CBS with Continuous-Time Revisit

Andy Li[1], Zhe Chen[1], Danial Harabor[1]

[1]Department of Data Science and Artificial Intelligence, Monash University, Melbourne, Australia
{andy.li,zhe.chen,daniel.harabor}@monash.edu

Abstract

In recent years, researchers introduced the Multi-Agent Path Finding in Continuous Time (MAPF$_R$) problem. Conflict-based search with Continuous Time (CCBS), a variant of CBS for discrete MAPF, aims to solve MAPF$_R$ with completeness and optimality guarantees. However, CCBS overlooked the fact that search algorithms only guarantee termination and return the optimal solution with a finite amount of search nodes. In this paper, we show that CCBS is incomplete, reveal the gaps in the existing implementation, demonstrate that patching is non-trivial, and discuss the next steps.

Introduction

Multi-Agent Path Finding (MAPF) is a problem that asks us plan collision-free paths for a set of moving agents. In classical MAPF, agents operate on a grid and all actions have unit costs. Many optimal algorithms for classical MAPF appear in the literature, including a large number employing Conflict Based Search (CBS) (Sharon et al. 2015; Boyarski et al. 2015; Li et al. 2021; Shen et al. 2023).

MAPF$_R$ is a generalisation of the classical MAPF model, where agents can move and wait in continuous time. Continuous Time CBS (CCBS) (Andreychuk et al. 2022) claims to be the first optimal algorithm for this problem. In recent years, a significant amount of improvements (Andreychuk et al. 2021; Walker, Sturtevant, and Felner 2020; Walker et al. 2021; Walker, Sturtevant, and Felner 2024) and extensions (Yakovlev, Andreychuk, and Stern 2024; Kulhan and Surynek 2023) have been made on top of CCBS. However, when we stopped for a moment and checked the base we stood on, we noticed that the proposed constraints to resolve conflicts between move motion and wait motion in CCBS, although sound, do not guarantee search termination. This is because the proposed constraints remove single points in the real number domain, leaving an infinite number of equivalent collision solutions unresolved. Moreover, the existing publicly accessible implementation of the CCBS algorithm is inconsistent with the design of the algorithm, which terminates the search by aggressively eliminating wait motions but returns suboptimal solutions. We then demonstrated that patching the algorithm is non-trivial, where we constructed a pair of sound constraints that is proved to eliminate collision solutions in a maximum range. However, it still failed to guarantee termination.

Problem Definition

Following Andreychuk et al. (2022), a continuous-time MAPF problem (MAPF$_R$) takes as input a weighted graph $G = \{V, E\}$ and a team of k agents $\{a_1, ..., a_k\}$. Every vertex in G maps to a coordinate in a metric space $\mathcal{M}$, and every edge is a connection between two vertices. Each agent has a start vertex $s_i \in V$ and a goal vertex $g_i \in V$. No agents shares the same start vertex: $\forall i \in \{1, ..., k\} \nexists j \in \{1, ..., k\} \setminus i : s_i = s_j$, as well as for goal vertex. A plan π_i with length l for agent a_i is a sequence of motions m_i^j with their start time τ_i^j, such as $\{m_i^1@\tau_i^1, ..., m_i^l@\tau_i^l\}$. Each motion m consists $\langle m.\varphi, m.D\rangle$:

- $m.D$ is a positive real number duration of the motion.
- $m.\varphi$ is a motion function $m_\varphi : [0, m.D] \to \mathcal{M}$ that maps time to a coordinate in the metric space.

With such a motion function, the problem is able to model the agent at a non-const speed and follow an arbitrary geometric curve. The motion function $m.\varphi(t)$ outputs the coordinate of the agent by executing motion m for t amount of duration. A motion function can be either a moving motion or a waiting motion. Moving motion function $m.\varphi(0)$ returns the current vertex v and $m.\varphi(m.D)$ returns the next vertex v', Waiting motion is when $v' = v$ and $m.\varphi(t)$ returns the coordinate of v for the entirety of $[0, m.D]$. In the MAPF$_R$ problem, we assume there is a finite set of move motions, as there is a finite number of edges in G, and the motion duration is fixed as edge weights. However, the set of wait motion is uncountably infinite, as agents can wait for any positive real amount of time. In a valid plan π_i, the from vertex v^n of $m_i^n : (v^n, v'^n)$ has to be the to vertex v'^{n-1} of previous motion $m_i^{n-1} : (v^{n-1}, v'^{n-1})$. Simiarily, the starting time τ_i^n of the motion m_i^n is always the finish time $\tau_i^{n-1} + m_i^{n-1}.D$ of the previous motion m_i^{n-1}.

