Clique Analysis and Bypassing in Continuous-Time Conflict-Based Search

Extended Abstract

Thayne T. Walker
University of Denver, Lockheed Martin Corporation
Denver, USA
thayne.walker@du.edu

Nathan R. Sturtevant
Department of Computing Science, Alberta Machine Intelligence Institute (Amii), University of Alberta
Edmonton, Canada
nathanst@ualberta.ca

Ariel Felner
Ben Gurion University
Be'er-Sheva, Israel
felner@bgu.ac.il

ABSTRACT

We study symmetry-breaking enhancements for Continuous-Time Conflict-Based Search (CCBS), a solver for continuous-time MAPF. Resolving conflict symmetries in MAPF can require an exponential amount of work. We adapt known symmetry-breaking enhancements from unit-cost domains for CCBS. We then improve upon these to produce a new state of the art algorithm: CCBS with disjoint k-partite cliques (CCBS+DK). Finally, we show empirically that CCBS+DK solves for up to 20% more agents in the same amount of time when compared to previous state of the art.

KEYWORDS

Multi-Agent Pathfinding, Heuristic Search

ACM Reference Format:
Thayne T. Walker, Nathan R. Sturtevant, and Ariel Felner. 2024. Clique Analysis and Bypassing in Continuous-Time Conflict-Based Search: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

1 INTRODUCTION

The objective of multi-agent pathfinding (MAPF) is to find paths for multiple agents being routed on a graph embedded in a metric space such that agents' paths are non-conflicting (i.e., not overlapping). MAPF has applications in warehouses [11], package delivery [6], games [4], firefighting [14], search and rescue [15] and intersection management [7, 18]. We seek optimal solutions to the *continuous-time MAPF problem* [1, 19], denoted MAPF_R for real-valued action durations and costs on general graphs.

Continuous-Time Conflict-Based Search (CCBS) [3] is an optimal solver for MAPF_R. CCBS re-formulates the Conflict-Based Search (CBS) algorithm [16] to allow variable-duration wait actions which account for continuous-time execution. CCBS was shown to be effective on settings that are inspired by real-world applications.

Conflict symmetries pose a problem for CCBS. A *conflict symmetry* [10] occurs when agents are situated such that one or more agents must increase their path cost to avoid conflict. For optimal algorithms like CCBS, many alternate paths must be explored before proving that the cost increase is necessary. Conflict symmetries can require an exponential amount of work to resolve [10]. We study the effectiveness of bypassing [5] and biclique constraints [20] in CCBS. We then combine these techniques with disjoint splitting [2, 9] to achieve new state of the art results.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), T.T. Walker, N.R. Sturtevant, A. Felner, May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

2 CONTINUOUS-TIME CBS

CCBS searches on two levels. The *high level* searches a constraint tree (*CT*). Each node N in the *CT* contains a *solution* $N.\Pi$, and a set of *constraints* $N.C$. Each path $\pi_i \in N.\Pi$ of agent i in N is constructed using a *low-level* search which respects constraints. A *constraint* blocks an agent from performing action(s) and is defined as a tuple $\langle i, v, t_1, t_2 \rangle$, where i is the agent, v is the vertex and $(t_1, t_2]$ is the continuous time range in which the agent must avoid the vertex. Next, CCBS checks for conflicts between any pairs of paths π_i and π_j in $N.\Pi$. If $N.\Pi$ contains no conflict, then N is a goal node and CCBS terminates. If $N.\Pi$ contains a conflict, then CCBS performs a *split*, where it generates two child nodes N_i and N_j of N and adds constraints c_i and c_j to $N_i.C$ and $N_j.C$ respectively. The *low level* solver uses safe-interval path planning (SIPP) [12] which treats the constraints as *safe intervals* to plan safe paths which avoid the conflict. CCBS systematically checks for conflicts, generates child nodes with constraints and re-plans the conflicting paths. CCBS prioritizes the search by the total cost of $N.\Pi$. It terminates when a feasible solution is found.

3 BYPASSING AND BICLIQUE CONSTRAINTS

When a split occurs, the *bypass* enhancement (BP) [5] inspects the re-planned paths. If a new path is available which: (1) does not have an increased cost, (2) respects all constraints and (3) has fewer conflicts with all agents, this new path is called a *bypass*. If a bypass is found, child nodes are not generated, instead, the current node is updated with the bypass path and re-inserted into the OPEN list. This enhancement improves performance by avoiding splits in the CT, eliminating new sub-trees. Our results show that BP is effective in problem instances with similarities to "classic" MAPF, but less effective in certain continuous-time settings.

