Optimal Multi-agent Path Finding in Continuous Time

Alvin Combrink[a,*], Sabino Francesco Roselli[a], Martin Fabian[a]

[a]*Division of Systems and Control, Department of Electrical Engineering, Chalmers University of Technology, Göteborg, Sweden*

Abstract

Continuous-time Conflict Based-Search (CCBS) has long been viewed as the standard optimal baseline for multi-agent path finding in continuous time (MAPF$_R$), yet recent critiques show that the theoretically described CCBS can fail to terminate on solvable MAPF$_R$ problems while the publicly available reference implementation can return sub-optimal solutions. This work presents an analytical framework that yields simple and sufficient conditions under which any CCBS-style algorithm is both sound (returns only optimal solutions) and solution complete (terminates on every solvable MAPF$_R$ problem). Investigating the reference CCBS implementation reveals that it violates our sufficient conditions for soundness, with counterexamples demonstrating sub-optimality.

Leveraging the framework, we introduce a branching rule (δ-BR) and prove it restores soundness and termination guarantees. Consequently, the resulting CCBS variant is both sound and solution complete. To our knowledge, this is the first MAPF$_R$ solver matching the guarantees of the discrete-time CBS. On a constructed example, CCBS with δ-BR improves sum-of-costs from 10.707 to 9.000 ($\approx$ 16% lower) compared to the reference CCBS implementation. Across benchmarks, the reference CCBS implementation is generally able to find solutions faster than CCBS with δ-BR due to its more aggressive pruning. However, this comes at the cost of occasional sub-optimality and potential non-termination when all solutions are pruned, whereas δ-BR preserves optimality and guarantees termination by design. Because δ-BR largely only affects the branching step, it can be adopted as a drop-in replacement in existing codebases, as we show in our provided implementation. Beyond CCBS, the analytical framework and termination criterion provide a systematic way to evaluate other CCBS-like MAPF$_R$ solvers and future extensions, thereby offering tools for rigorous analysis of next-generation MAPF$_R$ algorithms.

Keywords: Multi-agent Path Finding, Continuous-time Conflict-based Search, Algorithm soundness and solution completeness, Optimal Path planning

1. Introduction

As robotics and automation take an increasingly larger role in society, there is a growing need to move agents from where they are to where they need to be without collision. This is the

*Corresponding author.

Email addresses: combrink@chalmers.se (Alvin Combrink), rsabino@chalmers.se (Sabino Francesco Roselli), fabian@chalmers.se (Martin Fabian)

problem of Multi-agent Path Finding (MAPF), which finds application across diverse domains, from computer games [1] to warehouses [2, 3], factories [4] to airports [5]. Progress toward fast and exact solution methods has been steady for decades, targeting the many variants of MAPF.

In its classical formulation, MAPF is about finding a collision-free path for each agent on a graph from its start vertex to goal vertex. Agents are represented as points and can wait at vertices and traverse edges, with time being discrete, edges taking unit-time to traverse, and an objective function based on arrival times at goal vertices being minimized. This problem is NP-hard [6] and has many variations, comprehensively detailed in, e.g., [7, 8, 9]. Early focus was placed on reducing computation and searching for sub-optimal solutions by, e.g., establishing a priority order among agents or restricting the search to a limited time-window [10, 11]. Shortly thereafter, exact methods such as M^* [12], *Increasing Cost Tree Search* [13] and *Conflict Based-Search* (CBS) [14] were introduced. These methods cleverly avoid searching the entire solution space — which grows exponentially with the number of agents — by adopting a "lazy" approach of planning agents individually and addressing collisions only as they occur. Of the three, CBS has enjoyed the largest adoption with multiple continuations. To name a few, *Meta-Agent CBS* [15] and *Improved CBS* [16] for performance enhancements, *Enhanced CBS* [17] and *CBS-Budget* [18] for bounded sub-optimal solutions, *Rolling Horizon Collision Resolution* [19] for *lifelong MAPF*, and *Multi-Goal CBS* [20] for MAPF where agents are assigned multiple goal vertices.

The simplifying assumptions of discrete time and unit-length edges in MAPF bring significant drawbacks to its practical applicability; in real-world settings, environments do not always follow grid-like structures, agent speeds and action times vary, and artificial wait times to ensure that agents move in *lockstep* inflate the objective value. The continuous-time MAPF variant, *MAPF$_R$* [21], removes such discrete-time assumptions. In MAPF$_R$, agents can traverse edges at any real-valued time by following one of potentially several motion functions that define a continuous trajectory between two connected vertices. Additionally, agents have volumes such that two agents collide when their volumes overlap. The work in [22] addresses MAPF$_R$ for sub-optimal solutions, while [23] finds optimal solutions to a simplified MAPF$_R$ problem where agents traverse edges in straight lines at constant speeds. To our knowledge, the only known method to address MAPF$_R$ in its entirety for optimal solutions is *Continuous-time CBS* (CCBS) [21] which claims to be *sound* (only returns optimal solutions) and *solution complete* (guaranteed to terminate on any solvable MAPF$_R$ problem). Since the introduction of CCBS, numerous continuation works have been published, such as [24] for performance enhancements, *T-Robust CCBS* [25] for solutions that are robust to delays, [26] for *Any-Angle MAPF*, *Continuous-time Prioritize Lifelong Planner* [27] for lifelong MAPF, and [28] where CCBS is modified and applied to quadcopter drones.

Despite CCBS's wide-spread adoption, only recently has a critical assessment in [29] established that CCBS does not fulfill its guarantees. The theoretical description of CCBS does not guarantee termination under the existence of an optimal solution (a condition for solution completeness) and the publicly available implementation does not guarantee that a returned solution is optimal (a condition for soundness). This is shown in [29] theoretically and with counter-examples. Therefore, these results reveal a significant gap between the theory and practice of CCBS and any continuation works that rely on the soundness and solution completeness guarantees.

In this work, we develop an analytical framework to aid the analysis of CCBS and find sufficient but not necessary requirements for CCBS to be sound and solution complete. We then show that the publicly available CCBS implementation does not satisfy these requirements, which by

itself does not mean that CCBS is not sound and solution complete, but it does complement the results in [29] and our experiments showing unsoundness. Leveraging this framework, we introduce a new branching rule (δ-BR) and prove that CCBS using δ-BR is sound and solution complete. That is, CCBS with δ-BR is guaranteed to return an optimal solution within a finite number of iterations on any solvable MAPF$_R$ problem. Our only assumption on the objective function is that it is strictly monotonically increasing with respect to the maximum agent arrival time, thus, we guarantee optimality for *sum-of-costs*, *makespan*, and any other objective function with this property. Therefore, to our knowledge, we provide the first provably sound and solution-complete method to the MAPF$_R$ problem. With our provided implementation, we show that δ-BR can be adopted as a drop-in replacement in existing CCBS implementations. These results are empirically validated on a constructed counterexample and benchmark problems, showing that the existing CCBS implementation is generally able to find solutions faster than CCBS with δ-BR due to its aggressive search space pruning, however, occasionally returns sub-optimal solutions after removing all optimal solutions. Furthermore, the introduced framework and non-termination criteria provide analytical tools for reasoning about CCBS-like solvers and their extensions, thereby laying a foundation for the next generation of sound and solution complete MAPF$_R$ algorithms.

The outline of this article is the following: In Section 2, we formally define the MAPF$_R$ problem, soundness, and solution completeness. CCBS is also introduced, along with details surrounding the findings in [29]. The framework for analyzing branching rules and CCBS's algorithmic properties is presented in Section 3. In Section 4, we apply the framework on CCBS's implemented branching rule. Our branching rule is presented in Section 5, along with soundness and solution completeness proofs. Section 6 presents empirical results. Finally, conclusions are found in 7.

2. Background

In this section, we formally introduce the MAPF$_R$ problem and define what we mean by *soundness* and *solution completeness*. Thereafter, CCBS [21] and the recent findings on CCBS's guarantees in [29] are presented.

2.1. Problem Formulation

A MAPF$_R$ problem is defined in [21] by a tuple $\langle \mathcal{G}, \mathcal{M}, S, G, \mathit{coord}, \mathcal{A} \rangle$, where $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ represents a graph with vertices $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, $\mathcal{M}$ denotes a 2D metric space, S and G respectively map agents to start and goal vertices, $\mathit{coord} : \mathcal{V} \to \mathcal{M}$ maps vertices to coordinates in $\mathcal{M}$, and $\mathcal{A}$ comprises a finite set of *move actions*.

A move action $m = (m_\varphi, m_D)$ offers a way for an agent to traverse an edge $(v, v') \in \mathcal{E}$. The *motion function* $m_\varphi : [0, m_D] \to \mathcal{M}$ describes a continuous trajectory with *duration* m_D starting at v and ending at v'; that is, $m_\varphi(0) = \mathit{coord}(v)$ and $m_\varphi(m_D) = \mathit{coord}(v')$. We denote the source and target vertices as $\mathit{from}(m) = v$ and $\mathit{to}(m) = v'$, respectively. Each edge in $\mathcal{E}$ admits a finite number of move actions, providing multiple options for an agent to traverse an edge. Figure 1 illustrates this concept through edge (v, v') and three associated move actions m_1, m_2, and m_3. While all move actions start and end at the same vertices, their trajectories (by $m_{1,\varphi}$, $m_{2,\varphi}$, $m_{3,\varphi}$) and durations ($m_{1,D}$, $m_{2,D}$, $m_{3,D}$) may be distinct.