A feasible solution to a MAPF$_R$ problem is a set of plans $\Pi = \{\pi_1, ..., \pi_k\}$ where every planned motion of each agent a_i is conflict-free with respect to every planned motion of every other agent $a_j \neq a_i$. A **conflict** $\langle a_i, a_j, m_i@\tau_i, m_j@\tau_j\rangle$ between two agents a_i and a_j happens when a collision detection function $InCollision(m_i@\tau_i, m_j@\tau_j)$ returns $true$, indicating there exists a time t where the shapes of two agents overlaps.

Following Andreychuk et al. (2022), we use disc-shaped agents with radius r in this paper. Thus function

$InCollision(m_i@\tau_i, m_j@\tau_j)$ returns true if there exists a time t that the distance between the centre of the two agents a_i and a_j, is smaller than $2r$, $||m_i - m_j||_2 < 2r$. Note that if $InCollision(m_i@\tau_i, m_j@\tau_j) = true$, then there must exist two time points t_s and t_e, where t_s is the first time when two agents' distance equal to $2r$, and t_e the second time when two agents' distance equal to $2r$ or the time one of the motion finishes. We refer to the interval (t_s, t_e) as the **collision interval**. Our objective is to compute a feasible plan that minimises the *Sum of Individual (path) Cost* (SIC), where $SIC = \sum_{i\in[1,m]} et_i$ with et_i indicating the end time of the last motion on path π_i.

Background

CBS with Continuous Time

CBS with Continuous Time (CCBS) (Andreychuk et al. 2022) is a two-level search algorithm that aims to solve the MAPF$_R$ problem optimally. CCBS assumes all agents have the same speed and the travel time for an edge is the same as the edge length. The high-level of CCBS is a best-first search on a binary *Constraint Tree* (CT). Each CT node N contains a set of conflict-resolving constraints $N.constraints$ applicable to agents, a solution $N.\Pi$ that satisfies $N.constraints$, and the cost $N.g$ of N is the sum of individual cost (SIC) of $N.\Pi$. The low-level search of CCBS finds a temporal shortest path that satisfies all the constraints for each agent at the current node. CCBS employs *Safe Interval Path Planning* (SIPP) (Phillips and Likhachev 2011) to compute paths. CCBS starts by generating a root CT node that contains no constraints ($N.constraints = \phi$) and a solution of individual shortest plans for all agents. It then iteratively selects a node N with the smallest $N.g$. If N contains no conflicts, then it is the goal node. Otherwise, it selects one of the conflicts that occurred in $N.\Pi$, let's say $\langle a_i, a_j, m_i@\tau_i, m_j@\tau_j\rangle$. Then CCBS generates two child CT nodes N_i and N_j with each node appending one more conflict-resolving constraint C to $N.constraints$, $N_i.constraints = N.constraints \cup C_i$ (resp. $N_j.constraints$). An **unsafe interval** $I_i : [t_i, t'_i)$ (resp. I_j) for a_i (resp. a_j) is a time period where if the starting time τ_i (resp. τ_j) of the motion m_i (resp. m_j) is within the I_i (resp. I_j), then it will conflict with $m_j@\tau_j$ (resp. $m_i@\tau_i$). A **motion constraint** $C_i : \overline{\langle a_i, m_i, I_i\rangle}$ indicates that a_i cannot start the motion m_i within the *unsafe interval* I_i.

