Disjoint Splitting for Multi-Agent Path Finding with Conflict-Based Search

Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Ariel Felner, Hang Ma, Sven Koenig

USC University of Southern California
MONASH University
Ben-Gurion University of the Negev

Abstract

Multi-Agent Path Finding (MAPF) is the planning problem of finding collision-free paths for a team of agents. We focus on Conflict-Based Search (CBS), a two-level tree-search state-of-the-art MAPF algorithm. The standard splitting strategy used by CBS is not disjoint, i.e., when it splits a problem into two subproblems, some solutions are shared by both subproblems, which can create duplication of search effort. In this paper, we demonstrate how to improve CBS with disjoint splitting and how to modify the low-level search of CBS to take maximal advantage of it. Experiments show that disjoint splitting increases the success rates and speeds of CBS and its variants by up to 2 orders of magnitude.

Background

Multi-Agent Path Finding (MAPF)

MAPF is the problem of finding a set of paths for a team of agents on a given graph. Each agent is required to move from a start vertex to a goal vertex, while avoiding collisions with others.

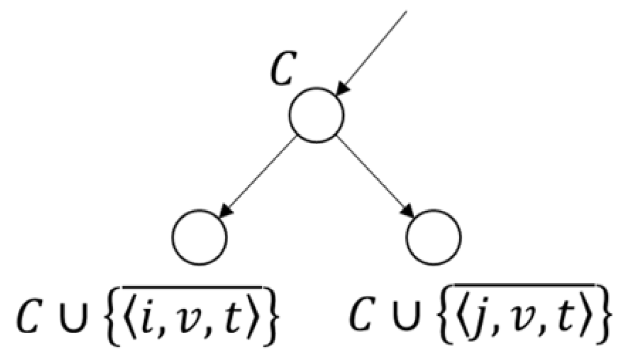
Conflict-Based Search (CBS) [2]

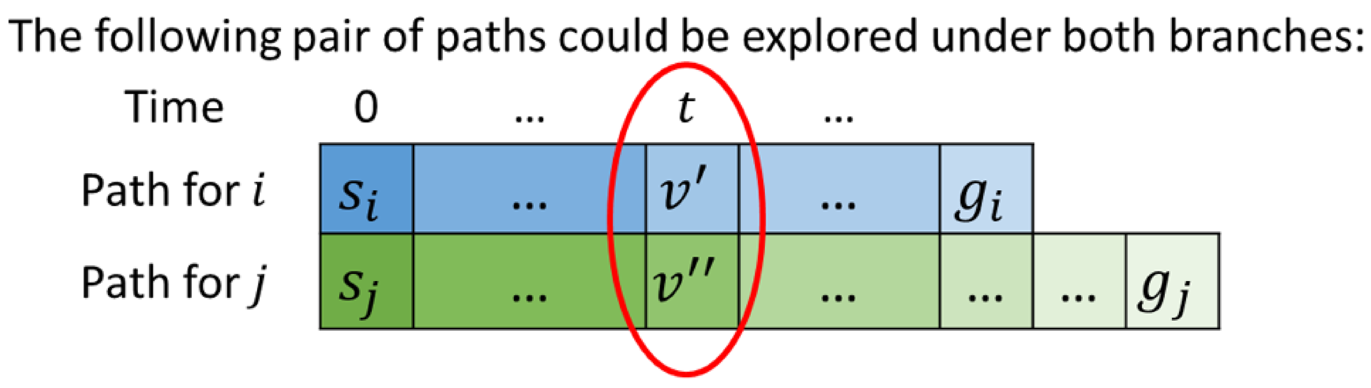
1. Find a path for every agent independently.
2. Check for collisions among paths.
3. If there is a collision where both agent i and agent j are at vertex v at time t:
   - Option 1: prohibit i from being at v at time t by adding a negative constraint $\overline{\langle i, v, t\rangle}$.
   - Option 2: prohibit j from being in v at time t by adding a negative constraint $\overline{\langle j, v, t\rangle}$.
4. Repeat until finding collision-free paths.

