Unbounded Sub-Optimal Conflict-Based Search in Complex Domains

Thayne T. Walker[1], **Nathan R. Sturtevant**[2], **Ariel Felner**[2]
[1] University of Denver, Denver, USA
[2] University of Alberta, Edmonton, Canada
[3] Ben-Gurion University, Be'er-Sheva, Israel
thayne.walker@du.edu, nathanst@ualberta.ca, felner@bgu.ac.il

1 Introduction

Multi-Agent pathfinding (MAPF) is the problem of finding collision-free paths for multiple agents to their respective goal locations. MAPF has applications in navigation, warehouse automation, package delivery and games. Finding optimal solutions to MAPF problems is NP-hard (Yu and LaValle 2013). However, sub-optimal solutions are acceptable for some applications. Polynomial time sub-optimal solvers exist for unit cost graphs (Kornhauser, Miller, and Spirakis 1984), but may not produce valid solutions in *complex domains* with non-unit costs, non-uniform action durations, shaped agents and/or non-holonomic or kinodynamic movement constraints.

Conflict-Based Search (CBS) (Sharon et al. 2015) is a state-of-the-art algorithm for MAPF used in the context of both unit cost and more complex domains. Sub-optimal variants of CBS have focused on techniques such as focal search and alternate search prioritization schemes (Barer et al. 2014; Cohen et al. 2018). This work instead focuses on relaxing requirements for optimal constraints in order to achieve further performance gains.

2 Toward More Comprehensive Constraints

CBS searches a conflict tree (CT) in best-first order. Each node in the CT contains a potential *solution*, which is a set of single agent paths, one for each agent. A solution is *feasible* if there are no conflicts between paths. Each path is created using a low-level single agent solver. When CBS chooses to expand a CT node, it is checked for conflicts. If no conflict is detected, it is a goal node. Otherwise, CBS chooses two agents, i and j, with conflicting paths and performs a *split* operation, which means that two child nodes are created with *constraint* sets C_i and C_j. A constraint $c \in C_i$ blocks an agent from performing a set of actions. Then a single-agent solver is used for both i and j to find new paths that conform to constraints C_i and C_j respectively. The new paths then replace the paths for i and j in the respective child nodes. This process continues until a goal is found.

For completeness, C_i and C_j must be *mutually disjunctive* (Li et al. 2019). That is, no pair of conflict-free paths for agent i and j violates both C_i and C_j. Previous work relied on manual verification of the mutually disjunctive property, but this process can be generalized and automated. It can be shown that C_i and C_j are mutually disjunctive by showing that the action sets that they block, A_i and A_j, are *mutually conflicting*. Sets of actions are mutually conflicting if for all pairs of actions (a_i, a_j) in the Cartesian product $A_i \times A_j$, a_i conflicts with a_j.

Reduction to Bicliques The conflicts between a pair of action sets A_i and A_j (an example of these action sets is shown in Figure 1(a)) can be represented as a *bipartite conflict graph* (BCG). A BCG, $G = (U, V, E)$, has two sets of vertices U and V such that each $u \in U$ represents an action $a_i \in A_i$ and each $v \in V$ represents an action $a_j \in A_j$. The set of edges E consists of the subset of vertex pairs $(u, v) \in U \times V$ for which the corresponding actions $(a_i, a_j) \in A_i \times A_j$ conflict. This is shown in Figure 1(b). Although Figure 1(a) shows actions starting from the same state, actions in A_i and A_j could start from many different times and locations. This representation is generalizable to k agents with k-partite graphs. It is also generalizable to any domain that reasons about discrete actions for multiple agents (not just pathfinding domains).

In order to find a mutually conflicting set, a *biclique* $G' = (U', V', E') \subseteq G$ must be found. A biclique is a fully biconnected bipartite graph, that is, $E' = U' \times V'$, meaning all $u \in U'$ are connected via an edge to all $v \in V'$. In order to make C_i and C_j mutually disjunctive, they are constructed to block the actions represented by U' and V' respectively. In order to block the largest set of actions from A_i and A_j, a max-vertex biclique should be found in G. This can be done in polynomial time (Garey and Johnson 2002).

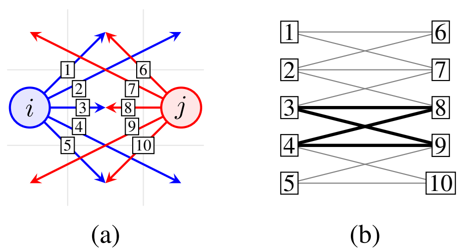

Figure 1: Example showing (a) a set of actions for agents i and j and (b) the corresponding bipartite conflict graph. A biclique subgraph is shown in bold.