The Claims of CCBS

Following Andreychuk et al. (2022), we define a *Sound pair of constraints* as:

Definition 1 (Sound pair of constraints)**.** *Given a MAPF$_R$ problem, a constraints pair is sound iff in every optimal feasible solution, at least one of these constraints holds.*

Andreychuk et al. (2022) makes the following claims:

Claim 1. *The pair of constraints for resolving a conflict is sound, and any MAPF$_R$ solution that violates both constraints must have a conflict:*

Lemma 1. *For any CCBS conflict $\langle a_i, a_j, m_i@\tau_i, m_j@\tau_j\rangle$, and corresponding unsafe intervals I_i and I_j, the pair of CCBS constraints $\overline{\langle a_i, m_i, I_i\rangle}$ and $\overline{\langle a_j, m_j, I_j\rangle}$ is sound.*

Claim 2. *CCBS is complete and is guaranteed to return an optimal solution if one exists.*

To support Claim 2, Andreychuk et al. (2022) argue that for a CT node N, letting $\pi(N)$ be all valid MAPF$_R$ solutions that satisfy $N.constraints$, $N.g$ be the cost of N, and N_1 and N_2 be the children of N, the following two conditions hold for any N that is not a goal node.

1. $\pi(N) = \pi(N_1) \cup \pi(N_2)$
2. $N.g \leq min(N_1.g, N_2.g)$

The first condition holds because N_1 and N_2 are constrained by a sound pair of constraints (Definition 1 and Lemma 1). The second condition holds because $N.solution$ by construction is the lowest cost solution that satisfies the constraints in N, and the constraints in N_1 and N_2 are a superset of constraint in N. These two conditions, when combined, aim to ensure the completeness of CCBS, guaranteeing that an optimal solution will be found if one exists. The first condition guarantees that any valid solution is reachable via one of the un-expanded CT nodes. As CCBS performs a best-first search over CT and explores CT nodes with minimal cost first, the second condition helps CCBS guarantee to find an optimal MAPF$_R$ solution.

However, existing work overlooked the fact that CBS, best-first search or any tree-based state-space search only guarantees to terminate and return the optimal solution with a countable number of search states.

Infinite Nodes Expansions

Andreychuk et al. (2022) assumes that the number of possible nodes in CT is finite. This holds if the problem only has move motions however, there are infinitely many wait motions. As a result, the search algorithm has to explore an infinite number of possible nodes to find the optimal solution. In this section, we will demonstrate how CCBS failed to eliminate conflicts between wait motion and move motion and generate infinite number of nodes.

Conflict Resolution Failure

Given a CT node N with cost $N.g$ and a collision $c = \langle a_i, a_j, w_i@t_1, m_j@t_2\rangle$ between a wait motion $w_i = \langle (v_1, v_1), w_i.D\rangle$ for a_i and a move motion $m_j = \langle (v_2, v_3), ||(v_2, v_3)||_2\rangle$ for a_j, where $||(v_2, v_3)||_2$ is the length of edge (v_2, v_3).

CCBS generates a sound pair of *motion constraints*:

- $\overline{\langle a_i, w_i, [t_1, t'_1)\rangle}$, which forbids a_i to take the wait motion w_i, with duration $w_i.D$, from t_1 to t'_1, and
- $\overline{\langle a_j, m_i, [t_2, t'_2)\rangle}$, which forbids a_j to take the move motion m_j from t_2 to t'_2.

The intervals $[t_1, t'_1)$ and $[t_2, t'_2)$ are the maximal possible *unsafe intervals* for w_i and m_j, respectively. If a_i starts w_i at any time within $[t_1, t'_1)$, it will inevitably collide with a_j starting m_j at t_2. Similarly, if a_j starts m_j at any time within $[t_2, t'_2)$, it will inevitably collide with a_i starting w_i at t_1.

Although a sound pair of motion constraints guarantees that no collision-free solutions will be eliminated on splitting, it is not enough for CCBS to find an optimal solution with wait motion that can be any real number duration.

The first issue is that using *motion constraints* to eliminate conflict between wait motions and move motions only removes one duration option for wait motion on (v_1, v_1). But there are an infinite number of duration options for motion (v_1, v_1) that collide with $m_j@t_2$:

Lemma 2. *Let N' with constraint $\overline{\langle a_i, w_i, [t_1, t_1')\rangle}$ be a child node of N with a collision interval (t_1^c, t_2^c). The constraint eliminates solutions that w_i conflicts with $m_j@t_2$, but permits an infinite number of solutions that any wait motion $w_i' = \langle(v_1, v_1), w_i.D + \delta\rangle$ conflicts with $m_j@t_2$, where $\delta \in (w_i.D - t_1^c + t_1, \infty) \setminus \{0\}$.*