In addition to move actions, there are also *wait actions* not included in $\mathcal{A}$. A wait action $w = (w_\varphi, w_D)$ maps to a single vertex $v \in \mathcal{V}$ such that its motion function $w_\varphi : [0, w_D] \to \mathit{coord}(v)$

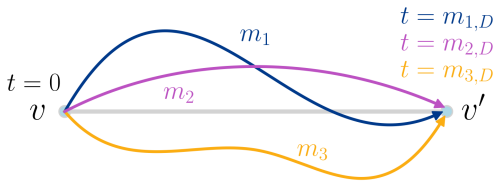


Figure 1: An edge $(v, v') \in \mathcal{E}$ and its associated move actions m_1, m_2, and m_3. Each action specifies a trajectory in $\mathcal{M}$ from v to v', starting at time $t = 0$ and ending after potentially different durations ($m_{1,D}$, $m_{2,D}$, $m_{3,D}$).

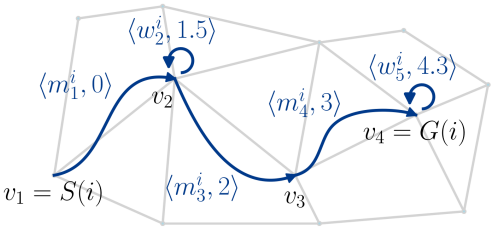


Figure 2: An illustration of a valid plan $\pi_i = \langle\langle m_1^i, 0\rangle, \langle w_2^i, 1.5\rangle, \langle m_3^i, 2\rangle, \langle m_4^i, 3\rangle, \langle w_5^i, 4.3\rangle\rangle$ with action durations $m_{1,D}^i = 1.5$, $w_{2,D}^i = 0.5$, $m_{3,D}^i = 1$, $m_{4,D}^i = 1.3$, $w_{5,D}^i = \infty$.

with $w_D \in \mathfrak{R} \cup \{\infty\}$. We denote $v(w) = \mathit{from}(w) = \mathit{to}(w) = v$. Wait actions allow for agents to wait at vertices, which may be necessary to e.g. avoid colliding with other agents.

A *timed action* $\langle a, t\rangle$ (with a either a move or wait action) specifies that an agent executes a starting at time $t \in \mathfrak{R}$, such that the motion function a_φ is shifted by t. That is, the location in $\mathcal{M}$ of an agent executing $\langle a, t\rangle$ is $a_\varphi(t' - t)$ during $t' \in [t, t + a_D]$. In other words, an action specifies how an agent moves and the duration to do so, while a timed action additionally specifies when the movement begins.

An agent i can execute multiple timed actions, one after the other, to thereby traverse the graph. A *plan* π_i for an agent i is a finite sequence of timed actions,

$$\pi_i = \langle\langle a_1^i, t_1^i\rangle, \langle a_2^i, t_2^i\rangle, \ldots, \langle a_n^i, t_n^i\rangle\rangle. \tag{1}$$

For a plan π_i to be *valid*, it must start at time 0 ($t_1^i = 0$), start and end at i's start and goal vertices ($\mathit{from}(a_1^i) = S(i)$ and $\mathit{to}(a_n^i) = G(i)$), and every successive timed action must start when and where the previous ends,

$$\left(t_k^i = t_{k-1}^i + a_{k-1,D}^i\right) \wedge \left(\mathit{to}(a_{k-1}^i) = \mathit{from}(a_k^i)\right) \quad \forall k = 2, .., n.$$

Additionally, the final action a_n^i is an *infinite wait action* with duration $a_{n,D}^i = \infty$ to capture that agents always remain idle indefinitely at their last location. The duration of π_i is $\pi_{i,D} = \sum_{k=1}^{n-1} a_{k,D}^i$, naturally excluding the final infinite wait action.

Figure 2 illustrates an example of a valid plan π_i starting at $S(i)$ and ending at $G(i)$. The first timed move action is $\langle m_1^i = \langle m_{1,\varphi}^i, 1.5\rangle, 0\rangle$, which specifies that agent i begins at $t = 0$ to move according to the motion function $m_{1,\varphi}^i$ which has duration 1.5. This is followed by the timed wait action $\langle w_2^i = \langle w_{2,\varphi}^i, 0.5\rangle, 1.5\rangle$ specifying that i waits at $v(w_2^i) = v_2$ from $t = 1.5$ for a duration of 0.5, that is until $t = 2$. Thereafter, i executes $\langle m_3^i = \langle m_{3,\varphi}^i, 1\rangle, 2\rangle$ to arrive at v_3 at $t = 3$, and then executes $\langle m_4^i = \langle m_{4,\varphi}^i, 1.3\rangle, 3\rangle$ to arrive at v_4 at $t = 4.3$. Finally, the timed wait action $\langle w_5^i = \langle w_{5,\varphi}^i, \infty\rangle, 4.3\rangle$ specifies that i waits at v_4 indefinitely from $t = 4.3$. Naturally, the plan π_i illustrated in the example represents merely one among countless valid plans; alternative wait actions could be employed at any of the vertices, different move actions could be selected for the same edges, or an entirely different path through the graph could be traversed.

Each agent occupies a non-zero volume; a collision occurs when the volumes of two agents overlap. A *conflict* $\langle\langle a^i, t^i\rangle, \langle a^j, t^j\rangle\rangle \in \pi_i \times \pi_j$ arises when agents i and j collide at some time while respectively executing $\langle a^i, t^i\rangle$ and $\langle a^j, t^j\rangle$. A conflict $\langle\langle m^i, t^i\rangle, \langle m^j, t^j\rangle\rangle$ with two move

actions is a *move-move* conflict, and a conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with a move and a wait action is a *move-wait* conflict. By convention, the move action in a move-wait conflict is stored in the first position. Note that a *wait-wait* conflict can only arise after a move-move or move-wait (assuming agents start in collision-free positions). Therefore, we sufficiently consider only move-move and move-wait conflicts, as a wait-wait conflict cannot exist without one of the other conflict types. Two plans π_i and π_j conflict if there exists a conflict $\langle\langle a^i, t^i\rangle, \langle a^j, t^j\rangle\rangle \in \pi_i \times \pi_j$.

A *joint plan* Π is a set containing one valid plan for each agent, a *solution* is a joint plan that does not contain any two conflicting plans, and an optimal solution Π^* is a solution that minimizes an objective function σ over all solutions. The *sum-of-costs* $\sigma(\Pi) = \sum_{\pi\in\Pi} \pi_D$ (sum over plan durations) and *makespan* $\sigma(\Pi) = \max_{\pi\in\Pi} \pi_D$ (maximum over plan durations) are the two most common objective functions in the MAPF context. Here, we assume that $\sigma(\Pi)$ is strictly monotonically increasing with respect to the maximum duration over all agent plans in Π, $\max_{\pi\in\Pi} \pi_D$, which both the sum-of-costs and makespan satisfy. Therefore, all proofs and guarantees provided in this work are valid for any objective function satisfying this assumption.

2.2. Algorithmic Properties

In this article, we say that a solution algorithm to the MAPF$_R$ problem is

- *sound* if it only returns optimal solutions;
- *solution complete* if it is guaranteed to terminate with a solution on any solvable MAPF$_R$ problem.

In [21], a sound algorithm returns only solutions, while an exact algorithm[1] returns only optimal solutions. These terms are never addressed separately in [21] or here. Therefore, we denote both by *sound* for simplicity. Also note that *completeness* guarantees the return of a solution if one exists or the report of no solution otherwise, while the weaker *solution completeness* only guarantees the former.

2.3. Continuous-time Conflict Based Search

CCBS is argued to be a sound and solution-complete MAPF$_R$ solver for minimal sum-of-costs [21]. CCBS closely follows CBS by simultaneously building and searching a binary constraint tree (CT).

To explain the CT and how it is searched, we begin by briefly introducing CCBS's underlying planner *Constrained Safe Interval Path Planning* (CSIPP) which is covered in more detail in Section 5.3.2. CSIPP — a variant of *Safe Interval Path Planning* (SIPP) [30] — computes a minimal-duration valid plan for a single agent that satisfies a given set of constraints C. In SIPP, an A^* [31] search is performed in the vertex-safe interval state space, with edges in $\mathcal{E}$ providing transitions from one state to the next, and safe intervals defining when an edge may be traversed or a vertex may be occupied. In CSIPP, constraints define the safe intervals and move actions provide transitions between states.

In the CT, a node N is associated with a set of constraints N_C and a joint plan N_Π. The plans in N_Π are computed using CSIPP given the constraint set N_C, such that every plan $\pi_i \in N_\Pi$ is valid and has minimal duration under the constraints in N_C. The CCBS search starts at the CT

[1] In [21], the term *optimal* is used synonymously with *exact*. However, an algorithm's optimality typically refers to its time and space complexity and not to the quality of the solutions that it produces.

root node N^R with the empty constraint set $N_C^R = \varnothing$. Since N_C^R is empty, every plan $\pi_i \in N_\Pi^R$ is unconstrained (apart from being valid) and therefore has the shortest possible duration over all valid plans for agent i. At every iteration of CCBS, a node N minimizing $\sigma(N_\Pi)$ over all unexpanded nodes is selected for expansion. If N_Π is a solution (i.e., conflict-free), then N_Π is returned as an optimal solution. Otherwise, a *branching rule* is applied to N to spawn two child nodes. To do so, an arbitrary conflict in N_Π is selected (say between agents i and j) and used to construct a pair of constraints $\langle c_i, c_j \rangle$. These constraints c_i and c_j respectively constrain the plans of agents i and j to avoid this specific conflict. The child nodes N^i and N^j each inherit one of the constraints in addition to all constraints at N: $N_C^i = N_C \cup \{c^i\}$ and $N_C^j = N_C \cup \{c^j\}$. The plans in N_Π^i and N_Π^j are then computed using CSIPP to satisfy N_C^i and N_C^j, respectively.

