Title: Multi-Agent Constraint CBS for Multi-Agent Path Finding

Author: Tuvya Macklin
Advisor: Dr. Dor Atzmon

1. Background

Multi-Agent Path Finding (MAPF)

Multi-Agent Path Finding is a computer science problem where you are given a graph and a set of agents each of which have a start and goal location on the graph. The goal is to find the optimal set of paths for all the agents with no collisions. There are two basic types of collisions/conflicts. A vertex conflict occurs when two agents occupy the same vertex at the same time step. An edge conflict occurs when two agents cross an edge between the same timesteps in opposite directions. If a set of plans contains either of these conflicts, it is an invalid solution. A common measure of optimality, and the one used in our research, is called “Flowtime”. This is evaluated by taking the sum of the time each agent takes to arrive at its goal from the initial time.

Conflict-Based Search (CBS)

Conflict-Based Search is a MAPF algorithm. It progressively develops a solution by identifying conflicts and placing constraints to resolve them. Initially, the algorithm uses A* to find paths for each agent, without considering the others. This is treated as the root node of the search tree. It then identifies a conflict to resolve. The conflict is resolved by constraining the vertex (or vertices in the case of an edge conflict) where it occurs. The constraint restricts one of the agents in the conflict from being at that vertex at that timestep. To explore all possible plans, two constraints are tried out, one for each agent in the conflict. They are applied to the scenario independently and form the first two child nodes. This is repeated for each child node recursively until a valid solution is found.

Disjoint CBS

Disjoint CBS is a variant of CBS that alters the constraints used to resolve conflicts. It is intended to be an improvement on regular CBS. Instead of the two constraints restricting the two agents respectively, only one agent from the conflict is constrained. One constraint restricts the agent from being at that vertex at that timestep, and the other *requires* the agent to be there. This second constraint is called a “positive constraint”. It effectively constrains all the agents since no other agent is allowed to be at that location since the chosen agent must be there.

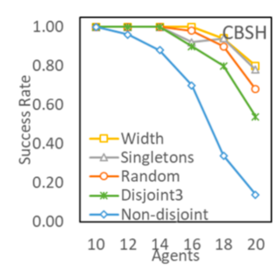


Figure 1

Figure 1 shows the success rate as the number of agents increases for various versions of Disjoint CBS as compared to Non-disjoint CBS. As you can see, the Disjoint versions perform better. [1]

2. Multi-Agent Constraint CBS (MAC-CBS)

We theorize that one of the reasons Disjoint CBS increases efficiency is because the positive constraints constrain many agents as opposed to just one. Based on this, we devised Multi-Agent Constraint CBS.

Multi-Agent Constraint CBS (MAC-CBS) resolves conflicts in the following way. After identifying the conflict to resolve and separating the two agents involved, it takes the rest of the agents and splits them into two groups. Each group is assigned to one of the agents in the conflict and the constraints generated are applied to every agent in each group.

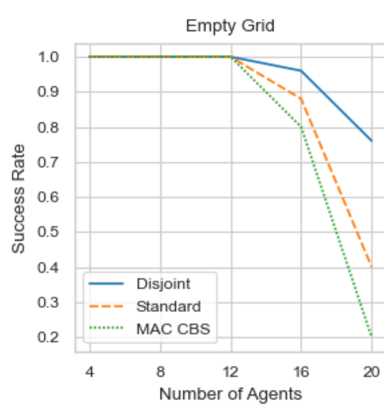
For example, if there are 10 agents and a conflict occurs between Agent 1 and Agent 2, the remaining agents (agents 3 through 10) will be divided into two groups. One group might include agents 3 through 6, and the other group might include agents 7 through 10. In one child node of the search tree, a constraint will be applied to agent 1 at the specific location and time step of the conflict, and the same constraint will also be applied to agents 3 through 6. In the other child node, a similar constraint will be applied to agent 2, along with the same constraint for agents 7 through 10.