Proof. Since the duration of wait motions can be any positive real number, thus there are infinite number of choices on δ. Thus leading to infinite number of choices on w_i' with δ in the given range, i.e. collision happens when $t_1 + w_i.D - \delta > t_1^c$. For every w_i', executing w_i' at any time $t' \in [t_1, t_1')$ collides with $m_j@t_2$, since the duration a_i waits on v_i is $[t', t' + w_i'.D)$ which always covers the duration of executing w_i at the same time and w_i collide with $m_j@t_2$. □

Termination Failure

With an infinite number of collision solutions permitted by one of the child nodes in each splitting, CCBS has to explore an infinite number of nodes and eliminate an infinite number of conflicts to find an optimal solution, if it exists:

Theorem 1. *Given a CT rooted at N, if there exists an optimal solution node N^* with cost $c^* = N.g + \Delta$ and Δ is a non-zero positive real number, CCBS, which uses pairs of* motion constraints *to resolve conflicts, have to explore an infinite number of nodes to find N^* if N has conflicts between wait and move motions.*

Proof. Assuming the problem only has a_i and a_j, where replacing the w_i with $w_i' = \langle(v_1, v_1), w_i.D + \delta\rangle$ in $N.\Pi$ and delaying the executing of following motions for $\delta > 0$ resulting in a new solution π' with cost $N.g + \delta$. There are an infinite number of solutions with a cost between $N.g$ and $N.g + \Delta$, since the number of options for $\delta \in (0, \Delta)$ is infinite in a real number domain. In case of $\delta < 0$, if $w_i' = \langle(v_1, v_1), w_i.D - \delta\rangle$ and $w_i'' = \langle(v_1, v_1), \delta\rangle$, $w_i@t$ is equivalent to $w_i'@t$ followed by $w_i''@(t + (w_i.D - \delta))$. Thus there are infinite equivalent solutions with cost $N.g$.

According to Lemma 2, each CCBS split removes only one duration option for $w_i.\varphi = (v_1, v_1)$ in one of its child nodes N' and permits an infinite number of duration choices between $(w_i.D, w_i.D + \Delta)$ and an infinite number of ways to replace w_i with two or more wait combinations, thus CCBS have to split or expand for infinite layers to remove all conflict solutions with cost in range $(N.g, N.g + \Delta)$ to reach N^* since it explores the CT in a best first manner. □

As a result, CCBS cannot always terminate and return the solution if strictly resolving conflicts with *motion constraints*. Even considering that most computers cannot operate on real number domains, CCBS have to expand for every representable number on wait motion durations to find a solution, which is almost impossible since each expansion doubles the amount of remaining work.

Alternative Implementations

While the previous section shows that CCBS is not able to terminate if there exist conflicts between wait motions and move motions, however, the existing publicly accessible CCBS implementation[1] from Andreychuk et al. (2022) often terminates with the existence of such conflicts. It resolves conflicts between wait and move motions by forbidding agents present on a vertex within a given duration. In this section, we show that such implementation eliminates collision-free solutions and discuss alternative implementations that utilise this type of constraint.

Vertex Constraint

Instead of motion constraints, the existing implementation defines a different type of constraint to resolve conflicts between wait and move motions, we call it **vertex constraint**, which is similar to the constraints proposed by Atzmon et al. (2018). *Vertex constraint* forbids the existence of an agent a_i within a given *time range* $[t, t')$. We use $\overline{\langle a, v, (t, t')\rangle}$ to denote such constraint. Compared to a *motion constraint*, which forbids the starting of a wait motion (with a single fixed duration), vertex constraint forbids agent a executing any wait motion $\langle(v, v), D\rangle$ at any time τ, as long as $[\tau, \tau + D]$ overlaps with (t, t').

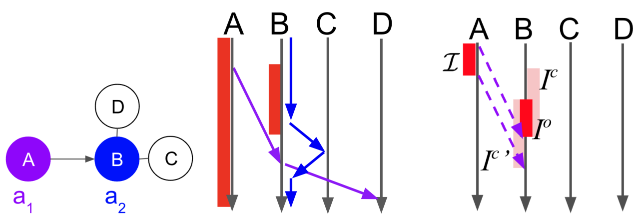
Given the same CT node N with the same conflict $c = \langle a_i, a_j, w_i@t_1, m_j@t_2\rangle$, the existing implementation introduces a pair of *vertex and motion constraints*:

- $\overline{\langle a_i, v_i, (t_s, t_e)\rangle}$, a *vertex constraint* forbids a_i use v_i, and
- $\overline{\langle a_j, m_j, [t_2, t_2')\rangle}$, an *motion constraint* forbids a_j execute action m_j.

