Revisiting Conflict Based Search with Continuous-Time

Anonymous submission

Abstract

Multi-Agent Path Finding in Continuous Time (MAPF_{R}) extends the classical MAPF problem by allowing agents to operate in continuous time. Conflict-Based Search with Continuous Time (CCBS) is a foundational algorithm for solving MAPF_{R} optimally. In this paper, we revisit the theoretical claims of CCBS and show the algorithm is incomplete, due to an uncountably infinite state space created by continuous wait durations. We then prove that the constraint generation method employed by CCBS inherently leads to incompleteness (i.e., there is no simple fix). For further insight into these issues, we consider two related sub-problems: $\text{MAPF}_{\text{R-NO}}$, which restricts interactions between moving and waiting agents to vertex collisions, and $\text{MAPF}_{\text{R-DT}}$, which restricts wait durations to fixed amounts. In the case of $\text{MAPF}_{\text{R-NO}}$, we employ a simple reduction to show that the problem is solvable, although not by CCBS. In the case of $\text{MAPF}_{\text{R-DT}}$, we show that the problem is optimally solvable including by CCBS. It remains an open question whether MAPF_{R}, which allows arbitrary wait durations, is optimally solvable.

Abstract

Multi-Agent Path Finding in Continuous Time (MAPF_{R}) extends the classical MAPF problem by allowing agents to operate in continuous time. Conflict-Based Search with Continuous Time (CCBS) is a foundational algorithm for solving MAPF_{R} optimally. In this paper, we revisit the theoretical claims of CCBS and show the algorithm is incomplete, due to an uncountably infinite state space created by continuous wait durations. We then prove that the constraint generation method employed by CCBS inherently leads to incompleteness (i.e., there is no simple fix). For further insight into these issues, we consider two related sub-problems: $\text{MAPF}_{\text{R-NO}}$, which restricts interactions between moving and waiting agents to vertex collisions, and $\text{MAPF}_{\text{R-DT}}$, which restricts wait durations to fixed amounts. In the case of $\text{MAPF}_{\text{R-NO}}$, we employ a simple reduction to show that the problem is solvable, although not by CCBS. In the case of $\text{MAPF}_{\text{R-DT}}$, we show that the problem is optimally solvable including by CCBS. It remains an open question whether MAPF_{R}, which allows arbitrary wait durations, is optimally solvable.

1 Introduction

Multi-Agent Path Finding (MAPF) is the problem of planning paths for a set of moving agents from initial states to goals while avoiding collisions. MAPF is gaining increasing attention because of its significance in a wide range of applications settings, both in the physical world such as warehouse management (Wurman, D'Andrea, and Mountz 2008) and robotics (Pradhan, Roy, and Hui 2018), as well as in virtual environments such as computer games (Harabor, Hechenberger, and Jahn 2022). Classical MAPF models (Stern et al. 2019) reduce these problems to a simplified (grid-based, unit-cost) combinatorial core, which is nonetheless NP-hard (Yu and LaValle 2013; Banfi, Basilico, and Amigoni 2017).

Conflict-Based Search(CBS) is currently the most widely used approach for optimally solving Classical MAPF (Sharon et al. 2015; Boyarski et al. 2015; Li et al. 2021; Shen et al. 2023). To tackle the problem CBS uses a two-level approach. At the high level, CBS searches over a Conflict Tree (CT), where each node represents a set of constraints on the actions of the agents. At the low level, it begins with each agent's individual shortest path as a lower bound and resolves conflicts by iteratively identifying colliding agent pairs and imposing a pair of sound constraints to eliminate collisions. Unfortunately, classical MAPF plans are seldom applicable in practice as many physical applications operate in continuous environments, where actions or motions have non-unit duration and their executions are not synchronised.

MAPF_{R} extends the classical MAPF model by generalising the time dimension to real-valued domains, allowing agents to move in *continuous* time and wait for any arbitrary amount of real-valued time duration. Continuous Time CBS (CCBS) (Andreychuk et al. 2022) is claimed to be the first optimal algorithm for solving this problem. The algorithm is widely popular in the literature and several subsequent works provide further improvements (Andreychuk et al. 2021; Walker, Sturtevant, and Felner 2020; Walker et al. 2021; Walker, Sturtevant, and Felner 2024a) and enhancements (Yakovlev, Andreychuk, and Stern 2024; Kulhan and Surynek 2023). In this work, we show that the original constraint definition in CCBS (Andreychuk et al. 2022) leads to two distinct interpretations regarding wait actions: the first interpretation assumes wait durations are specified a priori, the second assumes durations are computed dynamically. We demonstrate that, under *both* of these interpretations, CCBS is either unsound, incomplete, or both. The implications of our findings affect numerous recent works, all of which implicitly rely on the correctness of the CCBS algorithm. It is an open question whether there exists an alternative constraint-based reasoning model which can resolve these issues.

To better understand the technical challenges in this area we investigate two closely related problems. The first problem, which we call MAPF$_{\text{R-NO}}$, limits the possible interactions between moving and waiting agents to vertex conflicts only. With this additional restriction we can derive a reduction to the Pebble Motion Problem (Kornhauser, Miller, and Spirakis 1984) and thus establish solvability guarantees for a special case of the general MAPF$_{\text{R}}$ problem, but not for CCBS. The second problem, which we call MAPF$_{\text{R-DT}}$, imposes a unit duration for all wait actions. With this additional restriction we show that the problem is optimally-solvable, including by CCBS. Our results emphasise that there currently exists no known algorithm for MAPF$_{\text{R}}$ which is truly sound and complete. This stands in contrast to Classical MAPF, whose solvability was well understood even in the early stages of research (Kornhauser, Miller, and Spirakis 1984).