When dealing with agents moving in continuous time and space, one action taken by an agent may conflict with a set of actions by other agents. Biclique constraints (BC) [20] computes *sets* of one or more new constraints C_i and C_j, for two conflicting agents

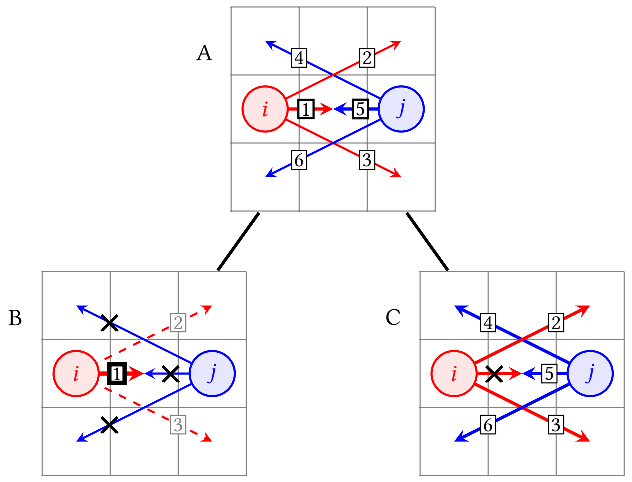

Figure 1: An example of a disjoint split with biclique constraints.

by performing *bipartite conflict analysis.* Although the original formulation for BC was shown to be very effective in continuous-time domains with *fixed* wait actions [20], when *arbitrary* wait actions are allowed, we found that BC was often detrimental to performance. With arbitrary wait times, BC causes agents to wait for an insufficient amount of time. This causes the conflict to recur at a slightly later time, resulting in another split in the sub-tree for the same, slightly delayed actions. However, when BC is combined with disjoint splitting (DS) [2, 9], *disjoint bicliques* (DB), there is a marked improvement. This improvement is possible because of the positive-negative nature of disjoint splits.

The placement of biclique constraints in a disjoint split is illustrated in Figure 1. Node A is a node in the CT with a conflict between actions 1 and 5 shown with bold arrows. Nodes B and C are child nodes. Node B shows the positive constraint for the red agent in bold for action 1, with negative constraints for the blue agent's conflicting actions shown with 'x's. The other actions for the red agent in node B are dashed, meaning that they are no longer reachable because of the positive constraint. Finally, node C shows a single negative constraint that mirrors the positive constraint in node B. When we use a set of negative constraints paired with a positive one, it avoids further conflicts in the CT, resulting in potentially exponential amounts of work avoided.

Additionally, when multiple agents conflict with the positively constrained action, it is helpful to additionally constrain these agents using negative constraints. We call this approach *disjoint k-agent cliques* (DK). For DK, we perform bipartite conflict analysis for all agents in conflict with the positively-constrained action and enforce negative, continuous-time constraints for all conflicting actions for up to all k agents. The result is that all agents avoid performing actions which conflict with the positively-constrained action, avoiding further splits in the tree.

4 EMPIRICAL RESULTS

We now analyze the enhancements, namely: bypass (BP) and disjoint k-partite cliques (DK). Disjoint bicliques (DB) is effective, but less effective than DK, hence, it is omitted from the results. All tests in this section were performed single-threaded, on cloud compute instances that report an Intel Xeon 2.5GHz processor. In addition to three roadmaps: "sparse", "dense" and "super-dense" [1], we test our enhancements in all 44 of the MAPF grid benchmarks [17] in continuous-time domains. Agents are circular, with a radius of $\sqrt{2}/4$, but any radius <0.5 could be used. All tests were run by starting with 2 agents and incrementing the number of agents by 2 until the problem instance became unsolvable in under 30 seconds.

Table 1: Summary of problems solved on 4-, 8-, 16- and 32-neighbor grid MAPF benchmarks and roadmaps

Graph Type	Base	BP	DK	BP+DK
4-neighbor	23,628	**24,872**	23,642	**<u>25,378</u>**
8-neighbor	28,216	**29,346**	28,660	**<u>30,110</u>**
16-neighbor	25,116	**25,634**	25,308	**<u>26,546</u>**
32-neighbor	21,892	21,582	22,416	**<u>24,090</u>**
Sparse	**434**	396	**440**	**<u>444</u>**
Dense	604	604	630	**<u>712</u>**
Super-dense	402	402	442	**<u>474</u>**

Table 1 shows the total of agents solvable for each problem category by graph type. The best result is underlined and those within the 95th percentile of the best are in bold. The label "Base" is the previous state of the art for CCBS. The first four lines of the table shows totals for MAPF grid benchmarks with 2^k neighborhood [13] connectivities, namely 4-, 8-, 16- and 32-neighborhoods. The last three lines of the table shows results for probabilistic roadmaps [8]. "Sparse", "Dense" and "Super-dense" roadmaps have a mean vertex degree of 4.2, 16.7, and 100.4 respectively.