In this constraints pair, (t_s, t_e) is the collision interval between $w_i@t_1$ and $m_j@t_2$ and $[t_2, t_2')$ is the unsafe interval for a_j that executing m_j at any time in this range must collide with $w_i@t_1$ for a_i. There is a detailed example in Appendix A on how to compute the collision interval.

Elimination of Feasible Solutions An example problem is shown in Figure 1, Fig1a shows the spatial graph and Fig1b shows the constraints and agents' movements over time, time flows from top to bottom. a_1 moves from A to B to D and a_2 parks at its goal B. When collision raises between a_1 moves from A to B and a_2 parks at B (waits since time 0), the existing implementation generates two constraints: The first constraint forbids a_1 moving to B (shown as the red range on A in Fig1b), since moving to B at any time collides with the wait motion of a_2 at B. The second constraint forbids a_2 from occupying B within the red region. These constraints eliminate the solution that a_1 waits a while and then departs (purple arrowed lines), and a_2 also waits a while then leaves and comes back (blue arrowed

[1] https://github.com/PathPlanning/Continuous-CBS



(a) (b) (c)

Figure 1: (a) shows the problem where a_1 is taking the action $m_1 = \langle (A,B), d_{AB} \rangle @0$, and a_2 is parking at its goal $w_1 = \langle (B,B), \infty \rangle @0$. (b)(c) shows constraints (red region) placed over the timeline and the pink regions are the collision interval. (b) motion and vertex constraints with a pair of collision-free solutions. (c) shifting constraints.

lines), therefore the existing CCBS implementation using a pair of *vertex and motion constraints* is not complete.

Since these constraints eliminate feasible solutions and violate Claim 1, they do not form a pair of sound constraints. Consequently, the approach is incomplete and may return suboptimal solutions. Appendix B provides a detailed example that CCBS implementation returns a suboptimal solution with a cost significantly larger than a hand-crafted solution.

Sound Pair of Shifting Constraints

With existing efforts proposing post constraints that eliminate wait motions of size larger than a single time instant. We show such instantiation of utilising *vertex constraint* is not sound. So then, is there any way to construct sound constraints using *vertex constraint* that eliminates waiting motions in a range larger than a single time instant? In this section, we present such constraints pair.

Given a conflict $\langle a_i, a_j, w_i@t_i, m_j@t_j \rangle$ between a wait motion w_i on vertex v and a move motion m_j with a *collision interval* $I^c = (t_s^c, t_e^c)$. Given δ that shifts m_j to start at $t_j + \delta$ results a *collision interval* $I^{c\prime} = (t_s^c + \delta, t_e^c + \delta)$. We call $\mathcal{I} = [t_j, t_j + \delta]$ as a shift interval, and then interval $I^o = (t_s^c + \delta, t_e^c)$ is the overlapping of I^c and $I^{c\prime}$. We define a pair of **Shifting Constraints** as:

- $\overline{\langle a_i, v, I^o \rangle}$, forbids a_i occupy v in I^o
- $\overline{\langle a_j, m_j, \mathcal{I} \rangle}$, forbids a_j starts m_j in $\mathcal{I}$.

Figure 1c shows an illustrative example of such constraints. Note that, depending on the choices of δ, the ranges of the constraint intervals vary. The length of collision interval $|I^c| = |I^o| + |\mathcal{I}|$ and δ decides how total length is distributed to vertex constraint and motion constraint, the range length is defined by the range end time minus the range start time. There also exists a maximum range length l_{max}^{mv} of collision interval between a move motion m_j and any wait motion on v, since l_{max}^{mv} is decided by r and the motion.