3 Sub-Optimal, Complete Constraints

Larger constraint sets are designed to maximize pruning of the CT. It is possible to increase the size of constraint sets by relaxing the mutually disjunctive requirement. However, doing so may result in incompleteness in two ways: (1) termination without finding a feasible solution or (2) agents getting stuck moving in circles or waiting without a way to get to the goal. Both situations will cause CBS to run forever. In order to preserve completeness, we must detect and avoid these two conditions. For this purpose, we introduce *conditional* constraints.

Conditional Constraints A conditional constraint (as opposed to a regular *permanent* constraint) may be turned off, meaning it no longer blocks any actions. It is turned off by omitting it from the low-level re-plan step during the split operation. Turning off a constraint can cause conflicts to be re-introduced back into the solution. Therefore, mutually-conflicting actions from the biclique are always permanent and other actions not in the biclique are conditional.

There are two conditions for turning off constraints which correspond to the causes of incompleteness: (1) when a low-level re-plan returns no path, the re-plan is re-executed, omitting all conditional constraints. (2) Conditional constraints are turned off probabilistically with probability $\rho_{off} = \text{MIN}(1, \frac{\Delta-1}{d_i})$ where d_i is the length of the path for agent i in the root CT node and Δ is the depth of the CT. As the search progresses, ρ_{off} increases, causing more and more conditional constraints to be turned off until eventually only mutually disjunctive constraints remain, hence completeness is guaranteed. This approach is not optimal since a goal node may be found with some optimal actions blocked.

Constraint Set Selection Strategies If mutually disjoint sets of constraints are permanently enforced, any additional conditional constraints may be added to C_i that may help find a solution quicker. We tested two such strategies.

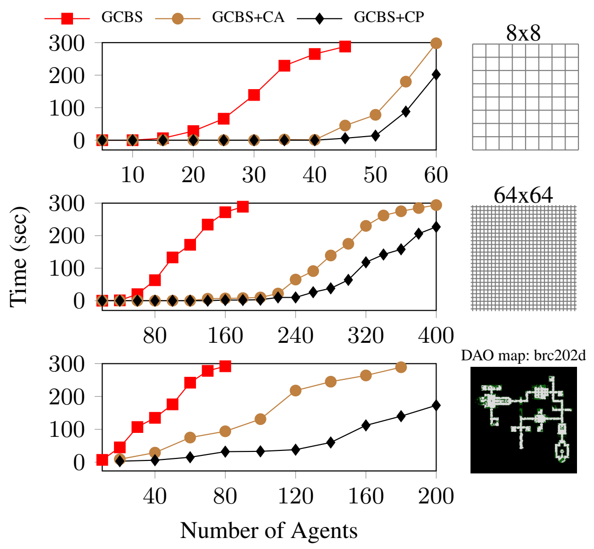

Figure 2: Comparison of runtime performance

The first strategy called *conflicting actions* (CA), builds a BCG, then finds a max-vertex biclique. All actions in the biclique are blocked using permanent constraints. All other actions in the BCG are blocked using conditional constraints. This technique pre-emptively blocks all actions within a temporospatial locality, increasing the likelihood that replanned paths will avoid collision.

The second strategy called *conflicting paths* (CP) accumulates constraints during the CBS feasibility check routine. The first conflict encountered during the check is the *core conflict*. Mutually conflicting actions between agents i and j in the core conflict are blocked using permanent constraints. For every conflict that is encountered thereafter involving agent i or j, conditional constraints for all actions in the BCG are added to C_i or C_j. As a result, when agent i (resp. j) is re-planned as part of the split operation, it will attempt to avoid conflicts with all other agents. This technique can result in a significant performance improvement because of aggressive pruning high in the CT.

4 Empirical Results

A comparison of the runtime of our two strategies versus previous state of the art, greedy CBS (GCBS) (Barer et al. 2014) is shown in Figure 2. Statistics in the plots are the average runtime over 50 problem instances on 16-connected grids with increasing numbers of agents. Any instance that was not solved in under 300 seconds is recorded as a failure and set to 300 seconds. The results show that the conflicting actions strategy (GCBS+CA) and the conflicting paths strategy (GCBS+CP) can solve for up to three to four times more agents in the same amount of time compared to GCBS.

References

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *SOCS*.

Cohen, L.; Greco, M.; Ma, H.; Hernandez, C.; Felner, A.; Koenig, S.; and Kumar, T. 2018. Anytime focal search with applications. In *IJCAI*.

Garey, M. R., and Johnson, D. S. 2002. *Computers and intractability*, volume 29. Wh Freeman New York.

Kornhauser, D. M.; Miller, G. L.; and Spirakis, P. G. 1984. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. Master's thesis, M. I. T., Dept. of Electrical Engineering and Computer Science.

Li, J.; Surynek, P.; Felner, A.; Ma, H.; and Satish, K. T. 2019. Multi-agent pathfinding for large agents. In *AAAI*.

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

Yu, J., and LaValle, S. M. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, AAAI'13, 1443–1449. AAAI Press.

Supported by BSF grant #2017692 and NSF grant #1815660