2 MAPF$_{\text{R}}$ Problem Definition

We use the continuous-time MAPF problem definition defined by Andreychuk et al. (2022), which takes a weighted graph $G = \{V, E\}$ and a set of k agents $\{a_1, ..., a_k\}$ as input. Every vertex v in G, maps to a coordinate in a metric space $\mathcal{M}$, and every edge connects between two vertices. Each agent has a unique start, $s_i \in V$ and goal vertex $g_i \in V$ with no two agents sharing the same start or goal vertices: $\forall i \in \{1, ..., k\} \nexists j \in \{1, ..., k\} \setminus i : s_i = s_j \vee g_i = g_j$. For agent a_i a plan π_i, with length l, is a sequence of motions m_i^j with start times τ_i^j, such that $\{m_i^1 @ \tau_i^1, ..., m_i^l @ \tau_i^l\}$. Each motion m consists of $\langle m.\varphi, m.D \rangle$:

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

The motion function $m.\varphi(t)$ enables the agent to move at a non-constant speed and follow an arbitrary geometric curve. However, in this paper, following the original CCBS assumption, all agents move in a straight line at a constant speed of 1 without any kino-dynamics. Therefore, we will use $d(v, v')$ to denote the length of the edge (v, v'), which also indicates the cost of the moving motion along this edge. Conveniently, the motion function is a linear function that outputs the coordinate of the agent by executing motion m for duration t.

In general, a motion function comprise of either a moving motion or a waiting motion. For a moving motion, $m.\varphi(0)$ returns the current vertex v and $m.\varphi(m.D)$ returns the next vertex v'. A waiting motion happens when $v' = v$ and $m.\varphi(t)$ returns the coordinate of v for the entirety of $[0, m.D]$. In the MAPF$_{\text{R}}$ problem, we assume a finite set of move motions, corresponding to the finite number of edges in G, with the motion duration being fixed and represented by the edge weights. However, the set of wait motions is assumed to be uncountably infinite, as agents can wait for any positive real-valued amount of time, leading to an infinite range of possible wait durations. In a valid plan π_i, the origin vertex v^n of $m_i^n : (v^n, v'^n)$ must coincide with the target vertex v'^{n-1} of the previous motion $m_i^{n-1} : (v^{n-1}, v'^{n-1})$. Similarly, the starting time τ_i^n of motion m_i^n is always equal to the finish time $\tau_i^{n-1} + m_i^{n-1}.D$ of the previous motion m_i^{n-1}, where $m_i^{n-1}.D$ denotes the duration of the previous motion.

A feasible solution to a MAPF$_{\text{R}}$ problem consists of a set of plans $\Pi = \{\pi_1, ..., \pi_k\}$ where each planned motion of every 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 occurs when a collision detection function $InCollision(m_i @ \tau_i, m_j @ \tau_j)$ returns $true$. This indicates that there exists a time t at which the locations of the two agents overlap.

Following the original CCBS assumption that all agents are circular with radius r, the function $InCollision(m_i @ \tau_i, m_j @ \tau_j)$ returns true if there exists a time t such that the distance between the centres 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 the distance between the two agents is exactly $2r$, and t_e is the second time when their distance is $2r$, or when one of the motions finishes. We refer to the interval (t_s, t_e) as the **collision interval**. The objective is to compute a collision-free plan that minimises the *Sum of Individual (path) Cost* (SIC), where $SIC = \sum_{i \in [1,k]} et_i$ with et_i indicating the end time of the last motion in path π_i.

3 Background

CBS with Continuous Time

CCBS is an extension of the original optimal MAPF solver CBS, specifically designed to solve the MAPF$_{\text{R}}$ problem optimally. CCBS is a two-level tree search algorithm, where the high-level search is a best-first search on a binary *Constraint Tree* (CT). Each CT node $N = \{N.constraints, N.\Pi, N.g\}$ contains:

- $N.constraints$, a set of conflict-resolving constraints,
- $N.\Pi$, a solution that satisfies these constraints, and
- $N.g$, the sum of the individual costs (SIC) of $N.\Pi$.

At the low level of CCBS, the algorithm 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 these paths. The process begins by generating a root CT node with no constraints ($N.constraints = \phi$) and a solution consisting of individual shortest plans for all agents. It then iteratively selects a node N with the smallest $N.g$. If N contains no conflicts, it is deemed the goal node. Otherwise, a conflict from $N.\Pi$, such as $\langle a_i, a_j, m_i @ \tau_i, m_j @ \tau_j \rangle$, is selected for resolution.

CCBS then generates two child CT nodes N_i and N_j, each appending a new conflict-resolving constraint C based on $N.constraints$, so that:

- $N_i.constraints = N.constraints \cup C_i$
- $N_j.constraints = N.constraints \cup C_j$

The paper proposes a pair of **Motion Constraints** to resolve the conflict:

- $C_i = \overline{\langle a_i, m_i, I_i \rangle}$
- $C_j = \overline{\langle a_j, m_j, I_j \rangle}$.

An **unsafe interval** $I_i : [t_i, t'_i)$ (resp. $I_j : [t_j, t'_j)$) for agent a_i (resp. a_j) is a time period during which if the starting time τ_i (resp. τ_j) of motion m_i (resp. m_j) falls within I_i (resp. I_j), it will result in a conflict with $m_j@\tau_j$ (resp. $m_i@\tau_i$). A **motion constraint** $C_i : \overline{\langle a_i, m_i, I_i \rangle}$ (resp. $C_j : \overline{\langle a_j, m_j, I_j \rangle}$) indicates that agent a_i (resp. a_j) cannot start motion m_i (resp. m_j) within the *unsafe interval* I_i (resp. I_j).

The Claims of CCBS

We begin with the following definition from CCBS (Andreychuk et al. 2022):

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 hold.*

CCBS then 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, CCBS proved Lemma 2.

lemma 2. *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 states that no collision-free solution should be eliminated, it holds because N_1 and N_2 are constrained by a sound pair of constraints (Definition 1 and Lemma 1). The second condition states that, during the search, node cost can only monotonically increase, it 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. The first condition ensures that any valid solution can be reached through one of the un-expanded CT nodes. Since CCBS conducts a best-first search over the CT and prioritises expanding CT nodes with the lowest cost, the second condition enables CCBS to guarantee finding an optimal MAPF$_R$ solution. Together, these two conditions aim to ensure the completeness of CCBS, meaning that if an optimal solution exists, it will be found.