We clarify the above by introducing an example in Figure 3, which will later be revisited in more detail. In this example, we consider only two agents i and j. At the CT node N (Figure 3a) a conflict is detected in the joint plan $N_\Pi = \{\pi_i, \pi_j\}$, illustrated in Figure 3b. Specifically, i and j collide while respectively executing the timed actions $\langle m_2^i, t_2^i \rangle$ and $\langle m_3^j, t_3^j \rangle$. Therefore, $\langle \langle m_2^i, t_2^i \rangle, \langle m_3^j, t_3^j \rangle \rangle$ is a conflict. Based on this conflict, a branching rule is used to create a pair of constraints $\langle c^i, c^j \rangle$. We will explore these constraints in more detail later; for now it is sufficient to know that c^i constrains when i may execute m_2^i, and c^j constrains when j may execute m_3^j, to avoid this specific conflict. Consider the first child node N^i (Figure 3a) with constraints $N_C^i = N_C \cup \{c^i\}$. Using CSIPP, the plans in N_Π^i are computed to satisfy N_C^i. In practice, it is sufficient to only compute i's plan since only the constraints on i have changed, while all other agents can inherit their plans from N. Figure 3c illustrates N_Π^i, where agent i now takes a different path through the graph. Similarly, the other child node N^j has constraints $N_C^j = N_C \cup \{c^j\}$, and a new plan for j is computed using CSIPP. The joint plan N_Π^j is illustrated in Figure 3d, where j now waits for some time at a vertex before continuing. In this particular example, the resulting joint plans at the child nodes happen to be conflict-free and are therefore both solutions. In general, however, the child nodes' joint plans may still contain conflicts, requiring further CCBS iterations until a solution is found.

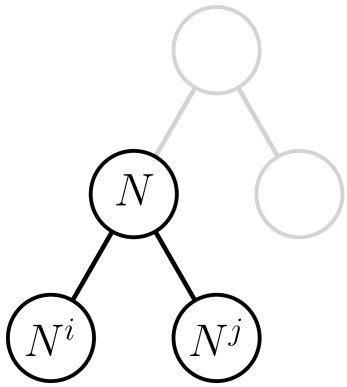
The soundness and solution completeness proofs of CCBS in [21] rely on the CT satisfying the following properties, where N is a parent to N^i and N^j and $\mathcal{S}(N)$ is the set of all solutions satisfying N_C:

$$\mathcal{S}(N) = \mathcal{S}(N^i) \cup \mathcal{S}(N^j), \tag{2}$$

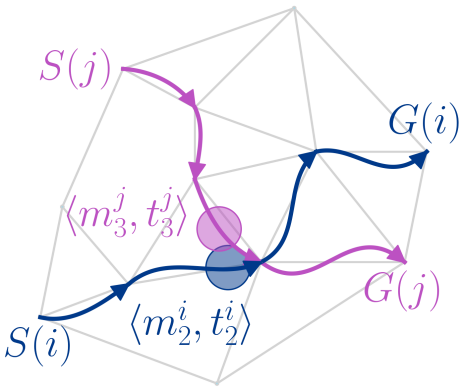
$$\sigma(N_\Pi) \leqslant \min\left(\sigma(N_\Pi^i), \sigma(N_\Pi^j)\right). \tag{3}$$

Property (2) ensures that no solutions are removed when branching a node. At the CT root N^R, since $N_C^R = \varnothing$ it holds that all solutions to a MAPF$_R$ problem are contained in $\mathcal{S}(N^R)$. If property (2) holds then $\mathcal{S}(N^R) = \mathcal{S}(N^i) \cup \mathcal{S}(N^j)$ (with N^i and N^j being children of N^R) which recursively implies that all solutions are reachable from the CT root. Since the search starts at N^R, all solutions to a MAPF$_R$ problem are included in the search and therefore can be found. Since it is the branching rule that determines $\mathcal{S}(N^i)$ and $\mathcal{S}(N^j)$ (by the constraint pair that it constructs), it is the branching rule that determines if property (2) is satisfied.

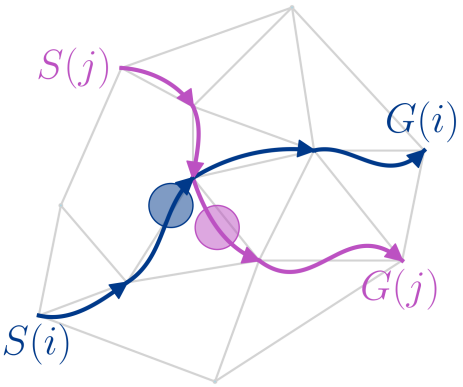
Property (3) is automatically satisfied by CSIPP: since plans in N_Π are computed using CSIPP given constraints N_C, every plan $\pi_i \in N_\Pi$ is the shortest-duration valid plan over all valid plans satisfying N_C. For all nodes N' below N we know that $N_C \subset N'_C$ since constraints are only added during a branching, never removed. Adding constraints can never result in shorter-duration plans. Therefore, it holds for all agents i that $\pi_{i,D} \leqslant \pi'_{i,D}$ where $\pi_{i,D} \in N_\Pi$ and $\pi'_{i,D} \in N'_\Pi$. Since $\sigma(\Pi)$



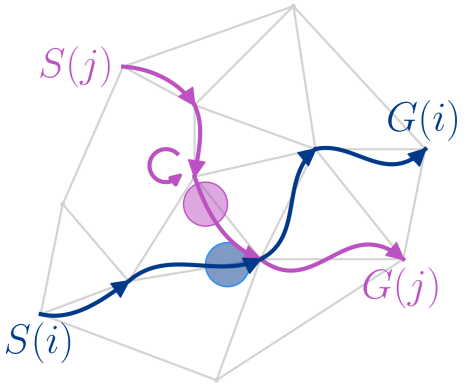
(a) The CT node N where a conflict is detected in N_Π, and its two children N^i and N^j.



(b) The joint plan $N_\Pi = \{\pi_i, \pi_j\}$ contains a conflict $\langle\langle m_2^i, t_2^i\rangle, \langle m_3^j, t_3^j\rangle\rangle$ between agents i and j.



(c) The joint plan N_Π^i. Additional constraints on agent i are added to N_C^i to specifically avoid the collision detected at N. A new plan $\pi_i \in N_\Pi^i$ satisfying N_C^i is then computed, where i traverses another path in the graph.



(d) The joint plan N_Π^j. Additional constraints on agent j are added to N_C^j to specifically avoid the collision detected at N. A new plan $\pi_j \in N_\Pi^j$ satisfying N_C^j is then computed, where j waits for some time at a node.

Figure 3: An example of a branching rule applied to a conflict $\langle\langle m_2^i, t_2^i\rangle, \langle m_3^j, t_3^j\rangle\rangle$ at CT node N. For agent i (j), a child node N^i (N^j) is spawned with additional constraints on i (j) to avoid this specific conflict. A new plan is then computed which satisfies these constraints.

is strictly monotonically increasing with the maximum plan duration over all plans $\pi \in \Pi$, it follows that $\sigma(N_\pi) \leqslant \sigma(N'_\pi)$, thereby satisfying property (3).

In this article, we focus our attention on two branching rules: the *theoretical branching rule* (TBR) which is used in the soundness and solution completeness proofs in [21], and the *implemented branching rule* (IBR) found in the publicly available reference CCBS implementation[2].

2.3.1. The Theoretical Branching Rule

The TBR uses the abstract definition of actions, encompassing move and wait actions alike, such that move-move and move-wait conflicts are addressed using the same procedure. Given a conflict $\langle\langle a^i, t^i\rangle, \langle a^j, t^j\rangle\rangle$, the TBR finds the earliest time for each agent to execute its action without colliding with the other agent. That is, the earliest time $t^i_u > t^i$ for agent i to execute $\langle a_i, t^i_u\rangle$ without colliding with j executing $\langle a^j, t^j\rangle$ is found. The interval $[t^i, t^i_u)$ is denoted the *unsafe interval* since $\langle a_i, t\rangle$ for any $t \in [t^i, t^i_u)$ certainly conflicts with $\langle a^j, t^j\rangle$. This unsafe interval is used to define the *motion constraint* $c_i = \langle i, a^i, [t^i, t^i_u)\rangle$ specifying that i is forbidden from executing $\langle a^i, t\rangle$ for any $t \in [t^i, t^i_u)$. Likewise for agent j, the unsafe interval $[t^j, t^j_u)$ is found and the motion constraint $c_j = \langle j, a_j, [t^j, t^j_u)\rangle$ is defined. Letting the conflict be detected at CT node N, such that $\langle\langle a^i, t^i\rangle, \langle a^j, t^j\rangle\rangle \in \pi_i \times \pi_j$ with $\pi_i, \pi_j \in N_\Pi$, the TBR uses the constraint pair $\langle c_i, c_j\rangle$ to branch on N. The CT using the TBR is shown in [21] to satisfy properties (2) and (3), by which they then conclude that CCBS is sound and solution complete.

