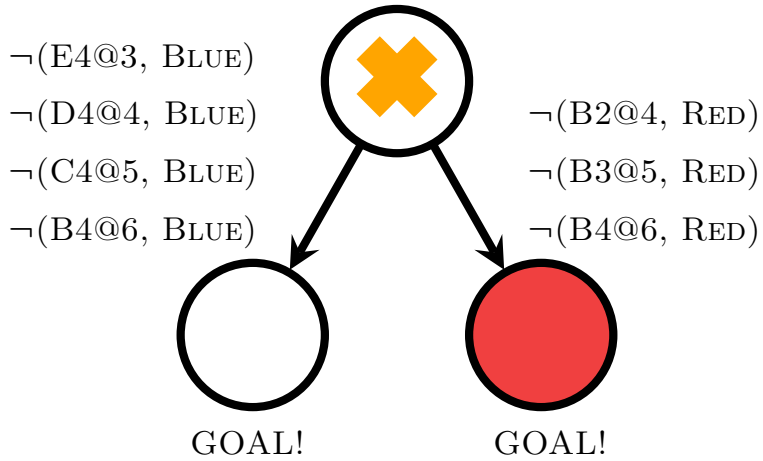


We replan Blue. Here, the barrier forces the agent to wait. Blue can wait anywhere but we prefer a path that avoids Red.

Barrier Constraints

Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, Sven Koenig. **Symmetry Breaking Constraints for Grid-based Multi-Agent Path Finding**. AAAI, 2019.





We replan Red. Again, the barrier forces the agent to wait. We can choose any optimal path but prefer one that avoids Blue.

Cardinal Rectangle Reasoning (CR)

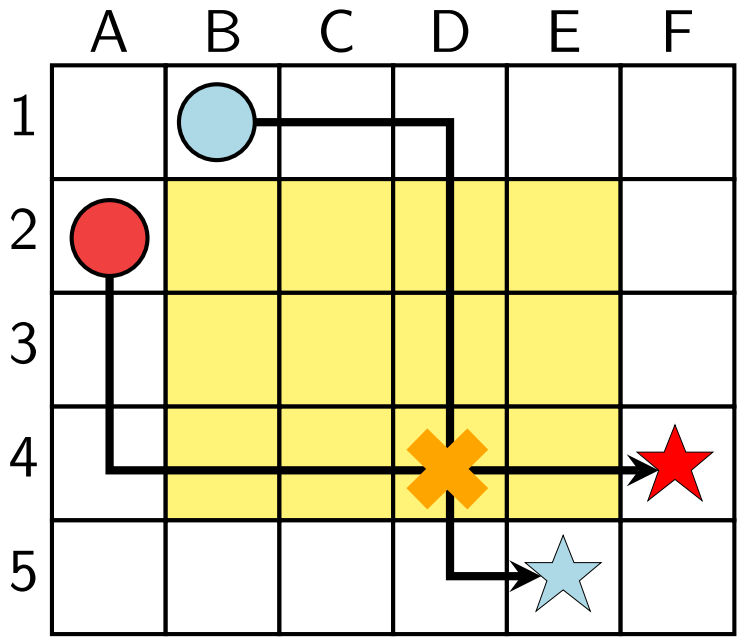
Each rectangle so far satisfies the following conditions:

(a) Both agents have a Manhattan optimal path, from start to target.

(b) Both agents reach every tile inside the rectangle area, at the same optimal time.

(c) All optimal paths pass through the rectangle area and are in collision.

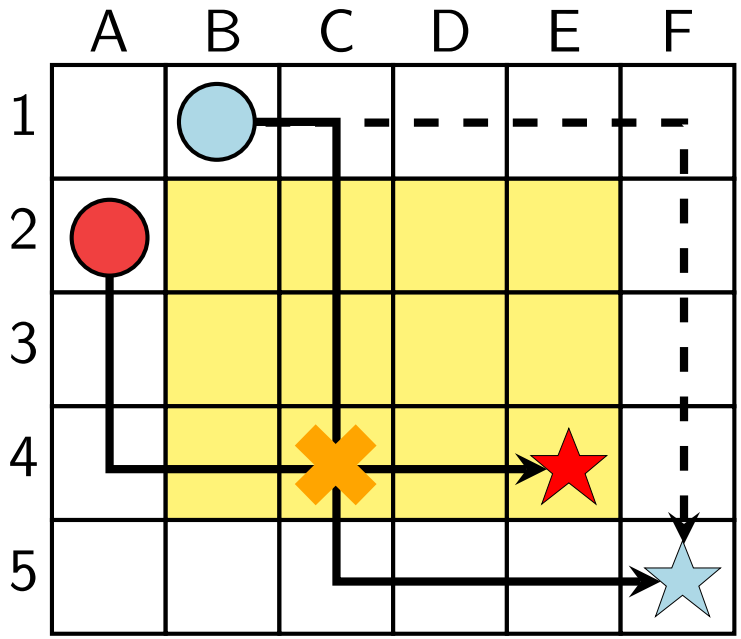
Cardinal Rectangle Reasoning (CR)



Resolving such *cardinal rectangles* increases solution cost by 1.

Rectangle Reasoning for Paths (R)

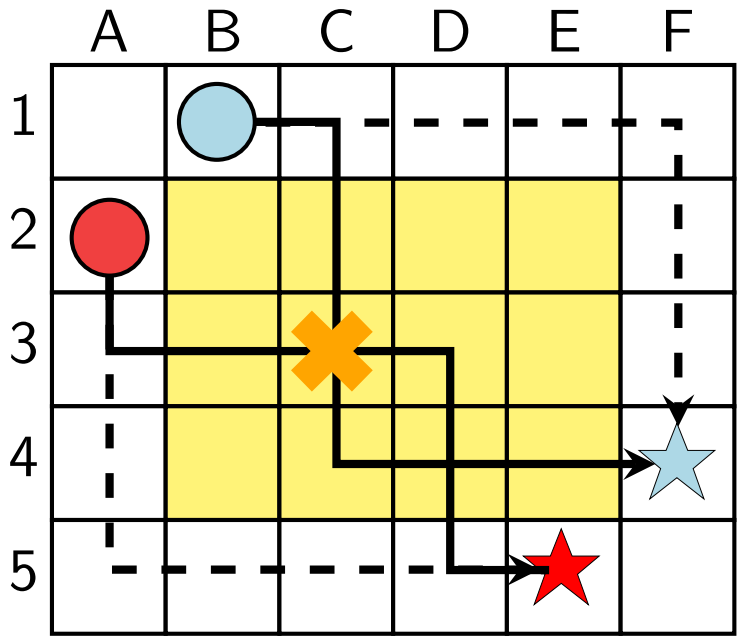
Relaxing the three conditions allows us to detect other types of rectangle symmetries.



Resolving this *semi-cardinal* rectangle raises the cost for only one agent.