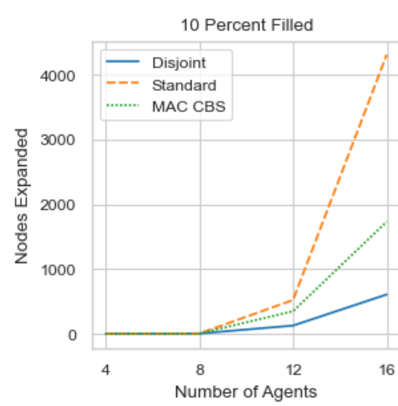
The empty map benchmarks were taken from the MovingAI benchmark library. They can be found at https://movingai.com/benchmarks/mapf/index.html.

3. Results

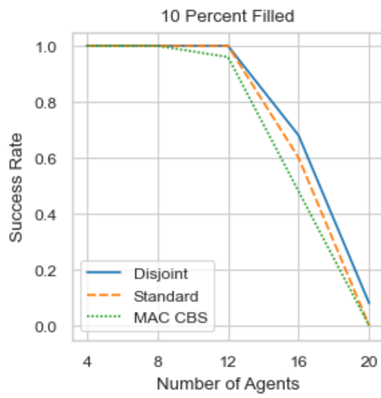
We ran tests using 50 grid-like instances. 25 were empty and 25 were randomly filled in 10% of the grid. Each instance was run 5 times with an increasing number of agents (4, 8, 12, 16, and 20). We ran each of these through regular CBS, Disjoint CBS, and MAC-CBS and gave each one minute to find a solution. The number of nodes expanded and whether the algorithm timed out were logged. See Figures 2 through 5 for the results. It seems that in terms of success rate, MAC-CBS performed poorly, even compared to regular CBS, but performed well compared to regular CBS in terms of nodes expanded. We think that the success rate is poor due to an issue with how the algorithm is implemented.

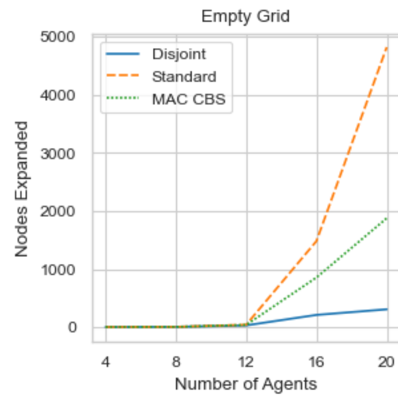
Figures 2 and 3





Figures 4 and 5





3. Conclusion

The analysis suggests that Multi-Agent Constraint CBS (MAC CBS) shows promise in standard Multi-Agent Path Finding (MAPF) due to its ability to reduce the number of nodes expanded during the search. This indicates potential for further efficiency improvements. Future work could focus on optimizing MAC CBS to enhance its performance even more.

Additionally, further work includes applying MAC-CBS to the Robust MAPF and Large Agent MAPF variants. It is challenging to apply Disjoint CBS to these problems for reasons that are beyond the scope of this article, however, it should be simple to apply MAC-CBS here. We attempted to apply MAC-CBS to Robust MAPF but were not able to due to time constraints.

4. References

1. Li, J., Harabor, D., Stuckey, P. J., Felner, A., Ma, H., & Koenig, S. (2021). Disjoint Splitting for Multi-Agent Path Finding with Conflict-Based Search. *Proceedings of the International Conference on Automated Planning and Scheduling*, *29*(1), 279-283. https://doi.org/10.1609/icaps.v29i1.3487
2. Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T. K. S., Boyarski, E., & Bartak, R. (2019). *Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks*. arXiv preprint arXiv:1906.08291.
3. Atzmon, Dor & Stern, Roni & Felner, Ariel & Wagner, Glenn & Barták, Roman & Zhou, neng-fa. (2018). Robust Multi-Agent Path Finding.