Examining the Theoretical Proof Limitations of CCBS

It is important to note that there is ambiguity in the MAPF community regarding the description of the constraints used to resolve conflicts between a moving agent and a waiting agent. This ambiguity gives rise to two different interpretation of the constraint applied to the waiting agent. For the sake of clarity, we assign distinct names to each interpretation:

- *Waiting durations are determined a priori*: In this model, the respective constraint, termed a *motion constraint*, forbids the start of an action (whether movement or wait) within the constraint interval. We analyse CCBS with motion constraints in Section 4.
- *Waiting durations are computed dynamically*: In this model, the respective constraint, termed a *vertex range constraint*, forbids any waiting motion from overlapping with the constraint interval, regardless of when it starts. We analyse CCBS with vertex range constraints in Section 5.

In the following two sections, we detail the differences between these two interpretations and formally demonstrate that they are either unsound, incomplete, or both.

4 Interpretation 1: Motion Constraints

This interpretation originates from strictly adhering to the problem and conflict definitions as stated in the original CCBS paper. Under this interpretation wait and move motions are modelled in the same way and their durations are specified in advance.

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

Although a sound pair of motion constraints ensures that no collision-free solutions are eliminated during splitting, this alone is not sufficient for CCBS to find an optimal solution. In particular, recall that a motion is defined by both a motion function and its duration. That means each real-valued waiting duration corresponds to a distinct wait motion. Thus, each constraint only forbids a single wait motion at vertex v_1. However, there are uncountably many wait motions at v_1 that can collide with the conflicting motion $m_j@t_2$ and it becomes infeasible to resolve such conflicts through motion constraints alone:

lemma 3. *Let N' with a motion constraint $\overline{\langle a_i, w_i, [t_1, t_1'))}$ be a child node of N with a_i and a_j collide in the collision interval (t_1^c, t_2^c). The constraint eliminates solutions where w_i conflicts with $m_j@t_2$, but permits an infinite number of solutions that have an alternative wait motion $w_i' = \langle(v_1, v_1), w_i.D + \delta\rangle$ which 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, there is an uncountable infinite number of choices on δ. This leads to an 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 any wait motion w_i' that executes at any time $t' \in [t_1, t_1')$, it will collide with $m_j@t_2$, since the wait interval—of a_i waiting 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 collides with $m_j@t_2$. □

A main consequence of Lemma 3 is that CCBS must explore an infinite number of nodes and resolve an infinite number of conflicts, in order to advance the lower bound and ultimately identify an optimal solution, if one exists. This leads to non-termination, making the algorithm incomplete under these conditions. More formally:

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, has to explore an infinite number of nodes to find N^* if N has conflicts between a pair of wait and move motion.*

Proof. CCBS searches in a best-first manner on solution cost g. To terminate the search and find an optimal solution node N^*, it must eliminate any collision solution with a cost smaller than $c^* = N.g + \Delta$ in its frontier. Assuming a_i with wait motion $w_i@t_1$ is colliding with a_j with move motion $m_j@t_2$, Lemma 3 shows that one child node of N permits an infinite number of solutions, each with a $w_i' \neq w_i$ colliding with $m_j@t_2$. As long as $\Delta > 0$, there exists an infinite number of choices of $\delta \in [0, \Delta]$. Using a wait motion $w_i' = \langle w_i.\varphi, w_i.D + \delta\rangle$ to replace the w_i in the solution of N, the resulting solution has cost increased by at most δ. Therefore, we have an infinite number of collision solutions that have costs smaller than $N.g + \Delta$. As a pair of *motion constraints* only removes collision solutions with one choice of wait motion duration, the algorithm requires an infinite number of expansions to remove all collision solutions with a cost smaller than $c^* = N.g + \Delta$ in its frontier. □

To sum up, CCBS may fail to terminate and return a solution when it strictly resolves conflicts using *motion constraints*. While real-world computers operate over finite-precision representations rather than true real numbers, CCBS would still need to explore every representable wait duration that could potentially resolve a conflict. This is nearly impossible, as each expansion can exponentially increase the remaining search space, effectively doubling the workload with every new branch.

5 Interpretation 2: Vertex Range Constraint

This interpretation is derived from Example *4.1.3. SIPP for the low-level of CCBS*, which appears in the CCBS paper (Andreychuk et al. 2022). In this model wait actions have dynamically computed wait durations and collisions with waiting agents are resolved by generating a corresponding *vertex range constraint*. This approach is commonly adopted within the community, as evidenced by its use in many subsequent works(Walker, Sturtevant, and Felner 2024a; Combrink, Roselli, and Fabian 2025; Yakovlev, Andreychuk, and Stern 2024), including in the original implementation of CCBS[1]. While vertex range constraints often lead to empirical termination, a closer examination reveals that this model is also incomplete, although the problem arises in a different manner than for motion constraints (which we discuss in Section 4). Concretely, a vertex range constraint forbids the existence of an agent a_i on a vertex v within a given *time range* $[t, t')$. We use $\overline{\langle a, v, (t, t')\rangle}$ to denote such a constraint. Whereas a motion constraint forbids the starting of a single wait motion (with a known duration), a vertex range constraint forbids any wait motion $\langle(v, v), D\rangle$ at any time τ as long as $[\tau, \tau + D]$ overlaps with (t, t').

Consider the same CT node N (appearing in Section 4) with the same conflict $c = \langle a_i, a_j, w_i@t_1, m_j@t_2\rangle$. Under this interpretation CCBS will introduce a pair of *vertex range and motion constraints*:

- $\overline{\langle a_i, v_i, (t_s, t_e)\rangle}$, a vertex range constraint which forbids agent a_i from using v_i, and
- $\overline{\langle a_j, m_j, [t_2, t_2')\rangle}$, a motion constraint which forbids agent a_j from executing motion m_j.