Let us now revisit the example in Figure 3 where the conflict $\langle\langle m^i_2, t^i_2\rangle, \langle m^j_3, t^j_3\rangle\rangle$ was detected. From this conflict, the motion constraints $c^i = \langle i, m^i_2, [t^i_2, t^i_u)\rangle$ and $c^j = \langle j, m^j_3, [t^j_3, t^j_u)\rangle$ are formed. At node N^i with $N^i_C = N_C \cup \{c^i\}$, agent i is no longer permitted to execute $\langle m^i_2, t\rangle$ for $t \in [t^i_2, t^i_u)$. Therefore, an alternative plan is found where i avoids executing m^i_2 entirely. Similarly at node N^j where $N^j_C = N_C \cup \{c^j\}$, agent j is no longer permitted to execute $\langle m^j_3, t\rangle$ for $t \in [t^j_3, t^j_u)$. Therefore, an alternative plan is found where j waits long enough to eventually execute m^j_3 at t^j_u.

2.3.2. The Implemented Branching Rule

The IBR differs from the TBR by how move-wait conflicts $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ are branched on. To avoid confusion between the IBR and the described implementation in [21], we first describe the latter and then how the IBR differs. With the TBR, the motion constraint $\langle j, w^j, [t^j, t^j_u)\rangle$ is created for the move-wait conflict above. Such a motion constraint forbids j from starting the execution of w^j at any time in $[t^j, t^j_u)$. That is, $\langle w^j, t\rangle$ is forbidden for all $t \in [t^j, t^j_u)$. However, in [21], CSIPP is described to implement such a motion constraint as a *vertex constraint* $\langle j, v(w^j), [t^j, t^j_u)\rangle$, forbidding j from occupying the vertex $v(w^j)$ during $[t^j, t^j_u)$. Observe that such a vertex constraint is more constraining than the motion constraint; the motion constraint forbids *specifically* w^j from being executed *starting* at any time in $[t^j, t^j_u)$, while the vertex constraint forbids *every wait action* w *at the same vertex* ($v(w) = v(w^j)$) from being executed if *it overlaps with* $[t^j, t^j_u)$, i.e., if $[t, t+w_D] \cap [t^j, t^j_u) \neq \varnothing$. Therefore, already here we see a dissonance between the TBR and the described implementation.

Similar to the described implementation, the IBR also implements vertex constraints. However, the IBR uses the *intersection interval* $\bar{I}^c$ instead of the unsafe interval $[t^j, t^j_u)$. Given the conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$, $\bar{I}^c$ (formally defined in Definition 3.4) is the time interval during which

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

agent i while executing $\langle m^i, t^i \rangle$ would collide with agent j waiting indefinitely at $v(w^j)$. Thus, the IBR creates a vertex constraint $\langle j, v(w^j), \bar{I}^c \rangle$. The intersection interval $\bar{I}^c$ is not necessarily the same as the unsafe interval $[t^j, t^j_u)$ since it does not take into account when j begins executing its wait action. For example, if for $\bar{I}^c = [\bar{t}^c_1, \bar{t}^c_2)$ we have $\bar{t}^c_1 < t^j < \bar{t}^c_2$, then j can only begin waiting at $v(w^j)$ from $\bar{t}^c_2$ else a collision occurs, resulting in the unsafe interval $[t^j, \bar{t}^c_2) \neq \bar{I}^c$. Besides this, timed move actions in both move-move and move-wait conflicts are handled in the same way as in the TBR, resulting in the motion constraint $\langle i, a^i, [t^i, t^i_u) \rangle$.

Therefore, the IBR uses the constraint pair $\langle \langle i, m^i, [t^i, t^i_u) \rangle, \langle j, m^j, [t^j, t^j_u) \rangle \rangle$ for move-move conflicts and $\langle \langle i, m^i, [t^i, t^i_u) \rangle, \langle j, v(w^j), \bar{I}^c \rangle \rangle$ for move-wait conflicts. No proofs of soundness and solution completeness are provided for CCBS under this branching rule.

2.4. A Critique on CCBS

The work in [29] questions the algorithm properties of CCBS. The claims are two-fold:

1. Under the TBR, CCBS does not guarantee termination;
2. Under the IBR, CCBS violates property (2).

At its core, the identified issue lies in how wait actions are handled when resolving conflicts. For Claim 1, [29] highlights that under the TBR on a wait action $\langle w^i, t^i \rangle$ (giving rise to a motion constraint $\langle i, w^i, [t^i, t^i_u) \rangle$), only one wait duration w^i_D out of infinitely many real-valued durations is removed from the search. E.g., a wait action $w = (w^i_\varphi, w^i_D + \epsilon)$ with ϵ arbitrarily small is still permitted during $[t^i, t^i_u)$. Therefore, despite satisfying property (2) ensuring that any optimal solution is reachable from the CT root, there may exist infinitely many nodes between such a solution and the root. Hence, CCBS under the TBR does not always terminate after finite iterations, despite the existence of a solution. Therefore, CCBS under the TBR is not solution complete. Essentially, [29] identifies that properties (2) and (3) are not sufficient to guarantee solution completeness.

For Claim 2, [29] provides two counterexamples showing how the IBR on a move-wait conflict removes valid solutions from the search (violating property (2)), and consequently returning sub-optimal solutions. Thus, CCBS under the IBR does not guarantee to only return optimal solutions and is therefore not sound.

Furthermore, [29] introduces *shifting constraints*. They show how a branching rule using shifting constraints, which relies on a parameter $\delta \geqslant 0$, does not remove collision-free solutions from the search and is therefore an improvement on the IBR. However, they conclude that if $\delta > 0$ then in the CT there will always remain permitted joint plans containing conflicts, leading to potentially infinitely many nodes being searched. If instead $\delta = 0$, then a similar issue as with the TBR arises, also potentially requiring the expansion of infinitely many nodes. That is, CCBS using shifting constraints has the same properties as using the TBR: sound but without termination guarantees.

3. Preliminaries

This section establishes the theoretical foundations for analyzing branching rules in CCBS. We first introduce a decomposed form of CCBS constraints and extend to this the notion of *sound constraint pairs*. We then define a *sound branching rule* as one that preserves all solutions during node expansion and prove that a branching rule is sound if and only if its underlying constraint

pair is sound. Using these definitions, we show that a sound branching rule is sufficient to ensure that CCBS is sound, and is also solution complete if CCBS is guaranteed to terminate. Finally, we formally introduce the collision interval and intersection interval.

3.1. Sound Branching

A definition for a *sound* pair of constraints $\langle c_1, c_2 \rangle$ is provided in [21, 32]. Let the set $\mathcal{S}$ contain all solutions to a given MAPF$_R$ problem, and let $c(\Pi)$ be true if a joint plan Π satisfies a constraint c, false otherwise.

Definition 3.1 (Sound pair of constraints [21, 32])**.** *For a given MAPF$_R$ problem with solutions $\mathcal{S}$, a pair of constraints $\langle c_1, c_2 \rangle$ is sound if*

$$c_1(\Pi) \vee c_2(\Pi), \quad \forall \Pi \in \mathcal{S}. \tag{4}$$

Throughout this article, we shall consider a decomposed form of the motion and vertex constraints by describing them as *constraint sets* comprised of *forbidding constraints*. These constraint forms are equivalent, however, the decomposed form allows us to analyze constraint pairs used in branching rules more explicitly than when using motion and vertex constraints. A forbidding constraint $\langle i, a, t \rangle$ forbids i from executing the specific timed action $\langle a, t \rangle$. A motion constraint $c = \langle i, a^i, [t^i, t^i_u) \rangle$, which forbids i from executing action a_i starting at any time $t \in [t^i, t^i_u)$, is equivalent to the dense constraint set

$$C = \left\{ \langle i, a^i, t \rangle \mid t \in [t^i, t^i_u) \right\}$$

such that $c(\Pi) \Leftrightarrow C(\Pi)$ where $C(\Pi) = \bigwedge_{c' \in C} c'(\Pi)$. We denote this equivalence by $c \Leftrightarrow C$. Similarly, a vertex constraint $c = \langle i, v, \bar{I}^c \rangle$ forbidding i from occupying v at any time $t \in \bar{I}^c$ is equivalent to $C = \{ \langle i, w, t \rangle \mid v(w) = v, [t, t + w_D] \cap \bar{I}^c \neq \varnothing \}$.

We now provide a corresponding soundness definition for a pair of constraint sets, and then define soundness of a branching rule. Then, in Lemma 3.1, we show that a branching rule using a pair of constraint sets is sound if and only if the constraint set pair is sound.