Theorem 2. *Given a δ,* Shifting Constraints *pair is sound.*

Proof. Give any $\hat{t} \in \mathcal{I}$ and $x = \hat{t} - t_j$, $\hat{I}^c = (t_s^c + x, t_e^c + x)$ is the collision interval if $m_j@\hat{t}$. Since $x \leq \delta$, $t_s^c + x \leq t_s^c + \delta$ and $t_e^c + x \geq t_e^c$. Thus $I^o \subseteq \hat{I}^c$ and a_j starts m_j in $\mathcal{I}$ collides with a_i occupies v within I^o. □

Theorem 3. *Given any* vertex constraint C_v *on* v *(resp. motion constraint* C_m *on* m*) with any constraint interval* I_v *(resp.* I_m*) and its size* $|I_v| \leq l_{max}^{mv}$ *(resp.* $|I_m| \leq l_{max}^{mv}$*), if the interval* I_m *(resp.* I_v*) of the corresponding* motion constraint C_m *on* m *(resp.* vertex constraint C_v *on* v*), violates, or constraint any other time beyond the interval* $\mathcal{I}$ *(resp.* I^o*) defined by,* Shifting Constraints *definition, the pair of constraints eliminates collision-free solution.*

Proof sketch. Given a vertex constraint C_v, if the corresponding motion constraint C_m constrains time t, which is outside of the interval defined by *Shifting Constraints*. There exists a range constrained by C_v, agents occupying v in this range do not collide with agents executing m at t, since the range is out of the maximum collision interval between $m@t$ and v. The same logic applies when given C_m and C_v constrains time outside the interval of the definition of *Shifting Constraints*. The full proof is provided in Appendix C. □

Unfortunately, such constraints still lead to termination failure, and, remember, Theorem 3 demonstrated that vertex and motion constraint pairs beyond the *Shifting Constraints* definition is not sound. If the $\delta > 0$, then I^o cannot eliminate the collision solution that a_j start $m_j@t_j$ and a_i occupy v at t_s^c, therefore the conflict is not resolved by *shift constraints* and leading to infinite expansions. And if $\delta = 0$, then $\mathcal{I} = [t_j, t_j]$, which means, this constraint will suffer a similar infinite expansion problem with the one stated in Lemma 2 and leading to infinite expansions.

Conclusion

In this paper, we demonstrated how CCBS overlooked the fact that the algorithm has to guarantee a finite number of node expansions to claim optimality and completeness. We then discussed how existing implementation attempts to patch the algorithm with a pair of *vertex and motion constraint*, but in an incomplete manner. But following their attempts, we proposed *shifting constraints*, a sound pair of *vertex constraint* and *motion constraint*, and proved that any constraint intervals beyond the definition of *shifting constraints* leading to the elimination of collision-free solution. However, such constraints still fail to terminate the search.

All these efforts demonstrated that patching the algorithm is non-trivial. Such a patch requires the algorithm to explore a finite number of nodes to push the lower bound and has a finite number of nodes on a given cost. Before continue pushing the area forward, we have to stop, as further progress necessitates either developing new techniques or proving the impossibility of such a patch. Since the optimality guarantee has not been achieved yet, we consider extending CCBS towards a relaxed guarantee algorithm or efficiency-focused algorithm to be valuable.

References

Andreychuk, A.; Yakovlev, K.; Surynek, P.; Atzmon, D.; and Stern, R. 2022. Multi-agent pathfinding with continuous time. *Artificial Intelligence*, 305: 103662.

Andreychuk, A.; Yakovlev, K. S.; Boyarski, E.; and Stern, R. 2021. Improving Continuous-time Conflict Based Search. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 11220–11227. AAAI Press.

Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; and Zhou, N.-F. 2018. Robust multi-agent path finding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 9, 2–9.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Betzalel, O.; Tolpin, D.; and Shimony, E. 2015. Icbs: The improved conflict-based search algorithm for multi-agent pathfinding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 6, 223–225.

Kulhan, M.; and Surynek, P. 2023. Multi-Agent Pathfinding for Indoor Quadcopters: A Platform for Testing Planning-Acting Loop. volume 1, 221 – 228. Cited by: 0; All Open Access, Hybrid Gold Open Access.

Li, J.; Harabor, D.; Stuckey, P. J.; Ma, H.; Gange, G.; and Koenig, S. 2021. Pairwise symmetry reasoning for multi-agent path finding search. *Artificial Intelligence*, 301: 103574.