In this constraint pair, (t_s, t_e) is the collision interval between $w_i@t_1$ and $m_j@t_2$, $[t_2, t_2')$ is the unsafe interval for a_j. Executing m_j at any time in this range must collide with $w_i@t_1$ for agent a_i. A detailed example in Appendix A shows how CCBS computes the collision interval, closely following the approach derived by (Walker and Sturtevant 2019). We also note that there is a subtle variation for the constraining time range used in the vertex range constraint. According to the unsafe interval definition (Definition 3) and constraint notation (Lemma 1) from the original paper, the constraining interval always begins at the start time of the conflict move, i.e. $\overline{\langle a_i, v_i, [t_1, t_e)\rangle}$. However, the interval $[t_1, t_e)$ is strictly larger than $[t_s, t_e)$, which we will later show already excludes some valid solutions. Therefore, using $[t_1, t_e)$ as the constraint interval would also eliminate feasible solutions.

Elimination of Feasible Solutions

We observe that the combination of a vertex range and motion constraints pair is not sound, as it eliminates feasible solutions.

An example illustrating the problem is shown in Figure 1. Figure 1a shows the spatial graph, while Figure 1b illustrates the constraints and agents' movements over time, with time flowing from top to bottom. In this example. agent a_1 needs

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

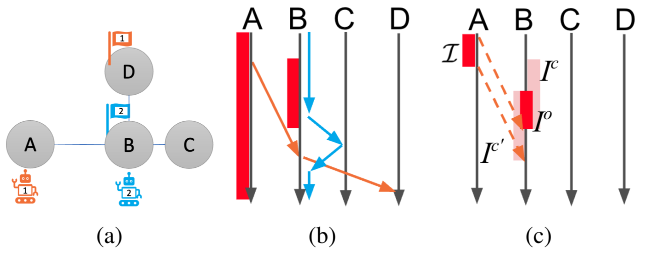


Figure 1: (a) Agent a_1 performs motion $m_1 = \langle (A,B), d_{AB}\rangle@0$, while agent a_2 has stopped at its goal $\langle (B,B),\infty\rangle@0$, causing a collision. (b) Motion and vertex range constraints with a pair of collision-free solutions. (c) Shifting constraints. Constraints are represented by red regions and collision intervals by pink regions.

to move from location A to B and then to D while agent a_2 stops at its goal at location B, causing a collision. The existing implementation generates two constraints:

1. The first constraint, $\overline{\langle a_1, m_1, [0,\infty)\rangle}$ prevents agent a_1 from moving to B (shown as red region on A, Fig1b).
2. The second constraint, $\overline{\langle a_2, B, (t_2, t_3)\rangle}$, prevents agent a_2 from occupying B within the time range of (t_2, t_3) (shown as red region on B, Fig1b).

These constraints eliminate the collision-free solution where agent a_1 waits for a while and then departs (represented by the orange arrowed lines), and agent a_2 also waits before leaving and returning (blue arrowed lines). This example empirically proves that a pair of *vertex and motion constraints*, is not complete. In Appendix B, we provide a concrete example where eliminating such a situation leads to a suboptimal solution.

6 Limitation of Vertex Range Constraint

In this section we propose a method to construct a pair of motion and vertex range constraints, which will not remove collision-free solutions. We call these a *sound pair of shifting constraints*. The ability to safely eliminate more than one single in-collision solution at a time brings us a step closer to resolving the theoretical difficulties in CCBS. However, we also discover some non-trivial properties of shifting constraints which make patching of the CCBS algorithm extremely difficult (i.e., we think a fix of this type is unlikely).

Sound Pair of Shifting Constraints

Given a move motion $m_j@t_j$, it collides with any agent occupies vertex v within the *collision interval* of $I^c = (t^c_s, t^c_e)$. If we shift or delay the move motion m_j to start at $t_j + \delta$, it results a new *collision interval* $I^{c'} = (t^c_s + \delta, t^c_e + \delta)$ with $w_i@t_i$, as shown in Figure 1c. We call $\mathcal{I} = [t_j, t_j + \delta]$ a *shift interval*, and then define an *overlapping interval* $I^o = (t^c_s + \delta, t^c_e) = I^c \cap I^{c'}$. The *overlapping interval* I^o is a subset of or equal to a conflict interval I^x between any motion $m_j@(t_j + d), d \in [0, \delta]$ and $w_i@t_i$, thus, any motion $m_j@(t_j+d), d \in [0,\delta]$ conflicts with another agent exists on v in range I^o. We thus define a pair of **Shifting Constraints** between m_j and any agent a_i on v as:

- $\overline{\langle a_i, v, I^o\rangle}$, which forbids a_i from waiting at v in the interval I^o
- $\overline{\langle a_j, m_j, \mathcal{I}\rangle}$, which forbids a_j from starting motion m_j in the interval $\mathcal{I}$.

Note that, by referring to Figure 1c and the definition of *shifting constraints*, it is obvious that the choice of the range of $\mathcal{I}$ decides the range of I^o, and $|I^c| = |I^o| + |\mathcal{I}|$.

theorem 2. *A pair of* Shifting Constraints *is sound as any motions violating both constraints must be in collision.*

Proof. Following the definition of *Shifting Constraints*, given any start time $\hat{t} \in \mathcal{I}$ for motion m_j. The collision interval that $m_j@\hat{t}$ collides with agent staying at v is $\hat{I}^c = I_c + x = (t^c_s + x, t^c_e + x)$, where $x = \hat{t} - t_j$ is the difference between the new and original move motion start time. Since $0 \le x \le \delta$, $I^o = (t^c_s + \delta, t^c_e)$, $t^c_s + x < t^c_s + \delta$ and $t^c_e + x > t^c_e$, thus I^o is always a subset of *collision interval* $\hat{I}^c$, indicating that any agent at the vertex v in the range I^o collides with any move motion $m_j@\hat{t}$. □

Next, we show that given a constraint interval on the motion constraint (equiv. vertex range constraint), if the range of the corresponding vertex range constraint (equiv. motion constraint) goes beyond the definition of a pair of *shifting constraints*, collision-free solutions are eliminated.