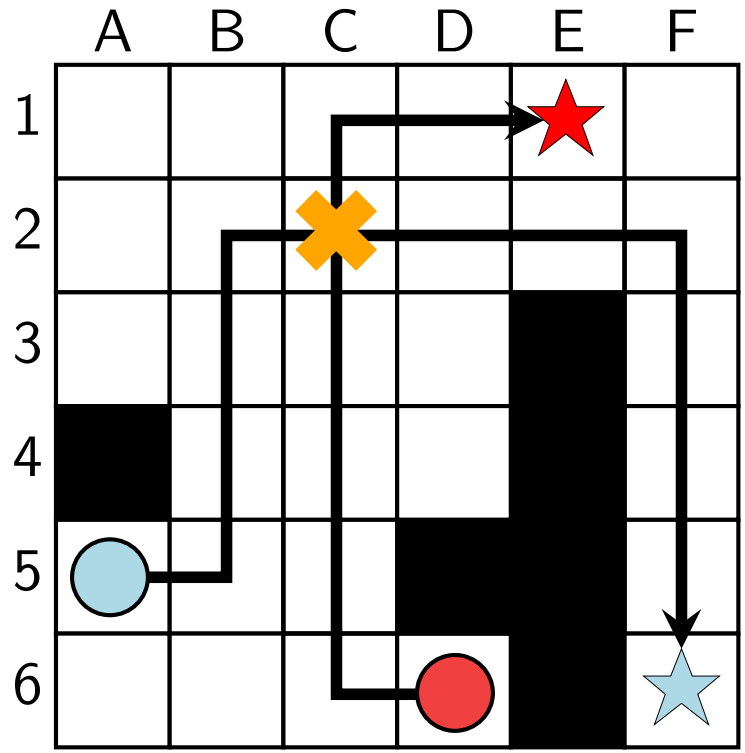
Rectangle Reasoning for Paths (R)

Relaxing the three conditions allows us to detect other types of rectangle symmetries.



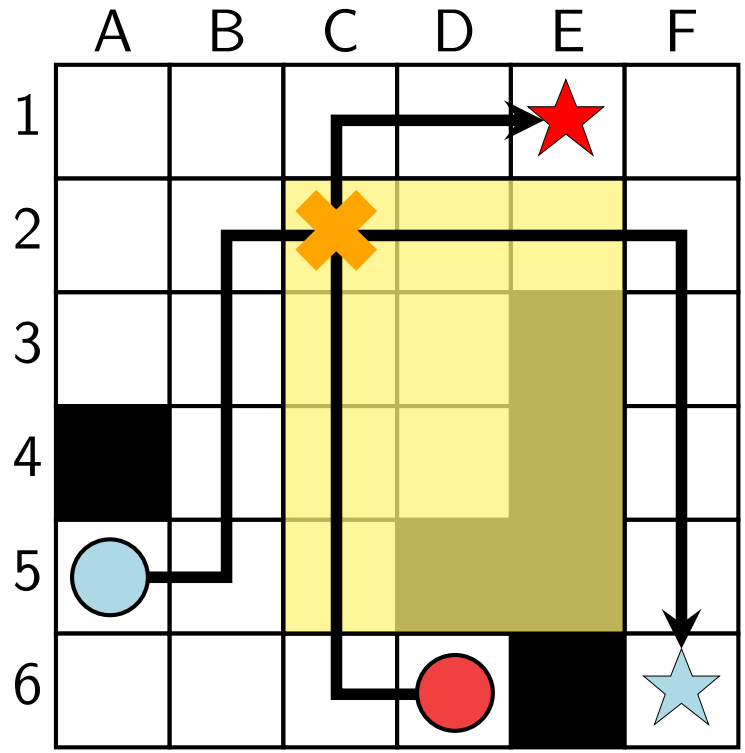
Resolving this *non–cardinal* rectangle eliminates some optimal paths but does not raise any costs.

Rectangle Reasoning for Path Segments (RM)



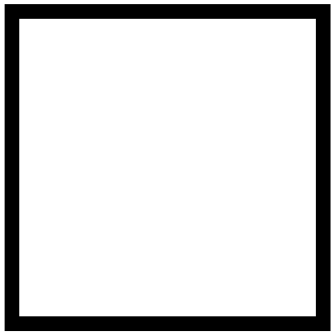
Neither of these agents has a Manhattan optimal path to their target.

Rectangle Reasoning for Path Segments (RM)

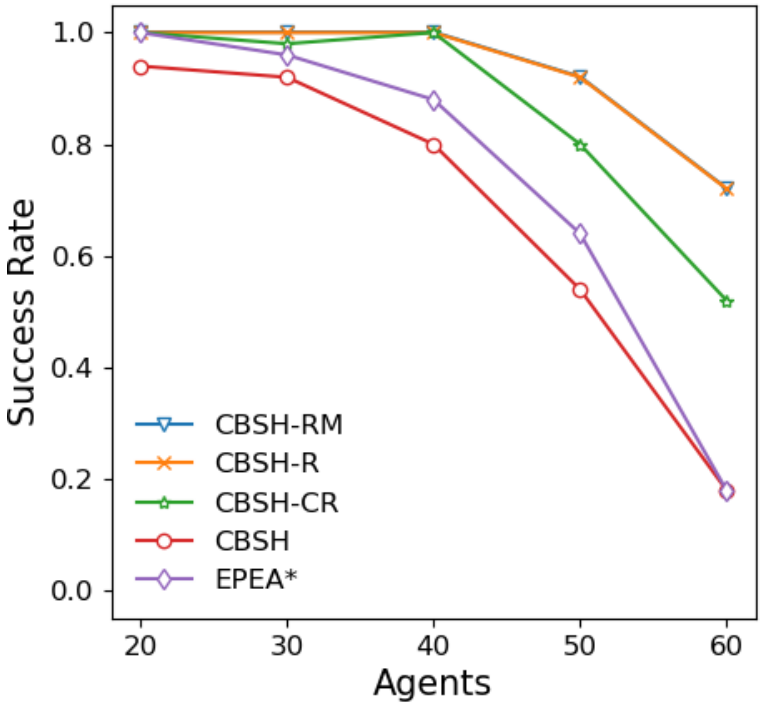


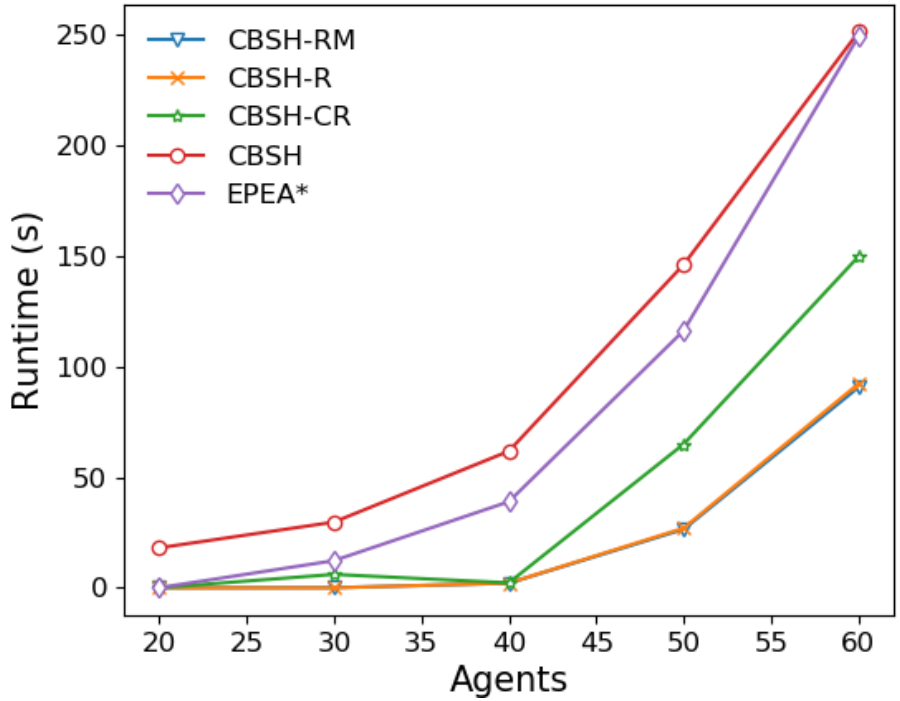
Rectangle symmetries exist, but only for Manhattan optimal subpaths.