Definition 3.2 (Sound pair of constraint sets)**.** *For a given MAPF$_R$ problem, a pair of constraint sets $\langle C_1, C_2 \rangle$ is sound if $\forall \langle c_1, c_2 \rangle \in C_1 \times C_2$, $\langle c_1, c_2 \rangle$ is sound.*

Definition 3.3 (Sound branching rule)**.** *A branching rule applied to a CT node N to spawn children N^1 and N^2 is sound if property* (2)*, $S(N) = S(N^1) \cup S(N^2)$, always holds.*

Lemma 3.1 (Equivalence of sound branching and sound constraint set pair)**.** *Let a branching rule applied to a CT node N use a constraint set pair $\langle C_1, C_2 \rangle$ to spawn two children N^1 and N^2, such that $N^1_C = N_C \cup \{C_1\}$ and $N^2_C = N_C \cup \{C_2\}$. The branching rule is sound iff $\langle C_1, C_2 \rangle$ is sound.*

Proof. To prove the branching rule's soundness, we must show that property (2), $\mathcal{S}(N) = \mathcal{S}(N^1) \cup \mathcal{S}(N^2)$, is always satisfied. For each child N^i, $i = 1, 2$, the solutions permitted under the constraints N^i_C are

$$\begin{aligned} \mathcal{S}(N^i) &= \left\{ \Pi \in \mathcal{S} \mid c(\Pi), c \in N^i_C \right\} \\ &= \{ \Pi \in \mathcal{S} \mid c(\Pi), c \in N_C \cup \{C_i\} \} \\ &= \{ \Pi \in \mathcal{S}(N) \mid C_i(\Pi) \} . \end{aligned}$$

Therefore, for the branching rule to be sound it must hold for every solution $\Pi \in \mathcal{S}(N)$ that $C_1(\Pi) \vee C_2(\Pi)$ since then $\mathcal{S}(N) = \mathcal{S}(N^1) \cup \mathcal{S}(N^2)$. For every $\Pi \in \mathcal{S}(N)$, consider two cases:

1. $C_1(\Pi) \vee C_2(\Pi)$ holds, then we immediately know that $\Pi \in \mathcal{S}(N^1) \cup \mathcal{S}(N^2)$;
2. otherwise, assume without loss of generality that $\exists c_1 \in C_1 : \neg c_1(\Pi)$.

For the latter case, we start by assuming that $\langle C_1, C_2 \rangle$ is sound. By Definition 3.2, every pair $\langle c_1, c_2 \rangle \in C_1 \times C_2$ is sound, meaning $c_1(\Pi) \vee c_2(\Pi)$ holds. Since we assumed $\neg c_1(\Pi)$, it follows that $c_2(\Pi)$ must be true. This reasoning applies to every pair $\langle c_1, c_2 \rangle \in C_1 \times C_2$, such that $\forall c_2 \in C_2 : c_2(\Pi)$ holds. This means that $C_2(\Pi)$ is true and therefore that $\Pi \in \mathcal{S}(N^2)$. Therefore, if $\langle C_1, C_2 \rangle$ is sound then it holds for all $\Pi \in \mathcal{S}(N)$ that $\Pi \in \mathcal{S}(N^1) \cup \mathcal{S}(N^2)$. Thus, if $\langle C_1, C_2 \rangle$ is sound then the branching rule is sound.

Assume instead that $\langle C_1, C_2 \rangle$ is unsound, then by Definition 3.2 there exists an unsound pair $\langle c_1, c_2 \rangle \in C_1 \times C_2$. Since $\langle c_1, c_2 \rangle$ is unsound, there exists a valid solution Π such that $\neg(c_1(\Pi) \vee c_2(\Pi))$, implying $\neg C_1(\Pi)$ and $\neg C_2(\Pi)$. Therefore, $\Pi \notin \mathcal{S}(N^1) \cup \mathcal{S}(N^2)$. Since $\mathcal{S}(N)$ can contain such a solution Π, it does not always hold for all $\Pi \in \mathcal{S}(N)$ that $\Pi \in \mathcal{S}(N^1) \cup \mathcal{S}(N^2)$. Thus, if $\langle C_1, C_2 \rangle$ is unsound then the branching rule is unsound.

□

Finally, we put branching rules back into the context of the CCBS algorithm by showing in Theorem 3.2 that CCBS is sound when using a sound branching rule. Additionally, we show in Theorem 3.3 that CCBS is solution complete if it is both sound and guaranteed to terminate.

Theorem 3.2 (Soundness)**.** *CCBS is sound when using a sound branching rule.*

Proof. Property (2) is satisfied since the branching rule is sound, such that all solutions are reachable from the CT root. CCBS performs a best-first search starting at the root, always expanding the CT node with minimum objective value $\sigma(N_\Pi)$. Since property (3) is satisfied (by CSIPP), expanding a node cannot spawn a new node with a better objective value. Therefore, the objective value of encountered CT nodes increases monotonically. This means that CCBS will encounter any optimal solution before encountering any sub-optimal solution.

CCBS finds and returns the first solution that it encounters, which must be the optimal solution among all reachable solutions (which includes all valid solutions). Therefore, if the branching rule is sound then CCBS is sound.

□

Theorem 3.3 (Solution completeness)**.** *CCBS is solution complete if it is sound and guaranteed to terminate.*

Proof. CCBS's soundness ensures that all solutions are reachable from the CT root. CCBS has exactly one termination condition: when it selects a CT node N such that N_Π is a solution, then N_Π is returned. CCBS continues expanding nodes until this condition is met. Since all solutions are reachable from the CT root where CCBS begins its search, and CCBS is guaranteed to terminate, it must eventually encounter and return a solution. Therefore, CCBS is solution complete if it is sound and guaranteed to terminate. □

Note that Theorem 3.2 shows that CCBS is sound if the branching rule is sound. However, it does not show that CCBS is unsound if the branching rule is unsound. Thus, the branching rule being sound is a *sufficient but not necessary* condition for CCBS to be sound.

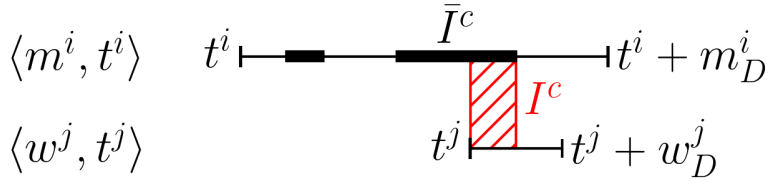


Figure 4: A timeline of conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with conflict interval I^c (red), the set of intersection intervals $\bar{\mathbb{I}}$ (black bars along the move action's timeline), and the intersection interval $\bar{I}^c$.

3.2. Collision Interval and Intersection Interval

We now formally define the *collision interval* and *intersection interval*, which will be useful in Section 5 when introducing δ-BR. Given a conflict $\langle\langle a^i, t^i\rangle, \langle a^j, t^j\rangle\rangle$, the collision interval I^c is the first contiguous time interval during which agents i and j are in collision with each other while executing their respective timed actions. Such a collision interval must exist for any move-move or move-wait conflict, otherwise it is not a conflict. On the other hand, an intersection interval $\bar{I}^c$ exists only for a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$. Informally, for such a move-wait conflict, $\bar{I}^c$ is the time interval during which i and j are in collision if j remains stationary at vertex $v(w^j)$ for all time instead of only during the time when executing $\langle w^j, t^j\rangle$. If multiple such intervals exist, then $\bar{I}^c$ is the interval containing I^c.

More concretely, let a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with collision interval I^c be given (illustrated in Figure 4). Suppose now that agent j waits at $v(w^j)$ for all time instead of only during $[t^j, t^j + w_D]$. Let all maximally connected time intervals when the agents are in collision be collected in the *set of intersection intervals* $\bar{\mathbb{I}}$. Since we know that the agents collide, during I^c, $\bar{\mathbb{I}}$ contains at least one interval super set of I^c.

Definition 3.4 (Intersection interval)**.** *Given a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with collision interval I^c, let the* intersection interval *be the interval $\bar{I}^c \in \bar{\mathbb{I}}$ such that $I^c \subseteq \bar{I}^c$.*

Figure 4 illustrates the relation between I^c, $\bar{I}^c$ and $\bar{\mathbb{I}}$. It is useful to note that the collision interval I^c is the intersection between j's occupancy of vertex $v(w^j)$ and the intersection interval $\bar{I}^c$,

$$I^c = [t^j, t^j + w^j_D] \cap \bar{I}^c. \tag{5}$$

In the continuous-time setting, we assume all collision intervals, unsafe intervals, and intersection intervals are non-degenerate. Singular intervals (where overlap occurs only at a single time instant) have no practical meaning in the continuous-time setting as such situations can ambiguously be interpreted as either a collision or not a collision. In other words, we adopt the interpretation of no collision occurring when agent volumes overlap during a singular time interval.

4. Analyzing CCBS's Implemented Branching Rule

We use the framework from Section 3 to show that the IBR is not sound. Therefore, CCBS using the IBR does not satisfy the sufficient but not necessary condition for soundness from Theorem 4.2: a sound branching rule. Additionally, since the proofs in [21] for CCBS's soundness rely on property (2) being satisfied, which is not fulfilled under the IBR, those proofs do not

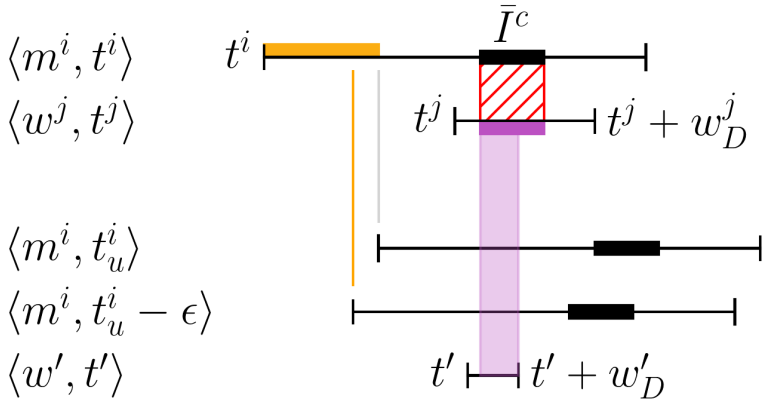


Figure 5: Timeline of a conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with intersection interval, $\bar{I}^c$ and the collision interval in red. The resulting constraints forbid i from executing m^i starting at $[t^i, t^i_u)$ (orange) and j from occupying $v(w^j)$ during $\bar{I}^c$ (purple). The timed action $\langle m^i, t^i_u\rangle$ and two non-conflicting forbidden timed actions $\langle m^i, t^i_u - \epsilon\rangle$ and $\langle w', t'\rangle$ are shown.

apply. Thus, this section provides complementary evidence to the counterexamples in [29] and Section 6 showing that CCBS using the IBR is indeed not sound.