Property 1. *Given a motion m_j (equiv. vertex v) of agent a_j (equiv. a_i) constrained in an arbitrary $\mathcal{I}$ (equiv. I^o), and the corresponding vertex v (equiv. motion m_j) for agent a_i (equiv. a_j) is constrained in interval $I^{o'}$ (equiv. $\mathcal{I}'$). If there exists a time $\hat{t} \in I^{o'}$ (equiv. $\hat{t} \in \mathcal{I}'$), but $\hat{t} \notin I^o$ (equiv. $\hat{t} \notin \mathcal{I}$), meaning $\hat{t}$ does not exist in the constraint interval defined by shifting constraints, collision-free solutions are eliminated.*

Proof. For the case of given motion m_j constrained by $\mathcal{I}$, to have collision-free solutions be not eliminated, we must guarantee that $\hat{t}$ is not included in any collision interval resulting from the motion m_j starting at any time $t'_j \in \mathcal{I}$. However, the definition of *shifting constraint* shows that, $\hat{t} \notin I^o$ means that $\hat{t}$ is either earlier than the collision interval start time when move motion m_j is started at a time close or equal to $t_j + \delta$, or is later than the collision interval end time when move motion m_j is started at a time close or equal to t_j. Thus collision-free solution is eliminated by such a pair of constraints. The same logic applies when given the vertex range constraint interval I_o on v and there exists a time $\hat{t} \in \mathcal{I}'$ and $\hat{t} \notin \mathcal{I}$. □

We have demonstrated that *shifting constraints* are a pair of sound constraints, and any pair of motion and vertex constraints beyond this definition will eliminate collision-free solutions. Unfortunately, we now demonstrate that such pair of constraints still leads to termination failure.

theorem 3. *Resolving conflicts between move motion and wait motion using sound pairs of* shifting constraints *leads to termination failure.*

Proof. Following the definition of *shifting constraints*, if $\delta > 0$, in other words the size of the constraint interval $\mathcal{I}$ on m_j is not 0, then the constraint interval I^o on vertex v cannot eliminate the collision solution that a_j starts $m_j@t_j$ and a_i occupies v at t^c_s, as $t^c_s \notin I^o$. Therefore, the conflict is not resolved in the child node with the vertex range constraint. If the $\delta = 0$, then the constraint interval $\mathcal{I}$ on m_j becomes a single real number (constraint on a single time) that does not eliminate the conflict between any $m_j@(t_j + d), d \in (0, |I_c|]$ and an agent occupying vertex v. As the amount of $d \in (0, I_c]$ is infinite, it suffers a similar infinite expansion problem to the one stated in Lemma 3, leading to infinite expansions. Therefore, *shift constraints* always have the conflict remaining in one of their child nodes and lead to infinite expansions. □

Theorem 3 and Property 1 show that patching the termination failure issue in CCBS is non-trivial, and even hints at fundamental limitations of relying on pairs of motion and vertex range constraints to achieve completeness. Such results underscore the necessity of exploring an entirely different approach to tackle the incompleteness challenge in CCBS, or to solve the MAPF$_\text{R}$ problem optimally in an entirely different way.

7 MAPF$_\text{R}$ with Discrete Time

Upon its introduction the MAPF$_\text{R}$ problem was defined to support arbitrary-duration wait motions (Walker, Sturtevant, and Felner 2018). However, the considered algorithms (optimal ICTS and bounded sub-optimal ϵ-ICTS and ω-ICTS) restrict wait actions to a single unit-time duration; presumably for simplicity and under the implicit assumption that there is no loss of generality. Subsequent works (e.g., (Walker, Sturtevant, and Felner 2020; Walker et al. 2021)) that consider optimal solving adhere to the original MAPF$_\text{R}$ definition but continue to adopt wait actions with unit durations. Other works (e.g., (Surynek 2021; Walker, Sturtevant, and Felner 2024a)) also adhere to the original MAPF$_\text{R}$ definition but without introducing wait-time restrictions. This has naturally caused some confusion in the community. Andreychuk et al. (Andreychuk et al. 2022) attempted to clarify these differences by presenting a comparative table of existing MAPF$_\text{R}$ solvers, however the relationship between the two problem models is not discussed.

In this section we attempt to clarify the situation by introducing MAPF$_\text{R}$-Discrete Time (MAPF$_\text{R-DT}$ for short), a variant of MAPF$_\text{R}$ which restricts all wait durations to a single unit-time interval. We show that the state-space of MAPF$_\text{R-DT}$ remains countable, unlike MAPF$_\text{R}$. We also show that MAPF$_\text{R-DT}$ is optimally solvable, including by CCBS. This result has direct implications for a variety of works which assume the wait-time restriction can be adopted without loss of generality. Our results show this assumption is false.

lemma 4. *The Search Space of MAPF$_\text{R-DT}$ is countable, i.e. given a time bound, the number of solutions (feasible and infeasible) for MAPF$_\text{R-DT}$ is countable.*

Proof. For every MAPF$_\text{R-DT}$ instance, an agent's motion at each step can either be move or wait. Let $G = (V, E)$ be the underlying graph. Then, for each agent:

- $|E|$ corresponds to the number of possible move actions between edges.
- $|V|$ corresponds to the number of possible wait actions on vertices.

Hence, the total number of distinct motion actions for an agent is finite, $|E| + |V|$ for any MAPF$_\text{R-DT}$ instance. Each agent's path is a concatenation of motions, which can be represented as $[E \cup V]*$, following regular expression syntax, where:

- $E = \{e_1, e_2, ...e_m\}$ denotes the set of all move motions in the graph; and
- $V = \{v_1, v_2, ...v_n\}$ denotes the set of all wait motions in the graph.

Since any regular expression is countable, each agent's possible path is also countable. Furthermore, each motion has an associated cost comprised of the following:

- Each move action e_j has a fixed cost $e_j.D$, and;
- The wait action, w_v, has a fixed cost, typically 1 unit cost per wait.