Experiments on Small Grids



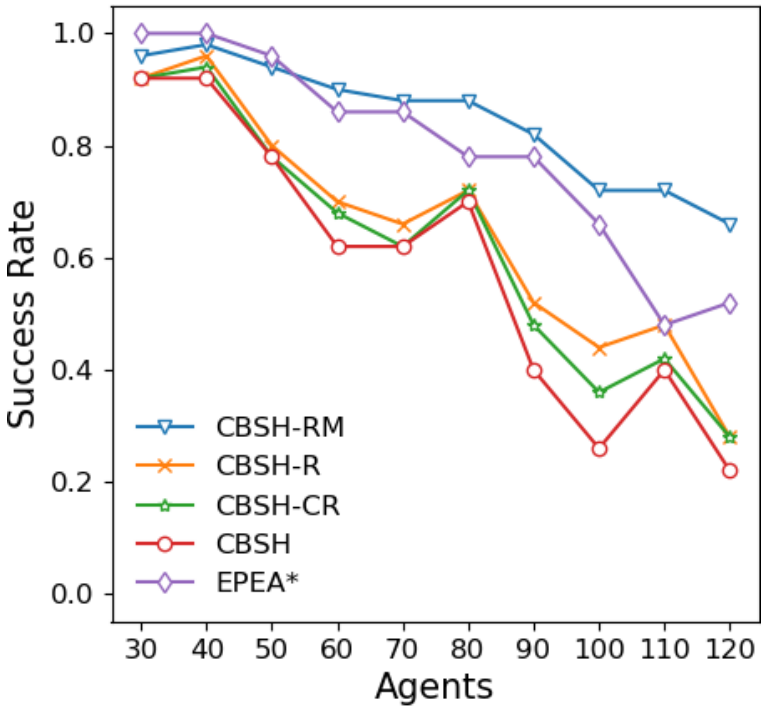
- Random start-target locations
- 50 instances per problem size (# agents).
- Sum-of-Costs objective
- *20x20 empty grid*

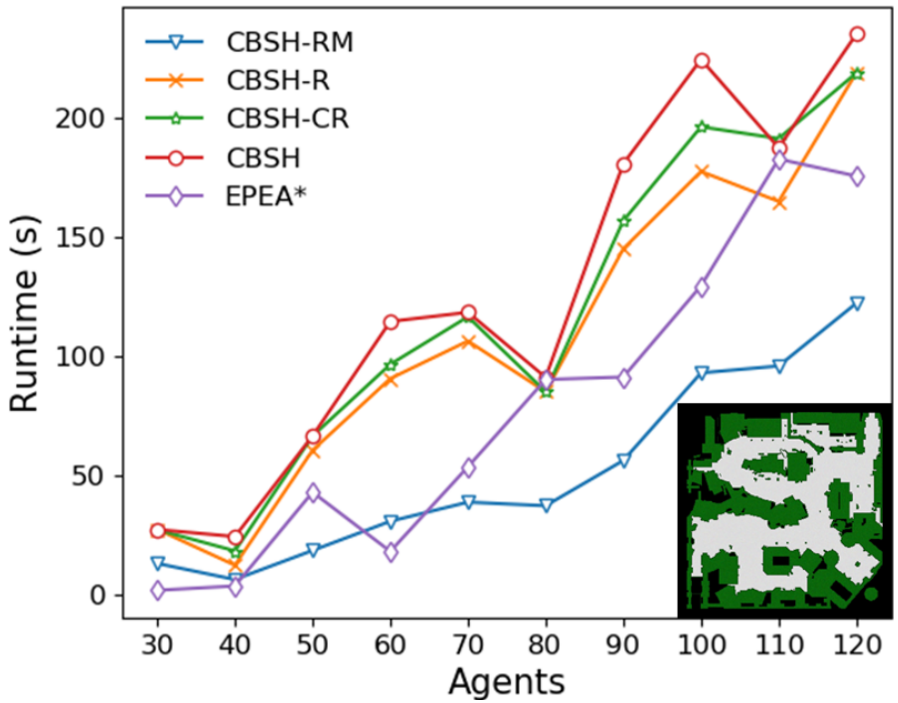




Experiments on Large Grids

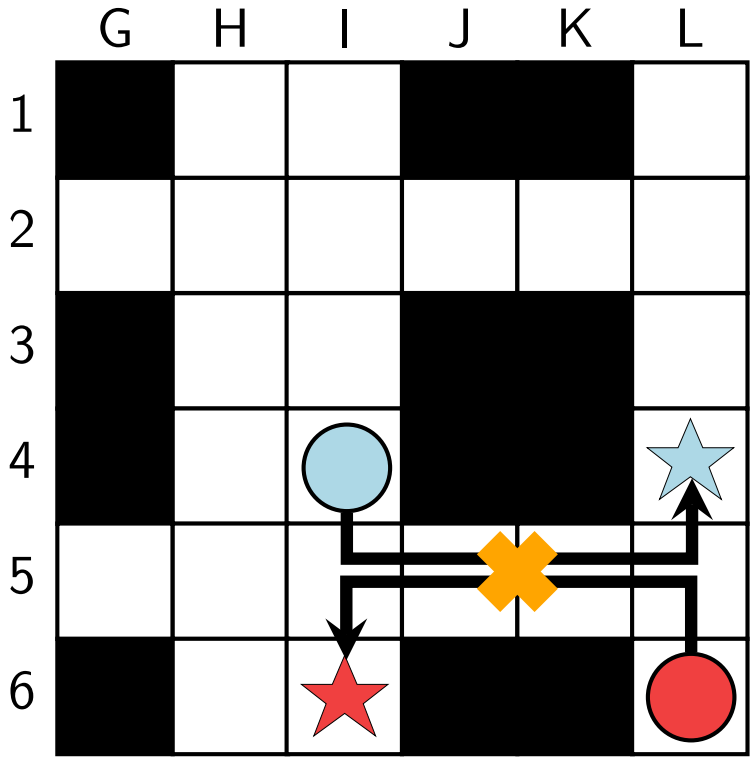
- Random start-target locations
- 50 instances per problem size (# agents).
- Sum-of-Costs objective
- *den520d:* 194 $\times$ 194 game grid.





Corridor Symmetry

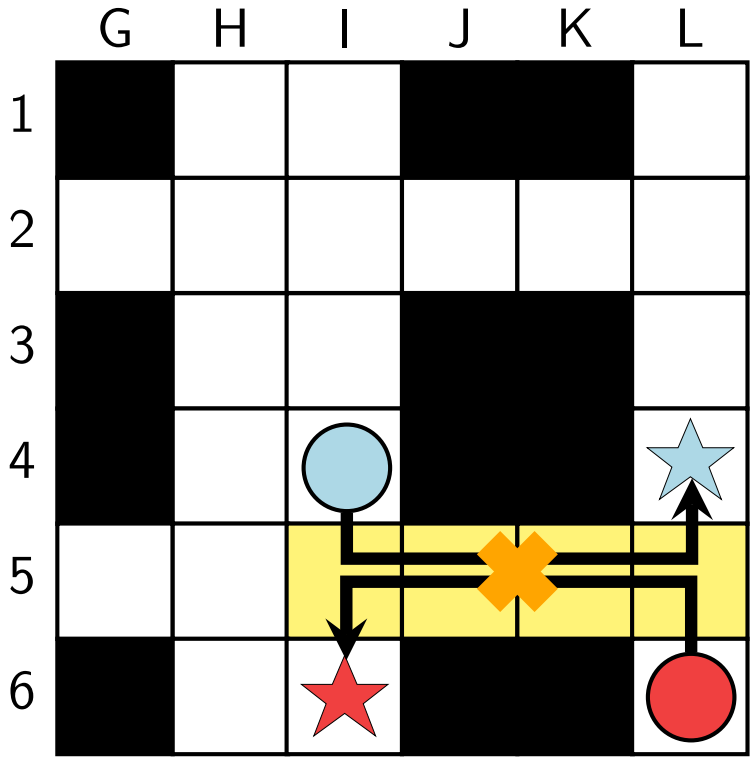
Li, J., Gange, G., Harabor, D., Stuckey, P.J., Ma, H. and Koenig, S., 2020. **New techniques for pairwise symmetry breaking in multi-agent path finding**. In Proceedings of ICAPS.