Lemma 4.1 provides intermediate results that are used in Theorem 4.2 to show that the IBR is not sound.

Lemma 4.1 (Collision timing for time-shifted move actions)**.** *Let $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ not be a conflict. If i instead executes $\langle m^i, t^i - \epsilon\rangle$ for some $\epsilon > 0$, then a collision with j executing $\langle w^j, t^j\rangle$ can only occur within the time interval $[t^j + w^j_D - \epsilon, t^j + w^j_D]$.*

Proof. Since j is stationary at $v(w_j)$ throughout $[t^j, t^j + w^j_D]$ and the move action m^i is identical in both $\langle m^i, t^i\rangle$ and $\langle m^i, t^i - \epsilon\rangle$, we can analyze when a collision occurs by comparing agent i's motion in both cases. When agent i executes $\langle m^i, t^i - \epsilon\rangle$, its motion during the interval $[t^j, t^j + w^j_D - \epsilon]$ corresponds exactly to its motion during $[t^j + \epsilon, t^j + w^j_D]$ when executing $\langle m^i, t^i\rangle$. Since no collision occurred during $[t^j + \epsilon, t^j + w^j_D]$ when executing $\langle m^i, t^i\rangle$, and agent j remains stationary at the same location $v(w^j)$, no collision occurs during $[t^j, t^j + w^j_D - \epsilon]$ in the shifted case. Therefore, any collision between i and j executing $\langle m^i, t^i - \epsilon\rangle$ and $\langle w^j, t^j\rangle$ must occur during the remaining time interval $[t^j + w^j_D - \epsilon, t^j + w^j_D]$. □

Theorem 4.2 (Unsoundness of the IBR)**.** *CCBS's IBR is not sound.*

Proof. Consider a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with intersection interval $\bar{I}^c = [\bar{t}^c_1, \bar{t}^c_2)$. From this, the IBR creates the constraints $\langle i, m^i, [t^i, t^i_u)\rangle$ and $\langle j, v(w^j), \bar{I}^c\rangle$. Letting $C_i \Leftrightarrow \langle i, m^i, [t^i, t^i_u)\rangle$ and $C_j \Leftrightarrow \langle j, v(w^j), \bar{I}^c\rangle$, the IBR branches using the constraint set pair $\langle C_i, C_j\rangle$. The move-wait conflict and the constraint intervals $[t^i, t^i_u)$ and $\bar{I}^c$ are illustrated in the upper part of Figure 5.

By construction of C_i, for $0 < \epsilon < t^i_u - t^i$ we know that $\langle i, m^i, t^i_u - \epsilon\rangle \in C_i$. By definition of t^i_u, $\langle m^i, t^i_u\rangle$ and $\langle w^j, t^j\rangle$ do not conflict. It then follows from Lemma 4.1 that any collision between agent i executing $\langle m^i, t^i_u - \epsilon\rangle$ and j executing $\langle w^j, t^j\rangle$ must occur during the time interval $[t^j + w^j_D - \epsilon, t^j + w^j_D]$. The timelines of $\langle m^i, t^i_u\rangle$ and $\langle m^i, t^i_u - \epsilon\rangle$ are illustrated in the lower part of Figure 5.

The constraint set C_j contains one forbidding constraint for each timed action that requires occupying $v(w^j)$ during $\bar{I}^c$. We now aim to find a forbidden timed wait action $\langle w', t' \rangle$ (i.e., $\langle j, w', t' \rangle \in C_j$) that does not conflict with $\langle m^i, t^i_u - \epsilon \rangle$. Let

$$[t', t' + w'_D] \subseteq [t_j, t_j + w^j_D],$$

that is, we restrict our search to timed wait actions that execute during a subset time interval of when $\langle w^j, t^j \rangle$ is executed. Recall that a collision between i and j executing $\langle m^i, t^i_u - \epsilon \rangle$ and $\langle w^j, t^j \rangle$ can only occur during $[t^j + w^j_D - \epsilon, t^j + w^j_D]$. The same also holds for $\langle w', t' \rangle$ since it entails j waiting stationary at the same vertex; a collision between i and j executing $\langle m^i, t^i_u - \epsilon \rangle$ and $\langle w', t' \rangle$ can only occur during $[t^j + w^j_D - \epsilon, t^j + w^j_D]$. We can use this by constraining $\langle w', t' \rangle$ to end before the start of this interval, that is $t' + w'_D < t^j + w^j_D - \epsilon$, and by doing so ensuring that $\langle w', t' \rangle$ and $\langle m^i, t^i_u - \epsilon \rangle$ do not collide:

$$[t', t' + w'_D] \subseteq [t^j, t^j + w^j_D - \epsilon). \tag{6}$$

Finally, to ensure that $\langle j, w', t' \rangle$ is indeed a member of C_j, we require $[t', t' + w'_D]$ to intersect with $\bar{I}^c$. Since ϵ can be arbitrarily small, let $\epsilon < |\bar{I}^c|$. It follows that the interval $[t^j + w^j_D - \epsilon, t^j + w^j_D]$ of size ϵ can only span at most a subset of $\bar{I}^c$. Thus, a non-degenerate subset of $[t^j, t^j + w^j_D - \epsilon)$ is also included in $\bar{I}^c$. Therefore, there exists a $\langle w', t' \rangle$ such that (6) is satisfied and $[t', t' + w'_D) \cap \bar{I}^c \neq \varnothing$. These properties ensure that $\langle w', t' \rangle$ does not conflict with $\langle m^i, t^i_u - \epsilon \rangle$ and that $\langle j, w', t' \rangle \in C_j$. An example of such a timed wait action is illustrated at the bottom of Figure 5.

In conclusion, the timed actions $\langle m^i, t^i_u - \epsilon \rangle$ and $\langle w', t' \rangle$ do not conflict. Therefore, a solution Π can contain both of these timed actions. However, letting $c_i = \langle i, m^i, t^i_u - \epsilon \rangle$ and $c_j = \langle j, w', t' \rangle$, $c_i(\Pi) \vee c_j(\Pi)$ is not satisfied. Thus, $\langle c_i, c_j \rangle$ is not sound, and since $c_i \in C_i$ and $c_j \in C_j$, $\langle C_i, C_j \rangle$ is not sound either. By Lemma 3.1, the IBR is therefore not sound. □

5. A Branching Rule for Sound and Solution-Complete CCBS

In this section, we present δ-BR and prove that CCBS using δ-BR is sound and solution complete. To do so, we first show in Section 5.1 that δ-BR is sound. Termination guarantees are provided by formulating a non-termination condition in Section 5.2 and then showing in Section 5.3 that CCBS under δ-BR does not satisfy this condition. Finally, these results are used in Section 5.4 to conclude that CCBS under δ-BR is sound and solution complete.

δ-BR is based on *shifting constraints* [29] which can be understood by considering a move-wait conflict $\langle \langle m^i, t^i \rangle, \langle w^j, t^j \rangle \rangle$ with intersection interval $\bar{I}^c = [\bar{t}^c_1, \bar{t}^c_2)$. By definition of $\bar{I}^c$, when agent i executes $\langle m^i, t^i \rangle$ then a collision occurs if agent j occupies $v(w^j)$ at any time in $\bar{I}^c$. Suppose now that i executes $\langle m^i, t^i + \Delta t \rangle$ with $\Delta t \in \mathfrak{R}$ instead. Then, the intersection interval will also shift by Δt such that a collision occurs if j occupies $v(w^j)$ at any time in $[\bar{t}^c_1 + \Delta t, \bar{t}^c_2 + \Delta t)$. Figure 6 illustrates the timeline of m^i starting at t^i and incrementally later times. Now, let some δ be given such that $0 \leqslant \delta < |\bar{I}^c|$. Then, if agent i executes $\langle m^i, t^i + \Delta t \rangle$ for any $0 \leqslant \Delta t \leqslant \delta$, we know that the shifted intersection interval $[\bar{t}^c_1 + \Delta t, \bar{t}^c_2 + \Delta t)$ will at least contain the interval $[\bar{t}^c_1 + \delta, \bar{t}^c_2)$ (shown in Figure 6). Therefore, we know that if agent i executes $\langle m^i, t^i + \Delta t \rangle$ for any $0 \leqslant \Delta t \leqslant \delta$, then agent j cannot occupy vertex $v(w^j)$ at any time in $[\bar{t}^c_1 + \delta, \bar{t}^c_2)$ without a collision occurring. This result is useful as it allows us to form a pair of sound constraints from a move-wait conflict.