Therefore, the total cost of a path is a finite sum of fixed-cost components: $\sum_{j=1}^{m} n_j \cdot e_j.D + n_w \cdot w$ where n_j is the number of times edge e_j appeared in the path, and n_w is the number of wait actions. Since all constants are fixed and the multipliers range over natural numbers, the set of possible path costs is countable. The full search space is a cross-product of all agents' possible paths, which is the product of a finite number of countable sets, since the number of agents is finite for any MAPF$_\text{R}$ problem. Thus, the search space for any MAPF$_\text{R-DT}$ instance is countable. □

theorem 4. *CCBS is sound, solution complete, and guaranteed to return an optimal solution for the MAPF$_\text{R-DT}$ problem, if one exists.*

Proof. The two conditions required for the correctness of the CCBS framework in Lemma 2 hold in the context of MAPF$_\text{R-DT}$. Furthermore, from Lemma 4, we know that the solution space for any MAPF$_\text{R-DT}$ instance is countable. Since all motion costs are positive and fixed, and the number of agents is finite, this implies that the number of valid (collision-free) solutions under any given cost bound is also finite. Therefore, if a solution exists, there exists a finite subset of the search space bounded by the cost of the optimal collision-free solution. CCBS resolves collisions incrementally by adding pairwise constraints that eliminate at least one colliding solution in each iteration. Because the number of such candidate solutions is finite, the algorithm must eventually converge to a collision-free solution if one exists. *Thus, if a solution exists, CCBS is solution complete, i.e. it will always find a solution if one exists.* Moreover, CCBS explores the solution space in a best-first manner based on path cost. This ensures that the first collision-free solution it encounters is the one with the lowest possible cost. *Thus,* *CCBS always returns the least-cost valid solution, if one exists.* Finally since CCBS is sound (for $\text{MAPF}_{\text{R-DT}}$), every returned solution is verified to be both collision-free and adherent to all imposed constraints. □

We have demonstrated that the $\text{MAPF}_{\text{R-DT}}$ problem can be effectively solved using enumerative algorithms such as CBS. However, as per Theorem 1, applying such enumerative methods to the original MAPF_{R} model leads to infinite expansions due to its continuous temporal domain. This raises an important open question: *how can we solve the $MAPF_R$ problem efficiently, or even determine whether it is solvable in the general case?*

8 Toward Solvability for MAPF_{R}

It is unclear whether theoretical guarantees which hold for $\text{MAPF}_{\text{R-DT}}$ also hold for MAPF_{R}. This is due to several generalisations introduced in MAPF_{R}, including continuous temporal reasoning and more complex spatial interactions between agents. As a result, standard approaches for verifying solvability—such as running a polynomial-time solver prior to invoking CBS—are no longer applicable. Currently, several suboptimal solvers have been proposed for MAPF_{R} (Park et al. 2023; Zou and Borst 2024); however, these either lack completeness guarantees or rely on the completeness of CCBS, which—as discussed—applies only to the $\text{MAPF}_{\text{R-DT}}$ variant. Consequently, we currently do not have an efficient suboptimal method for verifying the solvability of a MAPF_{R} instance.

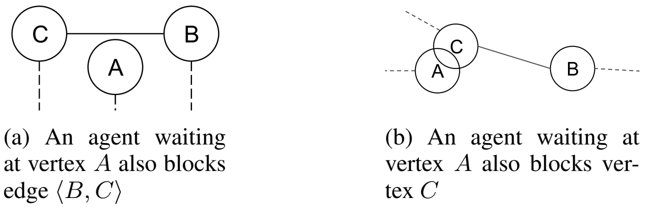
As demonstrated above, the primary challenge in solving MAPF_{R} lies in handling continuous wait durations. One might attempt to sidestep this issue using a reduction similar to the one used in classical MAPF—namely, by having agents wait until the moving agent reaches its destination. However, MAPF_{R} introduces additional complexity not present in classical MAPF: agents are modelled with physical dimensions (e.g., circular agents with a given radius). This physical embodiment fundamentally changes how waiting behaviour affects the environment. In discrete MAPF, a stationary agent does not impede the traversability of adjacent vertices or edges. By contrast, in MAPF_{R}, a waiting agent can physically obstruct nearby paths, depending on its position and size (see Figure 2). This spatial interference significantly complicates conflict resolution and motion planning.

We define two types of spatial interference caused by agents waiting in MAPF_{R}:

- *Edge Overlapping*: This occurs when a waiting agent blocks the traversal of an edge due to its physical presence. Formally, Let agents a_i and a_j be such that:
  - a_i is waiting at a vertex v_k, represented as $w_i = \langle e_k.D, v_k \rangle$; and
  - a_j is moving on the edge e_k, represented as $m_j = \langle e_k.D, e_i \rangle$,

  where $e_k.D$ is the duration to traverse edge e_k. We say that vertex v_k is *Edge Overlapping* with edge e_k, if $InCollision(w_i@t, m_j@t) = true$ for some time t. (Fig. 2a).
- *Vertex Overlapping*: This occurs when the physical sizes of two waiting agents cause them to collide, even if they are located at distinct vertices. Let agents a_i and a_j be such that:
  - a_i is waiting at a vertex v_{k1}, represented as $w_i = \langle I, v_{k1} \rangle$; and
  - a_j is waiting at another vertex v_{k2}, represented as $w_j = \langle I, v_{k2} \rangle$,

  for some non-zero duration I. We say that vertex v_{k1} is *Vertex Overlapping* with vertex v_{k2}, if $InCollision(w_i@t, w_j@t) = true$ for some time t (Fig. 2b).