Corridor symmetries arise when replanning agents after a head-to-head collision in a congested area.

Corridor Symmetry

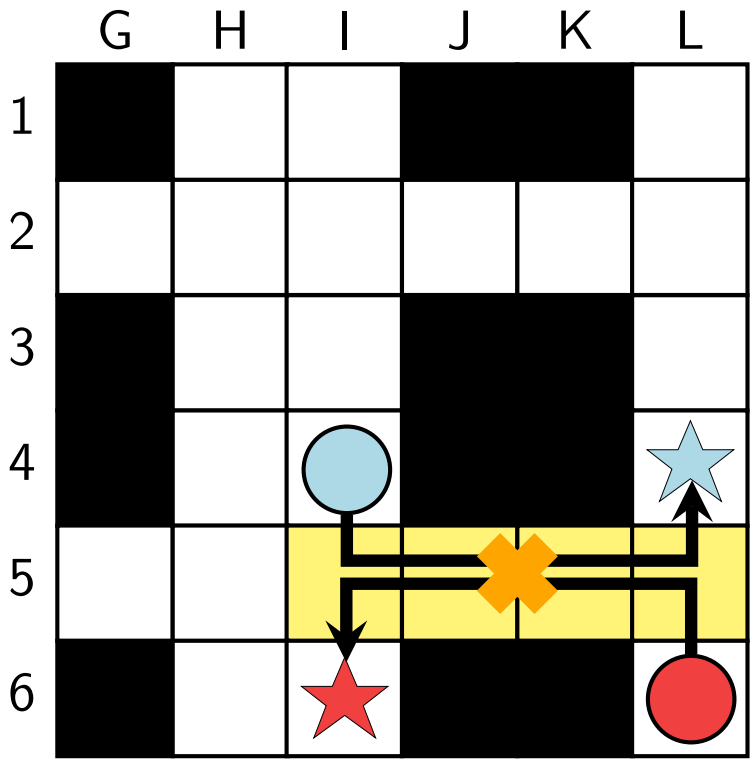
Li, J., Gange, G., Harabor, D., Stuckey, P.J., Ma, H. and Koenig, S., 2020. **New techniques for pairwise symmetry breaking in multi-agent path finding**. In Proceedings of ICAPS.

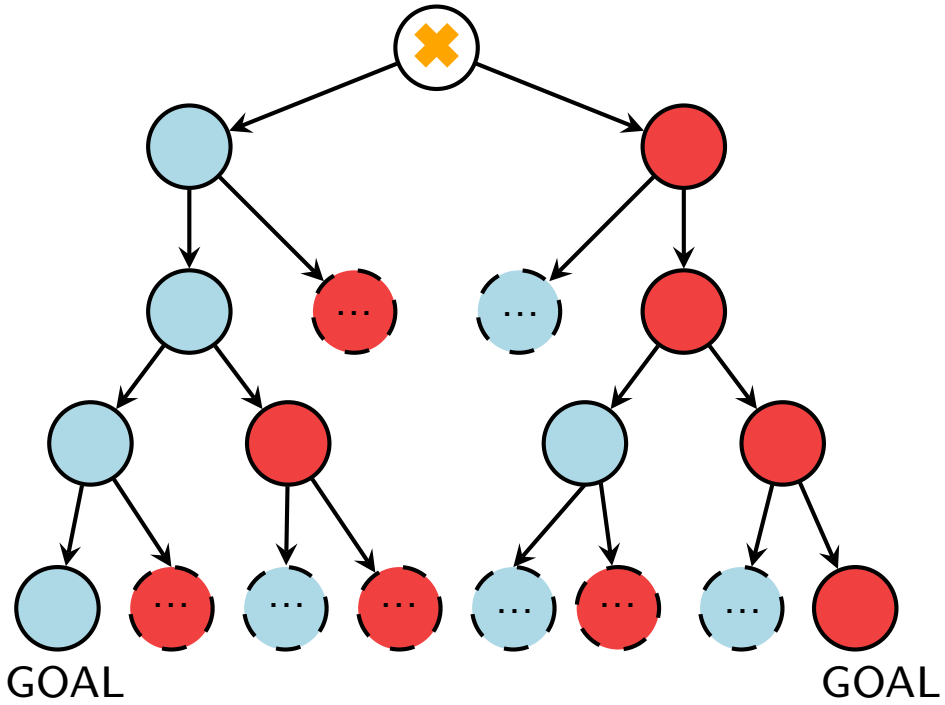


The optimal strategy is for one of the agents to postpone its corridor traversal (or to find an optimal-cost detour).

Corridor Symmetry

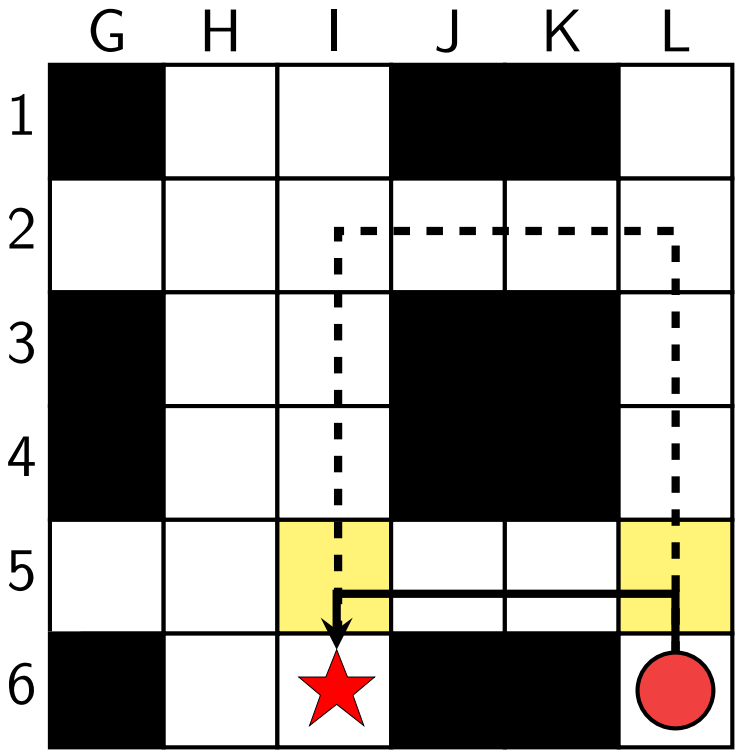
Li, J., Gange, G., Harabor, D., Stuckey, P.J., Ma, H. and Koenig, S., 2020. **New techniques for pairwise symmetry breaking in multi-agent path finding**. In Proceedings of ICAPS.