CBS Splitting

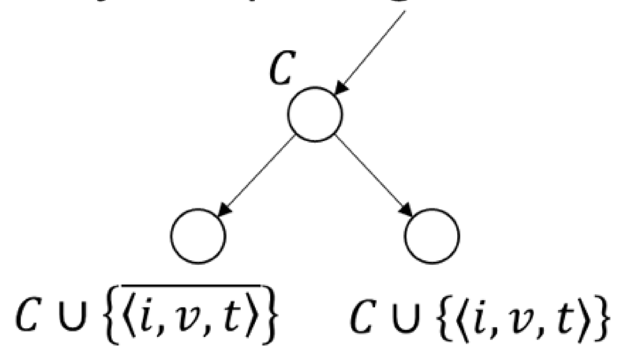
To resolve a collision between agent i and agent j at vertex v at time t:

- Non-disjoint splitting:





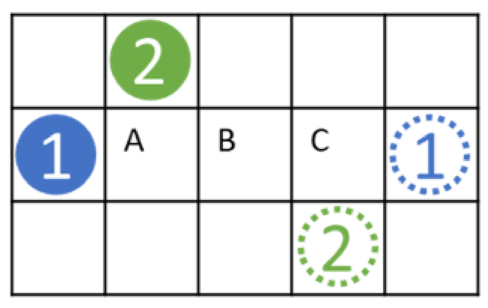
- Disjoint splitting:



- **Negative constraint $\overline{\langle i, v, t\rangle}$:**
  - Agent i cannot be at v at time t.
- **Positive constraint $\langle i, v, t\rangle$:**
  - Agent i must be at v at time t.
  - Any other agents (including agent j) cannot be at v at time t.

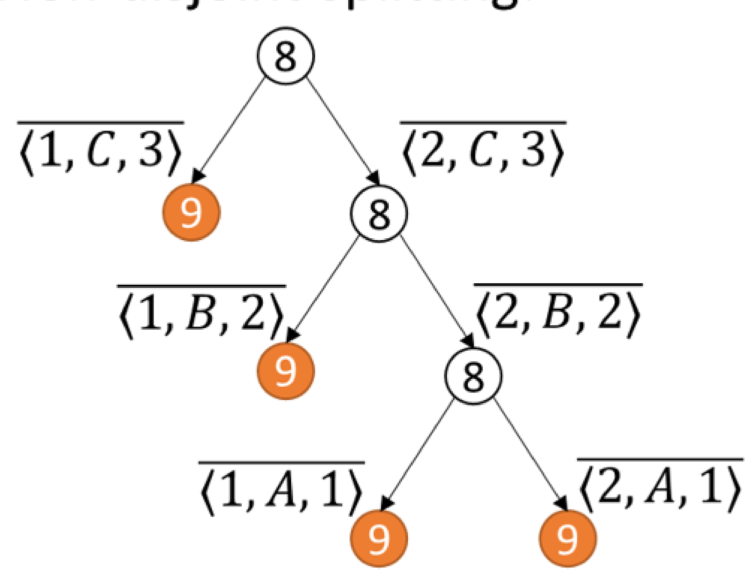
- The positive constraint $\langle i, v, t\rangle$ is **tighter** than the negative constraint $\overline{\langle j, v, t\rangle}$.
- Disjoint splitting is **complete** since one of the two constraints must hold for any candidate collision-free paths in the parent node, and it is **disjoint** since both constraints cannot hold simultaneously.
- In the low-level search, instead of replanning the entire path, disjoint splitting only replans the path segment between two positive constraints.

Example

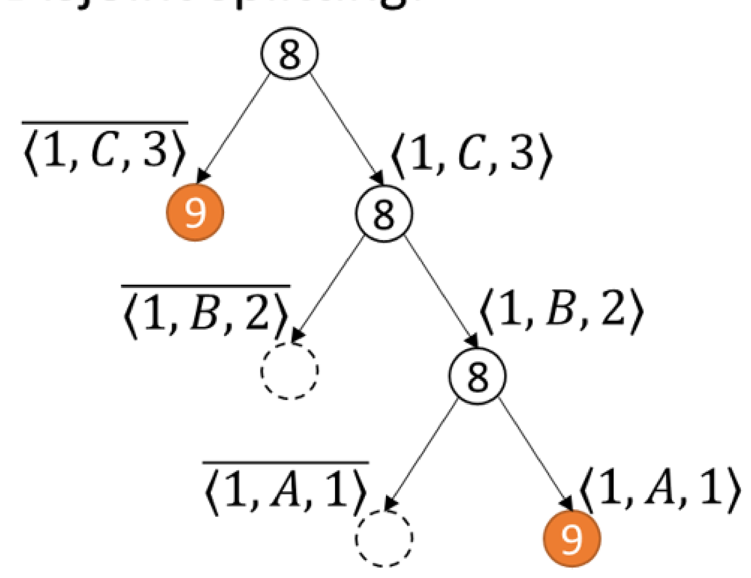


Nodes are labeled with the sum of their path costs.