(a) An agent waiting at vertex A also blocks edge $\langle B, C \rangle$

(b) An agent waiting at vertex A also blocks vertex C

Figure 2: Example of a waiting agent interfering with non-adjacent edges or vertices

We now define a special subclass of MAPF_{R}, called MAPF_{R}-Non-Overlapping($\text{MAPF}_{\text{R-NO}}$), which excludes the complications caused by overlapping waiting agents. Formally, a MAPF_{R} instance belongs to $\text{MAPF}_{\text{R-NO}}$ if and only if no vertex has an edge overlapping with any edge, and no vertex overlaps with any other vertex.

theorem 5. *The non-optimal version of $MAPF_{R\text{-}NO}$ is solvable in polynomial time.*

Proof. From a reduction perspective, $\text{MAPF}_{\text{R-NO}}$ is equivalent to MAPF, therefore, the same techniques apply. We thus proceed by way of the Pebble Motion Problem. Details of this approach are described in (Kornhauser, Miller, and Spirakis 1984) and (Röger and Helmert 2012). □

Although the non-optimal version of $\text{MAPF}_{\text{R-NO}}$ is complete and can be solved in polynomial time, applying CCBS to find optimal solutions faces the same challenges as with general MAPF_{R}, due to the presence of arbitrary wait durations in $\text{MAPF}_{\text{R-NO}}$. As a result, the question of optimal solvability for $\text{MAPF}_{\text{R-NO}}$ remains open, alongside the broader unresolved issue of MAPF_{R} 's general solvability.

Essential Conditions for Establishing Optimality

Mapping the MAPF problem to a tree search framework is one of the most popular approaches in the field. However, without careful problem modelling, tree search algorithms risk losing their guarantees of completeness and optimality. Ensuring completeness in a tree search algorithm requires satisfying two fundamental conditions:

1. The state space must be countable (either finite or countably infinite).
2. All step costs must be non-negative.

Following the CBS structure, for a given CT node N, the state is defined as the set of all solutions that satisfy $N.constraints$. Due to the uncountably infinite ways to split a wait action, each agent has an uncountably infinite number of possible paths that satisfy its constraints, resulting in an uncountable set of possible solutions for each state. To resolve this issue, an effective extension of CCBS must introduce a novel type of constraint that not only excludes subsets of colliding solutions but also ensures that a finite number of such constraints can raise the solution cost lower bound to any arbitrary finite value C^*. This property will be crucial for maintaining both completeness and optimality in the continuous domain.

9 Discussion and Conclusion

In this paper, we examined the limitations of the CCBS algorithm, which is not only a cornerstone method for MAPF_R but also the only optimal MAPF_R solver. First, we showed that the CCBS model for waiting motions is ambiguous and gives rise to two possible interpretations for how to handle the corresponding conflicts: *motion constraints* and *vertex range constraints*. We analyse both possibilities and prove that they are unsound and incomplete. In a subsequent result, we also examined the limitations of *vertex range constraints* and showed that this style of reasoning cannot lead to completeness for CCBS. These findings fundamentally undermine the completeness and optimality guarantees of CCBS and its many enhancements and derivatives. For instance, SMT-CCBS, discussed in the latter part of (Andreychuk et al. 2022), tackles the MAPF_R problem using Satisfiability Modulo Theories. While this may appear to be a distinct method, its incorporation of constraints into propositional logic follows the same formulation as CCBS. As a result SMT-CBS inherits the same set of problems: unsound termination or unbounded expansion and eventual failure. All improvements directly built upon CCBS, such as Bypass, Biclique Constraints (Walker, Sturtevant, and Felner 2024b), or Disjoint Splitting (Andreychuk et al. 2021), are similarly affected.

To better understand the theoretical challenges in this area we consider two variant problems which are closely related to MAPF_R but which assume additional restrictions. $\text{MAPF}_\text{R-DT}$ is a variant MAPF_R that only allows unit-cost wait durations. We showed that this problem has an enumerable state-space and is optimally solvable, including by CCBS. $\text{MAPF}_\text{R-NO}$ allows waiting of arbitrary duration but restricts the possible interactions between agents such that waiting agents can not overlap/conflict with agents occupying other vertices and edges. We showed that this problem is solvable, by a reduction to the well known Pebble Motion Problem. Unfortunately $\text{MAPF}_\text{R-NO}$ is not solvable by CCBS and the solvability of the general of general MAPF_R remains unknown. Finally, we consider the essential properties that any future complete and optimal MAPF_R algorithm must satisfy, thus paving the way for future work. Moving forward, we believe that meaningful progress in the area requires either novel constraints within the CCBS framework or entirely new algorithmic approaches. From a theoretical standpoint, establishing the solvability of MAPF_R, or determining whether it can be addressed by CBS-like conflict-based tree search, is a critical step.

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.

Banfi, J.; Basilico, N.; and Amigoni, F. 2017. Intractability of time-optimal multirobot path planning on 2D grid graphs with holes. *IEEE Robotics and Automation Letters*, 2(4): 1941–1947.

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.

Combrink, A.; Roselli, S. F.; and Fabian, M. 2025. Prioritized Planning for Continuous-time Lifelong Multi-agent Pathfinding. arXiv:2503.13175.

Harabor, D.; Hechenberger, R.; and Jahn, T. 2022. Benchmarks for pathfinding search: Iron harvest. In *Proceedings of the International Symposium on Combinatorial Search*, volume 15, 218–222.

Kornhauser, D.; Miller, G.; and Spirakis, P. 1984. Coordinating Pebble Motion On Graphs, The Diameter Of Permutation Groups, And Applications. In *25th Annual Symposium onFoundations of Computer Science, 1984.*, 241–250. ISSN: 0272-5428.

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.

Park, C.; Lee, S.; Yang, H.; Shin, D.; Kang, S.; and Kim, Y. 2023. Conflict-based search with partitioned groups of agents for real-world scenarios. In *2023 20th International Conference on Ubiquitous Robots (UR)*, 986–992. IEEE.

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.

Pradhan, B.; Roy, D. S.; and Hui, N. B. 2018. Motion planning and coordination of multi-agent systems. *International Journal of Computational Vision and Robotics*, 8(5): 492–508.

Röger, G.; and Helmert, M. 2012. Non-optimal multi-agent pathfinding is solved (since 1984). In *Proceedings of the International Symposium on Combinatorial Search*, volume 3, 173–174.

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.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, 151–158.