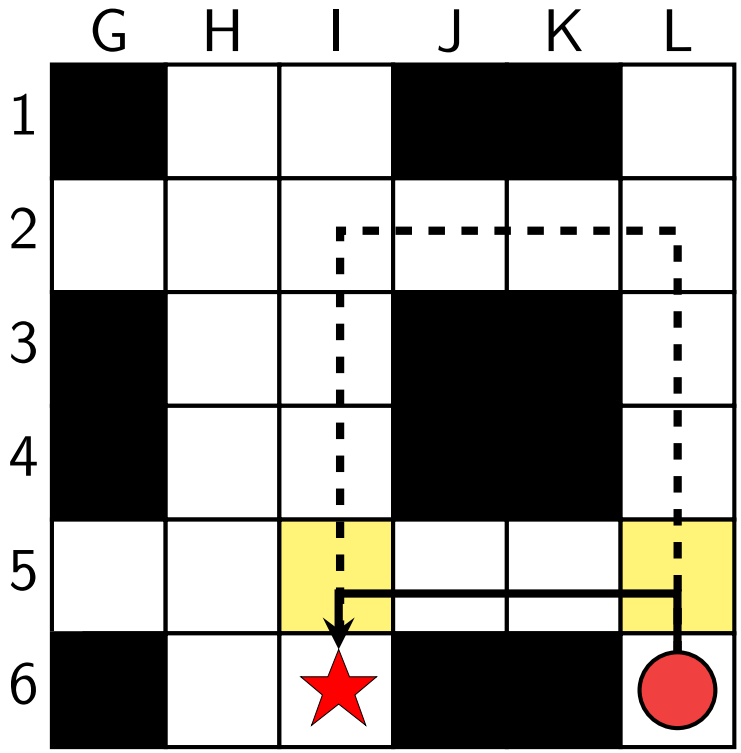
When corridor symmetries arise, they can be very difficult to to resolve without specialised reasoning.

Resolving Corridor Symmetries



Here, red could wait for blue or bypass the corridor.

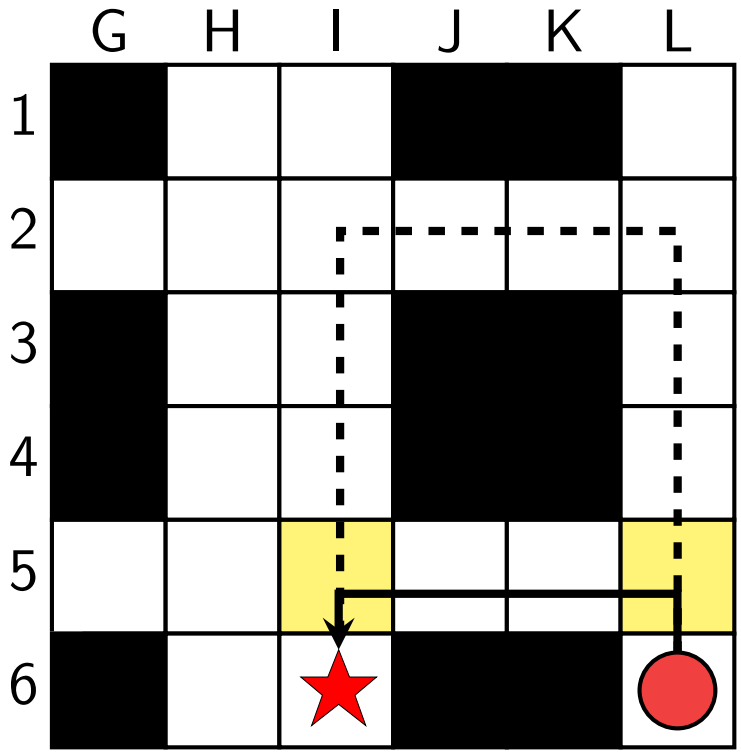
Resolving Corridor Symmetries



Wait Option

- The corridor has length $k = 3$
- Blue exits corridor at location L5 at time $t_2 = 4$.
- Red exits corridor at location I5 at time $t_1 = t_2 + 1 + k = 8$

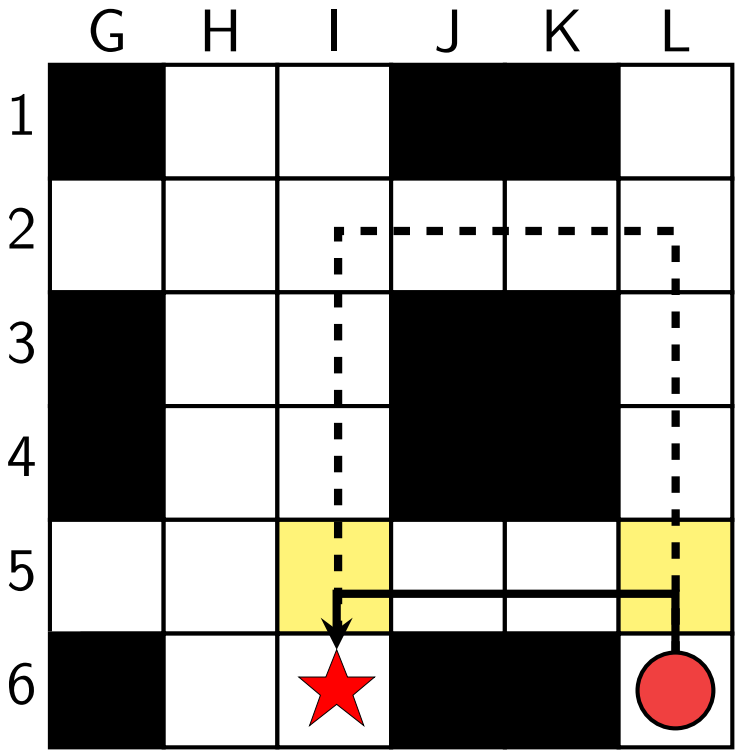
Resolving Corridor Symmetries



Bypass Option

- Red reaches location I5 at time $t_1' = 10$

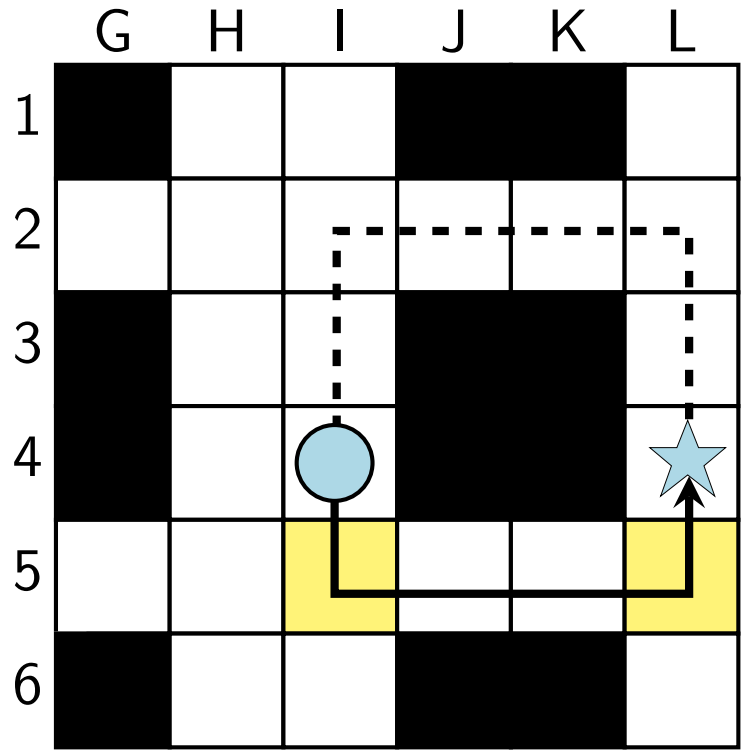
Resolving Corridor Symmetries



Waiting looks better, so we add a new constraint:

$$\langle \mathrm{red}, \mathtt{I5}, [0, \mathbf{min}\{8, 10\}] \rangle$$

Resolving Corridor Symmetries

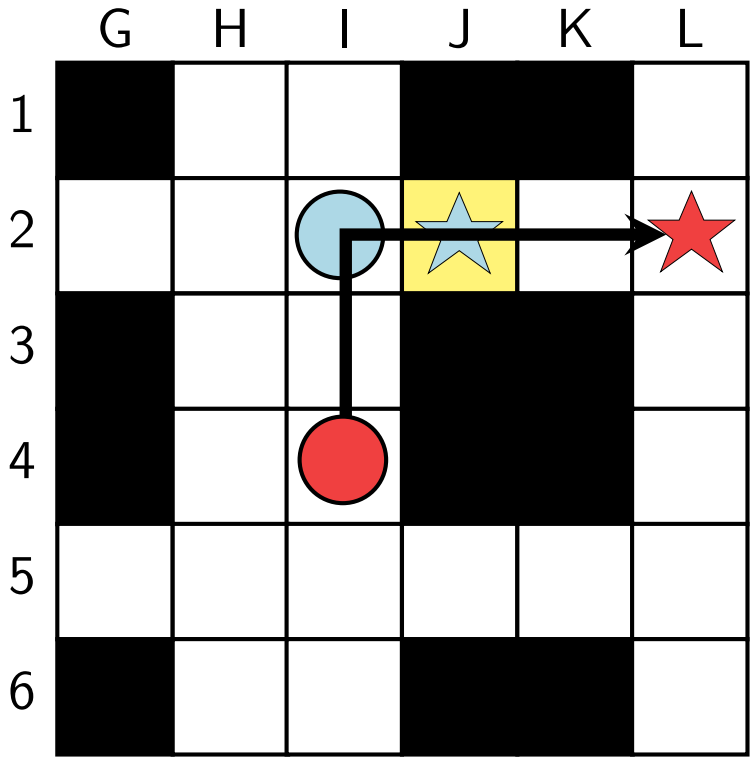


When replanning blue, the bypass option looks better:

$\langle \text{blue}, \texttt{L5}, [0, \textbf{min}\{9, 7\}] \rangle$

Target Symmetry

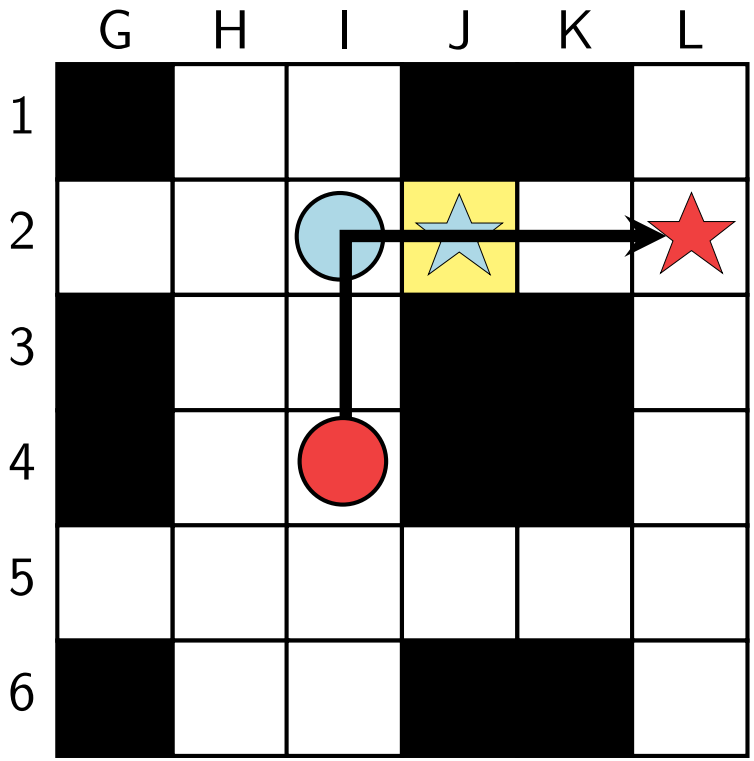
Li, J., Gange, G., Harabor, D., Stuckey, P.J., Ma, H. and Koenig, S., 2020. **New techniques for pairwise symmetry breaking in multi-agent path finding**. In Proceedings of ICAPS.



Target symmetries arise when one agent arrives at its target but later needs to make room for another agent.

Target Symmetry

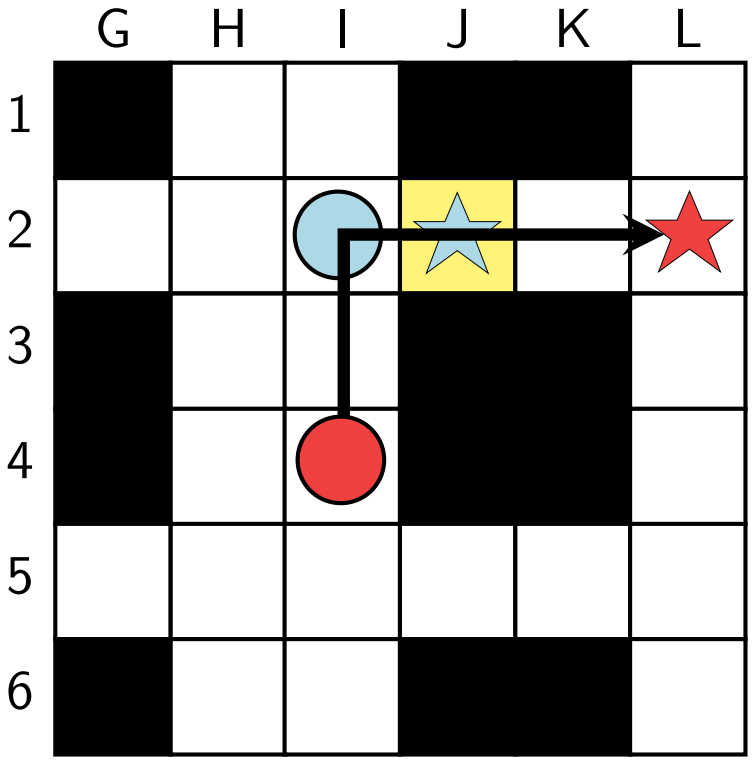
Li, J., Gange, G., Harabor, D., Stuckey, P.J., Ma, H. and Koenig, S., 2020. **New techniques for pairwise symmetry breaking in multi-agent path finding**. In Proceedings of ICAPS.

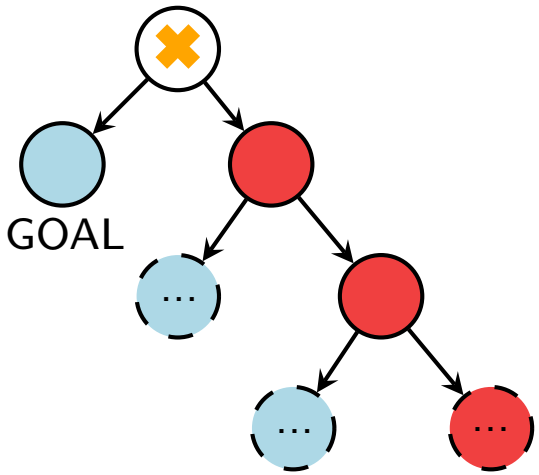


The optimal strategy is for the first agent to avoid waiting at its target (or for the second to find an optimal-cost detour).