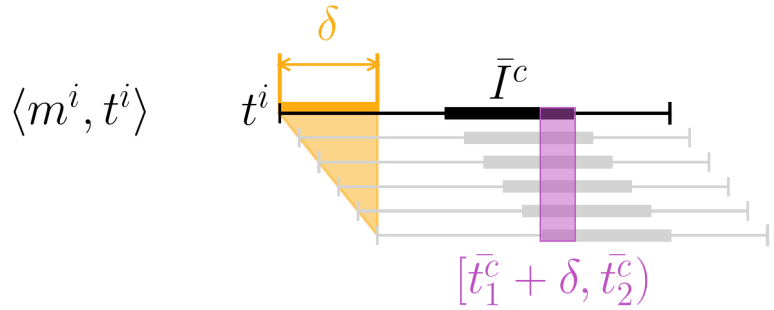


Figure 6: Shifting the start of the timed move action $\langle m^i, t^i \rangle$ also shifts the intersection interval $\bar{I}^c$. Executing m^i at any time in the (orange) interval $[t^i, t^i + \delta)$ will result in a shifted intersection interval which necessarily includes the (purple) interval $[\bar{t}_1^c + \delta, \bar{t}_2^c)$.

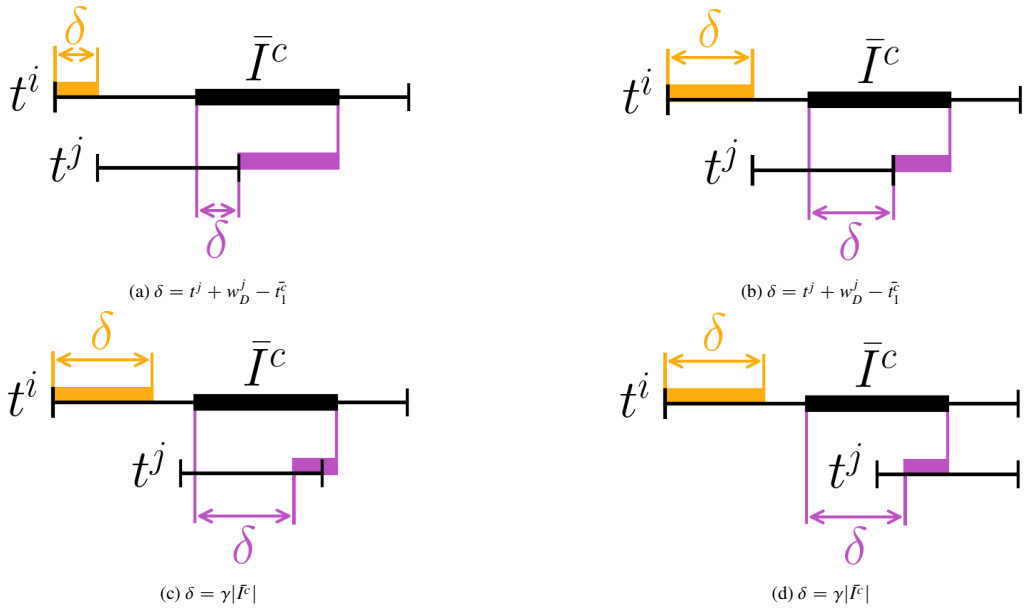
Definition 5.1 (δ-BR)**.** *δ-BR handles move-move and move-wait conflicts differently:*

- *Given a move-move conflict $\langle\langle m^i, t^i\rangle, \langle m^j, t^j\rangle\rangle$, the TBR/IBR is applied: the unsafe intervals $[t^i, t_u^i)$ and $[t^j, t_u^j)$ are found and the pair of motion constraints $\langle i, m^i, [t^i, t_u^i)\rangle$ and $\langle j, m^j, [t^j, t_u^j)\rangle$ are used to branch on. That is, agent i is forbidden from executing $\langle m^i, t\rangle$ for any $t \in [t^i, t_u^i)$ and agent j is forbidden from executing $\langle m^j, t\rangle$ for any $t \in [t^j, t_u^j)$.*
- *Given a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with intersection interval $\bar{I}^c = [\bar{t}_1^c, \bar{t}_2^c)$, let*

$$\delta = \min\left(\gamma|\bar{I}^c|,\ t^j + w_D^j - \bar{t}_1^c\right) \tag{7}$$

where $0 < \gamma < 1$ is a fixed constant. For agent i, the motion constraint $\langle i, m^i, [t^i, t^i + \delta)\rangle$ is created. For agent j, the vertex constraint $\langle j, v(w^j), [\bar{t}_1^c + \delta, \bar{t}_2^c)\rangle$ and motion constraints $\langle j, m, [\bar{t}_1^c + \delta, \bar{t}_2^c)\rangle$ for every $m \in \mathcal{A} : from(m) = v(w^j)$ is created. That is, agent i is forbidden from executing m^i at any time in $[t^i, t^i + \delta)$, and agent j is forbidden from occupying $v(w_j)$ during $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ or executing any move action from $v(w_j)$ at any time in $[\bar{t}_1^c + \delta, \bar{t}_2^c)$.

The fixed constant $\gamma \in (0, 1)$ controls which agent in a move-wait conflict is more constrained: setting γ near 1 increases the constrained interval on the moving agent while setting γ closer to 0 increases the constrained interval on waiting agent. Figure 7 illustrates four different cases where δ-BR (with $\gamma = 0.7$) is applied to a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$. In the first two cases (Figures 7a and 7b) we have $\delta = t^j + w_D^j - \bar{t}_1^c$ since $t^j + w_D^j - \bar{t}_1^c < \gamma|\bar{I}^c|$. Thus, agent j is forbidden from occupying $v(w^j)$ from $t^j + w_D^j$ until the end of $\bar{I}^c$. On the other hand, in the second two cases (Figures 7c and 7d) we have $\delta = \gamma|\bar{I}^c|$. Thus, a non-singular part of agent j's occupation of $v(w^j)$ is included in the constrained interval $[\bar{t}_1^c + \delta, \bar{t}_2^c)$. Furthermore, it is also visible that agent j's constrained interval $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ is smaller than agent i's constrained interval $[t^i, t^i + \delta)$ as a result of $\gamma > 0.5$.



(a) $\delta = t^j + w^j_D - \bar{t}^c_1$

(b) $\delta = t^j + w^j_D - \bar{t}^c_1$

(c) $\delta = \gamma|\bar{I}^c|$

(d) $\delta = \gamma|\bar{I}^c|$

Figure 7: An illustration of δ-BR applied to four different move-wait conflicts, $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$, where the start time t^j of wait action w^j progressively increases for each sub-figure. $\delta = \min(\gamma|\bar{I}^c|,\ t^j + w^j_D - \bar{t}^c_1)$ with $\gamma = 0.7$ is shown; $\delta = t^j + w^j_D - \bar{t}^c_1$ in Figures 7a and 7b, and $\delta = \gamma|\bar{I}^c|$ in Figures 7c and 7d. Agent i's constrained interval $[t^i, t^i + \delta)$ (orange) and agent j's constrained interval $[\bar{t}^c_1 + \delta, \bar{t}^c_2)$ (purple) are shown.

5.1. Soundness

We now show that δ-BR is sound. δ-BR applies TBR/IBR on move-move conflicts, which is shown in Section 4.2 of [21] to satisfy property (2) and is therefore sound. Hence, we refer the reader to there for more details. What remains is to prove the soundness of δ-BR on move-wait conflicts.

Theorem 5.1 (Soundness of δ-BR)**.** *δ-BR is sound when applied to a move-wait conflict.*

Proof. Consider a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ with intersection interval $\bar{I}^c$, on which we apply δ-BR. For agent i, δ-BR creates the motion constraint $\langle i, m^i, [t^i, t^i + \delta)\rangle$. Since the motion constraint is in the form of a shifting constraint, and by (7) we know that $\delta < |\bar{I}^c|$, we know that a collision occurs if agent j occupies vertex $v(w^j)$ at any time during $[\bar{t}^c_1 + \delta, \bar{t}^c_2)$. The motion constraint $\langle i, m^i, [t^i, t^i + \delta)\rangle$ can equivalently be expressed in its decomposed form — a set of forbidding constraints — $C_i = \{\langle i, m^i, t\rangle \mid t \in [t^i, t^i + \delta)\}$.

On the other hand, δ-BR creates for agent j a vertex constraint $\langle j, v(w^j), [\bar{t}^c_1 + \delta, \bar{t}^c_2)\rangle$ and the motion constraints $\langle j, m, [\bar{t}^c_1 + \delta, \bar{t}^c_2)\rangle$ for every $m \in \mathcal{A} : \mathit{from}(m) = v(w^j)$. These can collectively be represented by the set of forbidding constraints

$$C_j = \{\langle j, w, t\rangle \mid v(w) = v(w^j), [t, t + w_D] \cap [\bar{t}^c_1 + \delta, \bar{t}^c_2) \neq \varnothing\}$$
$$\cup \{\langle j, m, t\rangle \mid \mathit{from}(m) = v(w^j), m \in \mathcal{A}, t \in [\bar{t}^c_1 + \delta, \bar{t}^c_2)\}$$

For every constraint $\langle j, a, t\rangle \in C_j$, the forbidden timed action $\langle a, t\rangle$ requires j being located at $v(w^j)$ at some time in $[\bar{t}^c_1 + \delta, \bar{t}^c_2)$; the forbidden timed wait actions require j occupying $v(w^j)$

during some time in $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ and the forbidden timed move actions start at $v(w^j)$ at some time in $[\bar{t}_1^c + \delta, \bar{t}_2^c)$. Thus, if agent i executes any of the actions forbidden by C_i and agent j executes any of the actions forbidden by C_j, then a collision will occur.