4-neighborhood maps have no crossing edges, hence few opportunities for biclique constraints to block multiple actions at once. As the connectivity increases from 8-neighborhood up to super dense roadmaps, the density of crossing edges increases. As the edge density increases, the effectiveness of BP decreases, this is because equivalent-cost alternate paths become scarce so that bypasses cannot be found. On the other hand, as the edge density increases, the size of constraint sets produced by biclique constraints increases dramatically, increasing the effectiveness of DK.

In summary, combining BP with DK consistently beats state of the art by statistically significant margins. Compared to the previous state of the art, our enhancements allow solutions for up to 10% more agents in 32-neighbor grid maps and for up to about 18% more agents in super dense roadmaps.

ACKNOWLEDGMENTS

The research at the University of Denver was supported by the National Science Foundation (NSF) grant number 1815660 and Lockheed Martin Corporation. Research at the University of Alberta was funded by the Canada CIFAR AI Chairs Program. We also acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC). Research at Ben Gurion University was supported by BSF grant number 2021643.

REFERENCES

[1] Anton Andreychuk, Konstantin Yakovlev, Dor Atzmon, and Roni Stern. 2019. Multi-Agent Pathfinding with Continuous Time. In *International Joint Conference on Artificial Intelligence*. 39–45.

[2] Anton Andreychuk, Konstantin Yakovlev, Eli Boyarski, and Roni Stern. 2021. Improving continuous-time conflict based search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11220–11227.

[3] Anton Andreychuk, Konstantin Yakovlev, Pavel Surynek, Dor Atzmon, and Roni Stern. 2022. Multi-agent pathfinding with continuous time. *Artificial Intelligence* 305 (2022), 103662.

[4] Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage, and Dana Nau. 2013. Pathfinding in games. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[5] Eli Boyarski, Ariel Felner, Guni Sharon, and Roni Stern. 2015. Don't Split, Try To Work It Out: Bypassing Conflicts in Multi-Agent Pathfinding. In *International Conference on Automated Planning and Scheduling*. 47–51.

[6] Shushman Choudhury, Kiril Solovey, Mykel J Kochenderfer, and Marco Pavone. 2021. Efficient large-scale multi-drone delivery using transit networks. *Journal of Artificial Intelligence Research* 70 (2021), 757–788.

[7] Kurt Dresner and Peter Stone. 2008. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research* 31 (2008), 591–656.

[8] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 4 (1996), 566–580.

[9] Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Ariel Felner, Hang Ma, and Sven Koenig. 2019. Disjoint Splitting for Multi-Agent Path Finding with Conflict-Based Search. In *International Conference on Automated Planning and Scheduling*. 279–283.

[10] Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, and Koenig Sven. 2019. Symmetry-Breaking Constraints for Grid-Based Multi-Agent Pathfinding. In *AAAI Conference on Artificial Intelligence*. 6087–6095.

[11] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. 2021. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11272–11281.

[12] Mike Phillips and Maxim Likhachev. 2011. Sipp: Safe interval path planning for dynamic environments. In *International Conference on Robotics and Automation*. IEEE, 5628–5635.

[13] Nicolas Rivera, Carlos Hernández, and Jorge A Baier. 2017. Grid Pathfinding on the 2k Neighborhoods.. In *AAAI Conference on Artificial Intelligence*. 891–897.

[14] Juan Jesús Roldán-Gómez, Eduardo González-Gironda, and Antonio Barrientos. 2021. A survey on robotic technologies for forest firefighting: Applying drone swarms to improve firefighters' efficiency and safety. *Applied Sciences* 11, 1 (2021), 363.

[15] Jürgen Scherer, Saeed Yahyanejad, Samira Hayat, Evsen Yanmaz, Torsten Andre, Asif Khan, Vladimir Vukadinovic, Christian Bettstetter, Hermann Hellwagner, and Bernhard Rinner. 2015. An autonomous multi-UAV system for search and rescue. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*. 33–38.

[16] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. 2015. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence Journal* 219 (2015), 40–66.

[17] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, T. K. Satish Kumar, Eli Boyarski, and Roman Barták. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *International Symposium on Combinatorial Search*. 151–159.

[18] Jiří Švancara, Marek Vlk, Roni Stern, Dor Atzmon, and Roman Barták. 2019. Online multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 7732–7739.

[19] Thayne T Walker, Nathan R Sturtevant, and Ariel Felner. 2018. Extended Increasing Cost Tree Search for Non-Unit Cost Domains.. In *International Joint Conference on Artificial Intelligence*. 534–540.

[20] Thayne T Walker, Nathan R Sturtevant, and Ariel Felner. 2020. Generalized and Sub-Optimal Bipartite Constraints for Conflict-Based Search. In *AAAI Conference on Artificial Intelligence*.