Target Symmetry

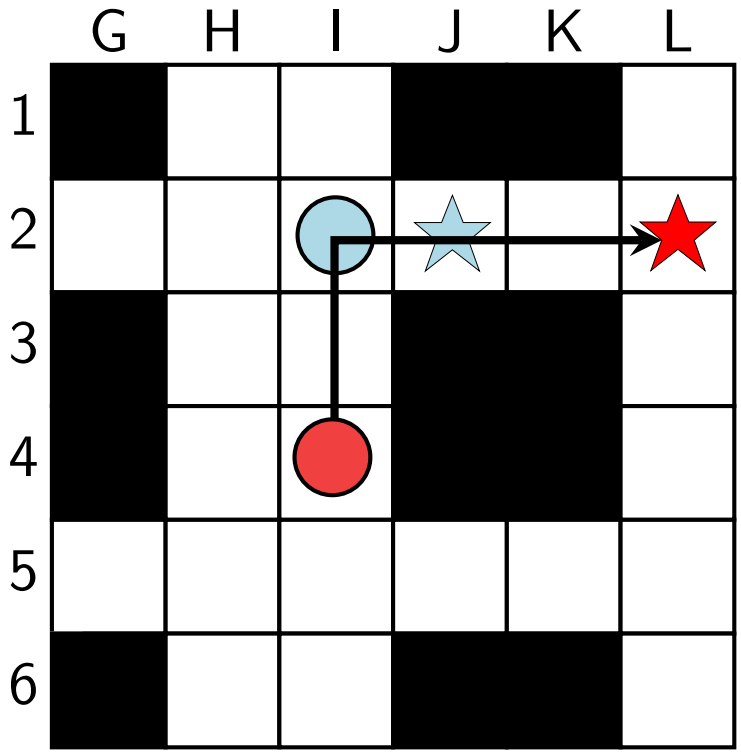
Li, J., Gange, G., Harabor, D., Stuckey, P.J., Ma, H. and Koenig, S., 2020. **New techniques for pairwise symmetry breaking in multi-agent path finding**. In Proceedings of ICAPS.





When target symmetries arise, they also can be difficult to resolve without specialised reasoning.

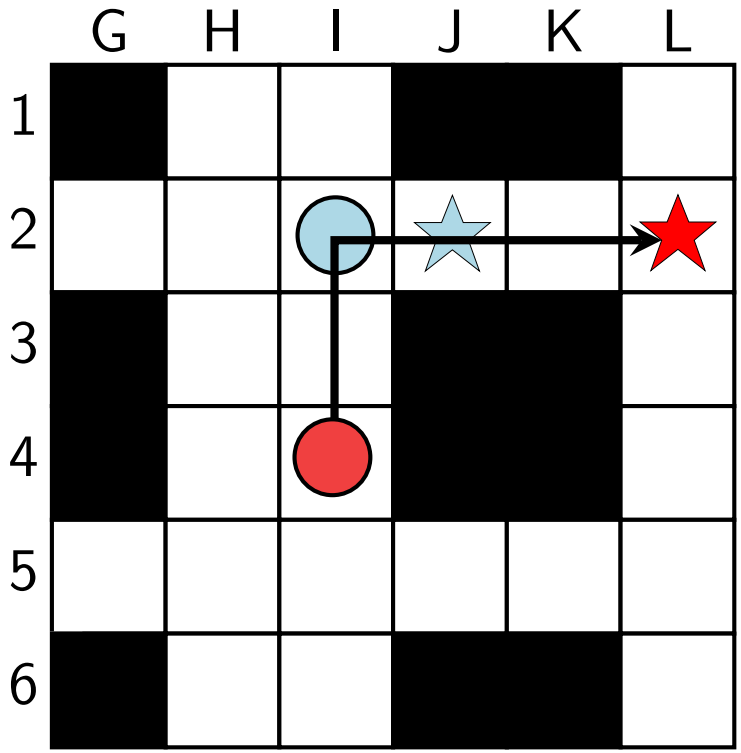
Resolving Target Symmetries



To resolve the conflict we *split only on blue*.

There are two choices: blue gets priority or blue waits.