Phillips, M.; and Likhachev, M. 2011. SIPP: Safe interval path planning for dynamic environments. In *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*, 5628–5635. IEEE.

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

Shen, B.; Chen, Z.; Li, J.; Cheema, M. A.; Harabor, D. D.; and Stuckey, P. J. 2023. Beyond pairwise reasoning in multi-agent path finding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 33, 384–392.

Walker, T. T.; Sturtevant, N. R.; and Felner, A. 2020. Generalized and Sub-Optimal Bipartite Constraints for Conflict-Based Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05): 7277–7284.

Walker, T. T.; Sturtevant, N. R.; and Felner, A. 2024. Clique Analysis and Bypassing in Continuous-Time Conflict-Based Search. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, 2540–2542. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400704864.

Walker, T. T.; Sturtevant, N. R.; Felner, A.; Zhang, H.; Li, J.; and Kumar, T. K. S. 2021. Conflict-Based Increasing Cost Search. *Proceedings of the International Conference on Automated Planning and Scheduling*, 31(1): 385–395.

Yakovlev, K.; Andreychuk, A.; and Stern, R. 2024. Optimal and Bounded Suboptimal Any-Angle Multi-agent Pathfinding (Extended Abstract). volume 17, 295 – 296. Cited by: 0; All Open Access, Bronze Open Access.

Appendix A. Computing Collision Interval

In this section we give a detailed example on how collision interval is computed in implementation. Following the conceptual example in Figure 2, if agent a_i waits at a vertex O infinitely with wait action $w_i@t$, and another agent a_j moves from vertex M to N with action $m@t_0$, $t_0 \gg t$. The *collision interval* between $w@t$ and $m@t_0$, also the time range of the range constraint for a_i, is $(t_0 + ||(M,H)||_2 - ||(P_1,H)||_2, t_1)$, t_0 is the moving action start time, P_1 is the location point on edge (M,N) where a_j starts overlapping with a_i if its centre is on P_1, and t_1 is the moving action end time.

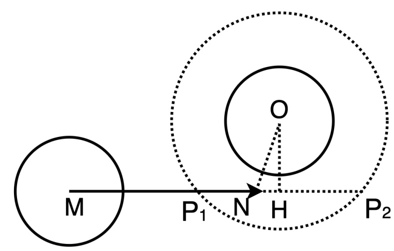


Figure 2: The moving action is $M@t_0 \rightarrow N@t_1$, and the waiting agent is parking at vertex O. The radius of the dashed circle shown in the figure is $2 * r$. P_1 is the point where two agents start to collide, and P_2 is the point the moving agent leaves the collision region if it keeps moving along the extension of (M,N) segment. (O,H) is the perpendicular bisector of (P_1,P_2).

Appendix B. Detailed example on sub-optimal Instances

In this section we will provide an example where CCBS implementation fails to find the optimal solution. For simplicity we will use $v@\tau \rightarrow v'@\tau'$ to denote an action $\langle m.D, (v,v')\rangle@\tau$ where $\tau' = \tau + m.D$, and a plan for an agent can be denoted by changing these notations together.

With a concrete example shown in Fig3, the first conflict is between a_1 and a_2 is $m_1 = \langle (A,C), 4.9\rangle@0$ and $\langle (C,C), \infty\rangle@3.1$. The constraints that CCBS implementation generated are $\overline{\langle a_1, m_1, [0,\infty)\rangle}$ and $\overline{\langle a_2, C, [3.9, 4.9)\rangle}$. Such a pair of constraints eliminate the solution of a_1 wait a while then depart, and a_2 also wait a while then leave. The final solution from CCBS has a solution with a cost 34.9852:

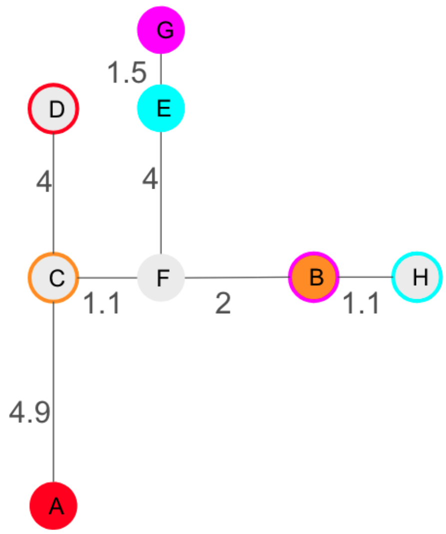


Figure 3: In this Example, $a_1 : (A \to D)$, $a_2 : (B \to C)$, $a_3 : (E \to H)$, $a_4 : (G \to B)$