Surynek, P. 2021. Continuous Multi-agent Path Finding via Satisfiability Modulo Theories (SMT). In Rocha, A. P.; Steels, L.; and Van Den Herik, J., eds., *Agents and Artificial Intelligence*, volume 12613, 399–420. Cham: Springer International Publishing. ISBN 978-3-030-71157-3 978-3-030-71158-0. Series Title: Lecture Notes in Computer Science.

Walker, T. T.; and Sturtevant, N. R. 2019. Collision Detection for Agents in Multi-Agent Pathfinding. arXiv:1908.09707.

Walker, T. T.; Sturtevant, N. R.; and Felner, A. 2018. Extended Increasing Cost Tree Search for Non-Unit Cost Domains. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 534–540. International Joint Conferences on Artificial Intelligence Organization.

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. 2024a. 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.; and Felner, A. 2024b. Clique Analysis and Bypassing in Continuous-Time Conflict-Based Search. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, 152–160.

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.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine*, 29(1): 9–20.

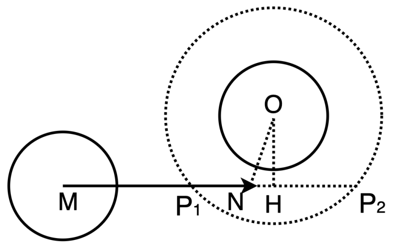


Figure 3: The moving action is $M@t_0 \rightarrow N@t_1$, and the waiting agent is parking at vertex O. The radius of the dashed circle shown in the figure is $2 * r$. P_1 is the point where two agents start to collide, and P_2 is the point the moving agent leaves the collision region if it keeps moving along the extension of (M, N) segment. (O, H) is the perpendicular bisector of (P_1, P_2).

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.

Yu, J.; and LaValle, S. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1): 1443–1449. Number: 1.

Zou, Y.; and Borst, C. 2024. Zeta*-SIPP: Improved Time-Optimal Any-Angle Safe-Interval Path Planning. In Larson, K., ed., *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, 6823–6830. International Joint Conferences on Artificial Intelligence Organization. Main Track.

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 3, 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.

B Counter Example

We now present a concrete example where eliminating such a situation leads to a suboptimal solution (Fig4). For simplicity, we use the notation $v@\tau \rightarrow v'@\tau'$ to represent a motion $\langle m.D, (v, v')\rangle@\tau$ where $\tau' = \tau + m.D$. A plan for an agent can be denoted by chaining these notations together.

In this example agent a_1 moves from A to D, agent a_2 moves from B to C, agent a_3 moves from E to H, and agent a_4 moves from G to B. The first conflict arises between agents a_1 and a_2 on $m_1 = \langle (A, C), 4.9\rangle@0$ and $w_2 = \langle (C, C), \infty\rangle@3.1$. The constraints generated by the CCBS

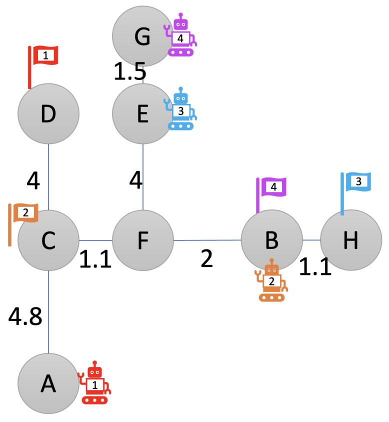


Figure 4: Agent a_1(red) is trying to move from A to D, agent a_2(orange) is trying to move from B to C, agent a_3(blue) is trying to move from E to H, and agent a_4(magenta) is trying to move from G to B. The cost of traversal is labelled on each edge. All agents have a radius of 0.5.

implementation are $\overline{\langle a_1, m_1, [0, \infty)\rangle}$ and $\overline{\langle a_2, C, [3.9, 4.9)\rangle}$. They eliminate the solution of agents a_1 and a_2 performing wait motions prior to departing. This solution allows agents a_3 and a_4 to move first, which avoids a collision.

The final CCBS solution has a cost of 34.6 (costs are rounded to 1 d.p.):
a_1: A@0 $\rightarrow$ C@4.8 $\rightarrow$ D@8.8
a_2: B@0 $\rightarrow$ B@3.2 $\rightarrow$ F@5.2 $\rightarrow$ F@5.2 $\rightarrow$ C@6.3
a_3: **E@0 $\rightarrow$ E@2.6** $\rightarrow$ F@6.6 $\rightarrow$ B@8.6 $\rightarrow$ H@9.7
a_4: **G@0 $\rightarrow$ G@2.1 $\rightarrow$ E@3.6 $\rightarrow$ E@4.0** $\rightarrow$
F@8.0 $\rightarrow$ B@10.0

Here we provide a handcrafted solution[2] with a much smaller cost of 32.1.
a_1: **A@0 $\rightarrow$ A@0.5** $\rightarrow$ C@5.3 $\rightarrow$ D@9.3
a_2: B@0 $\rightarrow$ F@2 $\rightarrow$ C@3.1 $\rightarrow$ **C@3.9 $\rightarrow$ F@5 $\rightarrow$ F@5.6 $\rightarrow$ C@6.7**
a_3: E@0 $\rightarrow$ F@4 $\rightarrow$ B@6 $\rightarrow$ H@7.1
a_4: G@0 $\rightarrow$ E@1.5 $\rightarrow$ E@3 $\rightarrow$ F@7 $\rightarrow$ B@9

In this solution, both agents a_1 and a_2 are required to wait so that agents a_3 and a_4 will not need to. However, CCBS's range constraint eliminates such a solution, therefore vertex and motion constraint pair violates Claim 1, hence they are not sound. This goes to show that the current CCBS implementation is incomplete and may return suboptimal solutions. Note that, the CCBS implementation expanded 780 high-level node before finding the solution, thus, this instance is non-trivial, i.e., not solvable by hand.

[2]**Acknowledgment:** We give special thanks to the original authors of CCBS, Andreychuk et al., for their valuable feedback on an earlier version of this handcrafted solution. Their input helped reduce its cost and improve its robustness against floating-point precision errors.