- Non-disjoint splitting:



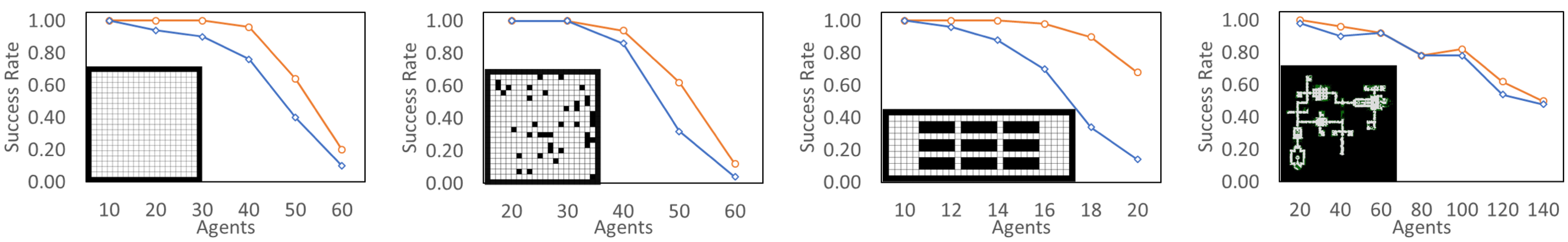
- Disjoint splitting:



The tighter positive constraints lead more often to node pruning and thus result in a smaller search tree.

Experimental Results

Figure 1: Success rate (= % of solved instances within 5 minutes) of CBS with non-disjoint splitting and disjoint splitting.



Conclusion: Disjoint splitting is at least as good as non-disjoint splitting and significantly speeds up CBS in many cases.

Open Question: Which Agent to Split on?

We examined the following strategies that split on the agent

1. uniformly at random (R);
2. whose path involves the most conflicts;
3. who has the most constraints imposed;
4. who has the least constraints imposed;
5. whose MDD has the most singletons before the conflicting time (S);
6. whose MDD has the fewest nodes at the conflicting time (W).

Empirically, strategies 1-4 performed similar, while strategies 5 and 6 performed slightly better.

Table 1: Average runtimes (in seconds) of CBS with splitting strategies 1, 5 and 6. m represents the number of agents.

Empty grid				10%-blocked grid				Warehouse grid				Game grid			
m	R	S	W	m	R	S	W	m	R	S	W	m	R	S	W
20	4.4	3.7	**0.2**	20	**0.05**	0.1	0.06	40	19.0	**13.8**	21.5	12	0.4	**0.2**	0.3
30	1.7	**1.0**	1.2	30	0.7	**0.4**	0.6	60	24.1	**22.6**	24.1	14	1.3	**0.9**	**0.9**
40	25.5	**15.5**	19.0	40	26.8	**18.4**	28.0	80	66.4	67.5	**66.3**	16	15.9	28.7	**15.6**
50	121.3	95.1	**92.5**	50	158.7	149.5	**139.7**	100	59.9	65.5	**59.8**	18	63.3	**36.8**	47.5
60	249.6	233.0	**187.6**	60	278.5	275.9	**275.5**	120	132.7	143.5	**128.4**	20	149.0	**103.6**	117.4

*An MDD for an agent is a directed acyclic graph that consists of all shortest paths for this agent.

**An MDD node is a singleton iff it is the only node at some level of the MDD.

This paper was published at ICAPS 2019 [1]. The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1409987, 1724392, 1817189 and 1837779 as well as a gift from Amazon. The research was also supported by the United States-Israel Binational Science Foundation (BSF) under grant number 2017692.

[1] J. Li, D. Harabor, P. Stuckey, A. Felner, H. Ma, and S. Koenig. Disjoint splitting for multi-agent path finding with conflict-based search. In *ICAPS*, pages 279–283, 2019.

[2] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.