$$a_1 : A@0 \to C@4.9 \quad \to D@8.9$$
$$a_2 : B@0 \to B@3.2 \quad \to F@5.2 \quad \to F@5.2142 \to C@6.3142$$
$$a_3 : E@0 \to E@2.6284 \quad \to F@6.6284 \quad \to B@8.6284 \to H@9.7284$$
$$a_4 : G@0 \to G@2.1284 \quad \to E@3.6284 \quad \to E@4.0426 \to F@8.0426 \quad \to B@10.0426$$

We also handcrafted a solution, which has a smaller cost 34.1. As it's handcrafted, the gap between agents is much larger than necessary, optimal cost would be smaller than that.

$$a_1 : A@0 \to A@2.1 \quad \to C@7 \quad \to D@11$$
$$a_2 : B@0 \to F@2 \quad \to C@3.1 \quad \to C@5.5 \to$$
$$F@6.6 \to \quad F@7.5 \to \quad C@8.5$$
$$a_3 : E@0 \to F@4 \quad \to B@6 \quad \to H@7.1$$
$$a_4 : G@0 \to E@1.5 \quad \to F@5.5 \quad \to B@7.5$$

In this solution, both a_1 and a_2 are required to wait so that a_2 and a_3 don't need to wait, however, CCBS's range constraint eliminates such a solution.

Appendix C. Theorem 3 Full Proof

In this Section, we give the full proof of Theorem 3 following after the proof of Theorem 2.

Theorem 3. *Given any* vertex constraint C_v *on* v *(resp. motion constraint* C_m *on* m*) with any constraint interval* I_v *(resp.* I_m*) and its size* $|I_v| \leq l^{mv}_{max}$ *(resp.* $|I_m| \leq l^{mv}_{max}$*), if the interval* I_m *(resp.* I_v*) of the corresponding* motion constraint C_m *on* m *(resp.* vertex constraint C_v *on* v*), violates, or constraint any other time beyond the interval* $\mathcal{I}$ *(resp.* I^o*) defined by,* Shifting Constraints *definition, the pair of constraints eliminates collision-free solution.*

Proof. Let C_v be a *vertex constraint* with interval $I_v = (I_v.s, I_v.e)$ where l^{mv}_{max} denotes the maximum collision interval length between move motion m and any wait motion on v.

According to the definition of *Shifting Constraints*, there exists a *shift interval* $\mathcal{I} = [\mathcal{I}.s, \mathcal{I}.e]$ for the *Motion Constraint* C_m. Starting m at $\mathcal{I}.s$ resulting a maximum collision interval $(I_v.e - l^{mv}_{max}, I_v.e)$ with agent occupy v. Here, $\mathcal{I}.e = \mathcal{I}.s + \delta$, where $\delta = l^{mv}_{max} - |I_v|$.

Suppose $\hat{t} \notin \mathcal{I}$ is constrained by C'_m, $(\hat{t}^c_s, \hat{t}^c_e)$ is the maximum collision interval $m@t$ collides with agent occupies v. By definition, $I_v.s$ is also the time move motion $m@\mathcal{I}.e$ starts overlapping with any agent on v.

If $t > \mathcal{I}.e$, then $\hat{t}^c_s > I_v.s$. Since $m@t$, constrained by C'_m, does not collide with an agent occupies v before $\hat{t}^c_s$, it does not collide with an agent occupies v in $(I_v.s, min(\hat{t}^c_s, I_v.e)) \subseteq I_v$, which is constrained by C_v. Thus collision-free solutions are eliminated.

Similarly, if $t < \mathcal{I}.s$, collision free solutions $m@t$ and agent occupies v in $(max(\hat{t}^c_e, I_v.s), I_v.e) \subseteq I_v$ are eliminated.

The same logic applies when given C_m and C_v constrains time outside the interval of the definition of *Shifting Constraints*.

Therefore, constraining time outside the intervals defined by *Shifting Constraints* will eliminate collision-free solutions. □