Resolving Target Symmetries

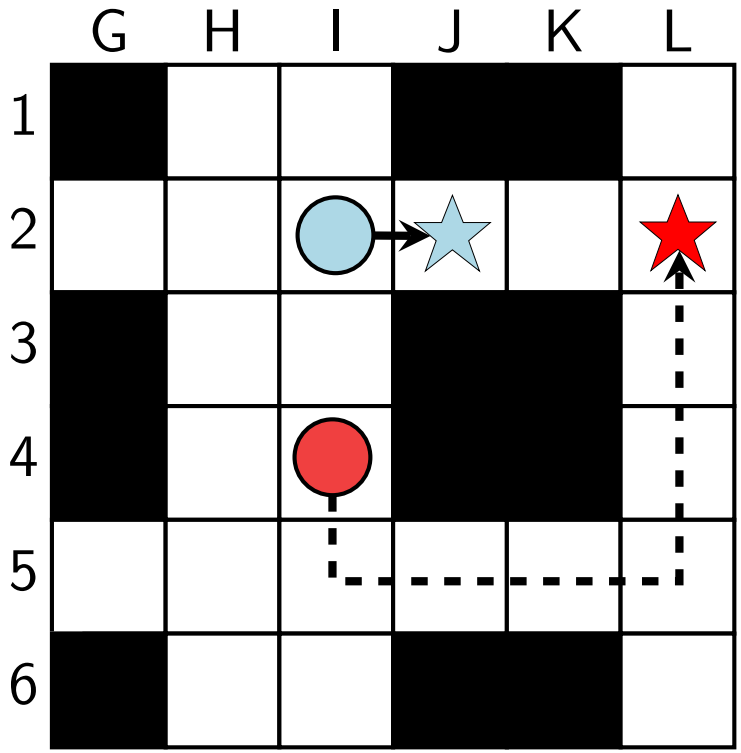


Blue Waits

We add a constraint on the length of the path for blue: $l_{\text{blue}} > 3$

Blue finds a new conflict-free path: $\langle$`I2, I1, I1, I2, J2`$\rangle$

Resolving Target Symmetries

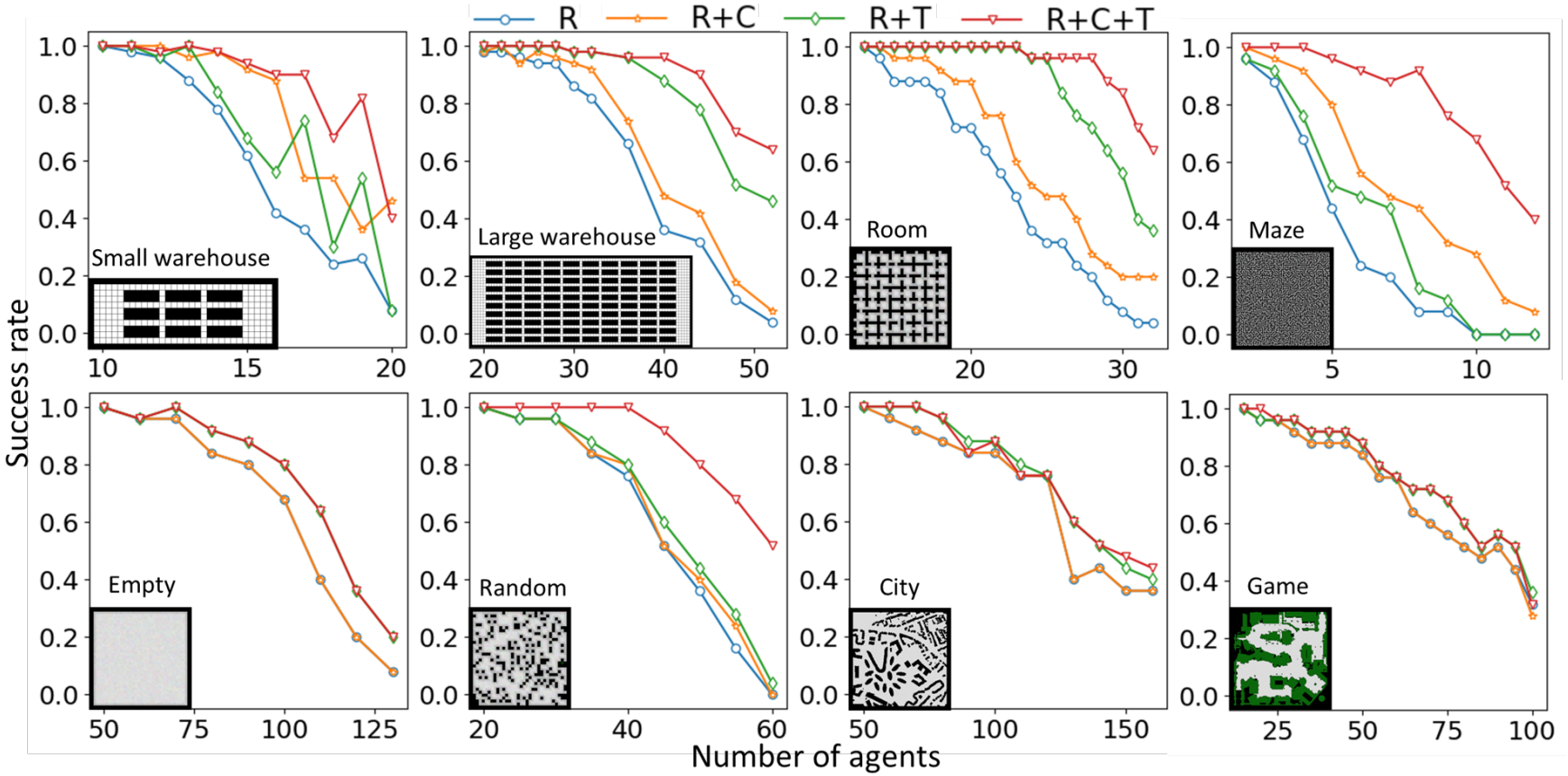


Blue Gets Priority

We add a constraint on the length of the path for blue: $l_{\text{blue}} \leq 3$

This forces red to find a bypass.

Experiments with Corridor and Target Reasoning



Recent Advances on CBS

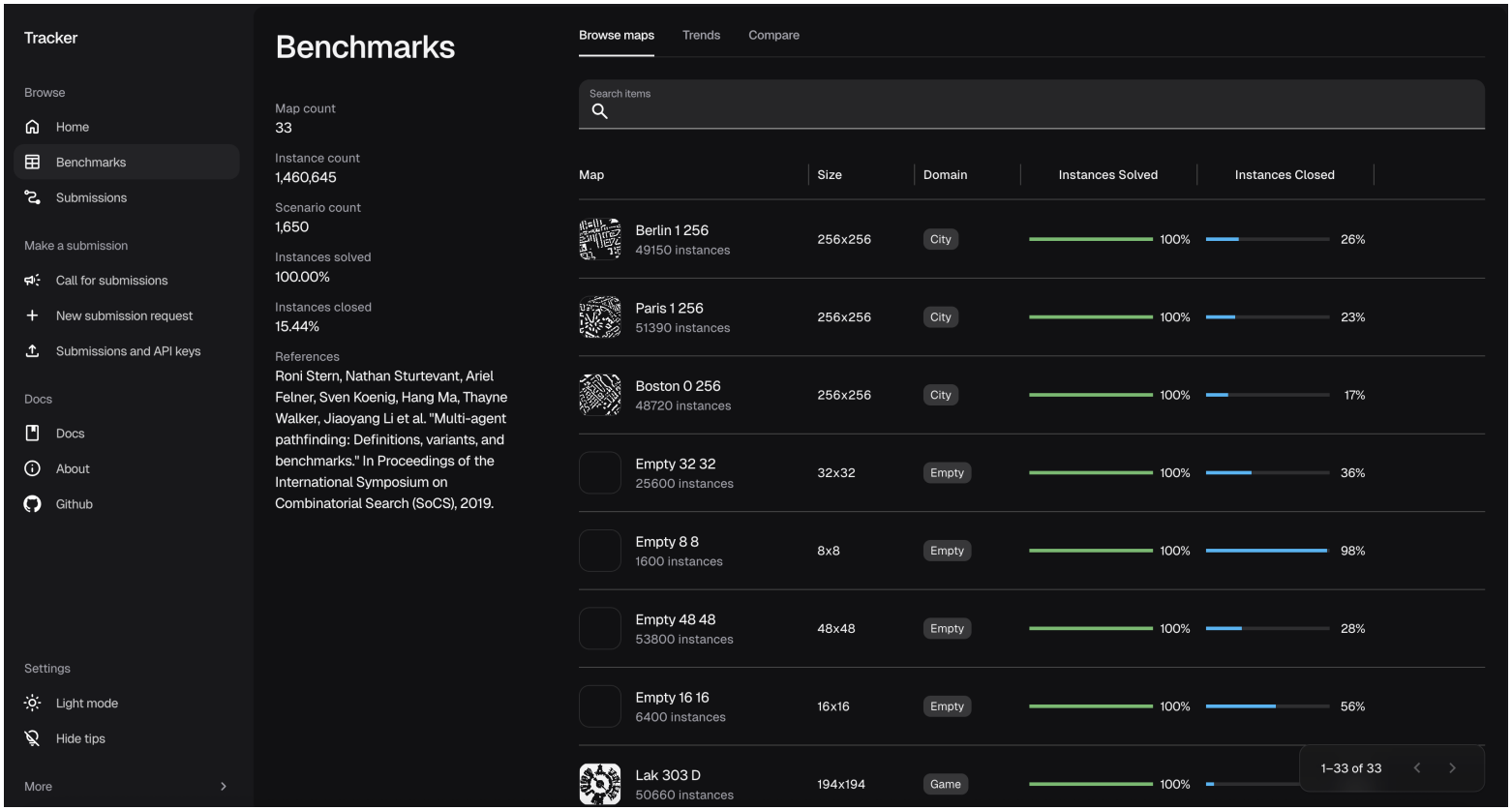
Symmetry Breaking

- Automatic symmetry detection
  - Zhang, H., Li, J., Surynek, P., Kumar, T.K.S. and Koenig, S., 2022. Multi-agent path finding with mutex propagation. Artificial Intelligence.

High-Level Heuristics

- Pairwise reasoning
  - Felner, A., Li, J., Boyarski, E., Ma, H., Cohen, L., Kumar, T.K.S. and Koenig, S., 2018. Adding heuristics to conflict-based search for multi-agent path finding. In Proceedings of ICAPS.
  - Li, J., Felner, A., Boyarski, E., Ma, H. and Koenig, S., 2019. Improved heuristics for multi-agent path finding with conflict-based search. In Proceedings of IJCAI.
- Beyond pairwise reasoning
  - Shen, B., Chen, Z., Li, J., Cheema, M.A., Harabor, D. and Stuckey, P.J., 2023. Beyond pairwise reasoning in multi-agent path finding. In Proceedings of ICAPS.

Awesome MAPF Tools



Please visit the website: https://tracker.pathfinding.ai/

End

That's it for this week