The above implies that for every $\langle c_i, c_j \rangle = \langle\langle i, a^i, t^i\rangle, \langle j, a^j, t^j\rangle\rangle \in C_i \times C_j$ the pair of timed actions $\langle\langle a^i, t^i\rangle, \langle a^j, t^j\rangle\rangle$ is a conflict. Thus, any solution must satisfy $c_i \vee c_j$, meaning that $\langle c_i, c_j \rangle$ is sound for every $\langle c_i, c_j \rangle \in C_i \times C_j$ and therefore that $\langle C_i, C_j \rangle$ is sound. By Lemma 3.1, this means that δ-BR is sound on any move-wait conflict. □

To conclude, δ-BR is shown to be sound when applied to move-move conflicts (in [21]) and move-wait conflicts (in Theorem 5.1). As these are the only two conflict types considered, δ-BR is therefore sound.

5.2. A Requirement for Non-termination

To prove that CCBS using δ-BR is solution complete and therefore terminates on a solvable MAPF$_R$ problem, we employ a proof by contradiction. We begin by assuming the opposite: that CCBS never terminates on a solvable MAPF$_R$ problem despite the existence of a solution. This will result in a condition which must hold if CCBS never terminates.

We will establish in this section that if CCBS runs indefinitely, then the CT must contain an infinite sequence of nodes that are continuously selected and branched upon. By analyzing this infinite sequence, we derive a specific requirement that any non-terminating execution must satisfy. It will later be show in Section 5.3 that CCBS using δ-BR does not satisfy this requirement, thus proving termination. Specifically, what we establish here is:

5.2.1) Infinite CT node sequence: There exists an infinite descending path in the constraint tree where each node contains conflicts and has objective value $\leqslant \sigma^*$, with σ^* being the optimal.

5.2.2) Concentration on τ^∞: We define a *trajectory* in Definition 5.2 and a mapping from a CT node to a trajectory in Definition 5.5, and find that infinitely many nodes in the sequence map to the same trajectory τ^∞.

5.2.3) Unbounded Constraint Accumulation: Constraints which determine how nodes map to τ^∞ accumulate unboundedly along the infinite node sequence, yet there must always exist another node mapping to τ^∞ which still maintains an objective value lower than the optimal value.

The final point formulates the requirement on the branching rule for non-termination. We begin by defining trajectories and how plans map to them.

Definition 5.2 (Trajectory)**.** *A trajectory τ is a finite sequence of move actions, $\tau = \langle m_1, m_2 \ldots, m_n \rangle \in \mathcal{A}^n$.*

Each move action $m_i \in \tau$ has a duration $m_{i,D}$. Thus, the duration of τ is $\tau_D = \sum_{k=1}^{n} m_{k,D}$.

Definition 5.3 (Joint Trajectory)**.** *A joint trajectory $\mathcal{T}$ is a set containing one trajectory τ_i for each agent i.*

Definition 5.4 (Plan-Trajectory Mapping)**.** *A plan π maps to a trajectory τ, denoted $\pi \sim \tau$, if τ contains exactly every move action in π in the order that they appear. Similarly, a joint plan Π maps to a joint trajectory $\mathcal{T}$, $\Pi \sim \mathcal{T}$, if it holds for every agent i that $\pi_i \sim \tau_i$ with $\pi_i \in \Pi, \tau_i \in \mathcal{T}$.*

Informally, a trajectory τ describes the sequence of move actions an agent takes when following a plan $\pi \sim \tau$, but without timing details. Multiple plans can therefore map to the same trajectory, as plans can contain various wait actions. For example, the following two plans

- $\pi_i = \langle\langle m_1^i, 0\rangle, \langle w_2^i, 1.5\rangle, \langle m_3^i, 2\rangle, \langle m_4^i, 3\rangle, \langle w_5^i, 4.3\rangle\rangle$ introduced in Figure 2, and
- $\pi_i' = \langle\langle m_1^i, 0\rangle, \langle m_3^i, 1.5\rangle, \langle m_4^i, 2.5\rangle\rangle$

both map to the same trajectory $\tau_i = \langle m_1^i, m_3^i, m_4^i\rangle$. Since each trajectory τ has a duration τ_D, a joint trajectory $\mathcal{T}$ can be evaluated on the objective function, $\sigma(\mathcal{T})$.

Observe that $\tau_D \leqslant \pi_D$ if $\pi \sim \tau$ since π contains at least all the move actions in τ but possibly additional wait actions. From this, and the fact that σ is strictly monotonically increasing with the maximum plan duration, if $\Pi \sim \mathcal{T}$ then

$$\sigma(\mathcal{T}) \leqslant \sigma(\Pi). \tag{8}$$

Finally, we observe that the set $\mathbb{T}$ of all possible joint trajectories $\mathcal{T}$ is countable. This follows from the trajectories τ being finite sequences from the countable set $\mathcal{A}$, meaning that the set of all τ is countable. Since the set of all τ is countable, and a joint trajectory $\mathcal{T}$ comprises a discrete number of trajectories (exactly one per agent), the set $\mathbb{T}$ is countable. We define the subset $\mathbb{T}^c \subset \mathbb{T}$ to contain all joint trajectories with an objective value at or below some fixed constant $c \in \mathfrak{R}$:

$$\mathbb{T}^c = \{\mathcal{T} \in \mathbb{T} \mid \sigma(\mathcal{T}) \leqslant c\}. \tag{9}$$

It holds that $\mathbb{T}^c$ is finite for any $c < \infty$.

5.2.1. Infinite CT node sequence

Since δ-BR is sound (Section 5.1), all solutions (including all optimal solutions) are reachable from the CT root. Non-termination of CCBS implies that at every iteration, a non-solution CT node N is expanded instead of a node representing an optimal solution with objective value σ^*. Since the search is best-first, it must hold that $\sigma(N_\Pi) \leqslant \sigma^*$ otherwise it would not be selected for expansion. Additionally, N_Π must contain a conflict, otherwise N_Π would constitute a valid solution. For non-termination to persist indefinitely, such nodes must be available at every iteration. This leads to the following recursive requirement: at every iteration there must exist at least one node N satisfying the above two conditions, with at least one child node also satisfying the above two conditions, otherwise eventually all nodes satisfying these properties will be depleted and the algorithm will terminate with an optimal solution. In summary, non-termination implies that at every iteration there exists at least one CT node N where

(I) $\sigma(N_\Pi) \leqslant \sigma^*$,

(II) N_Π contains conflicts and is therefore not a solution, and

(III) at least one child node of N satisfies (I) and (II).

From this it follows the existence of an infinite descending path

$$\varsigma = \langle N^1, N^2, \dots\rangle \tag{10}$$

in the CT, where for every $k = 1, 2, \dots$ it holds that N^k satisfies (I)-(III) and N^{k+1} is a child of N^k.

5.2.2. Concentration on τ^∞

Consider a node $N^k \in \varsigma$. When N^k is expanded, CCBS detects a conflict between two plans in N^k_Π and applies a branching rule to create two child nodes. By definition of sequence ς, exactly one of these children continues the sequence as N^{k+1}. For one of the plans involved in the conflict, say $\pi_i \in N^k_\Pi$, a constraint is created for agent i and added to the constraint set N^{k+1}_C. In the context of our analysis, we say that π_i is "branched on" at N^k.

Definition 5.5 (Node-trajectory mapping)**.** *A node $N \in \varsigma$, where π is branched on, maps to the trajectory τ for which $\pi \sim \tau$.*

With Definition 5.5, we can map every one of the infinitely many nodes in ς to a specific trajectory. We now establish that infinitely many nodes in ς map to a specific trajectory, denoted τ^∞. Consider $N \in \varsigma$. The joint plan N_Π satisfies $N_\Pi \sim \mathcal{T}$ for some $\mathcal{T} \in \mathbb{T}$, and N maps to a trajectory $\tau \in \mathcal{T}$. By inequality (8), $\sigma(\mathcal{T}) \leqslant \sigma(N_\Pi)$, and by condition (I), $\sigma(N_\Pi) \leqslant \sigma^*$. Therefore, it follows that $\sigma(\mathcal{T}) \leqslant \sigma^*$ and therefore that $\mathcal{T} \in \mathbb{T}^{\sigma^*}$. Since $\mathbb{T}^{\sigma^*}$ is finite and each $\mathcal{T} \in \mathbb{T}^{\sigma^*}$ contains a finite number of trajectories, the set of all possible trajectories that nodes in ς map to is finite. However, ς contains infinitely many nodes. By the Pigeonhole principle, there exists at least one trajectory τ^∞ which infinitely many nodes in ς map to.

5.2.3. Unbounded Constraint Accumulation

For every branching at a node $N^k \in \varsigma$, a constraint c is created and propagated to all subsequent nodes $N^h \in \varsigma$ for $h > k$, such that $c \in N^h_C$. Since infinitely many nodes in ς map to τ^∞, constraints from branching on plans $\pi \sim \tau^\infty$ accumulate unboundedly along ς, yet there must always exist some $\pi' \sim \tau^\infty$ permitted under these constraints. On the other hand, by condition (I), every $N^k \in \varsigma$ must satisfy $\sigma\left(N^k_\Pi\right) \leqslant \sigma^*$. Since σ is strictly monotonically increasing with the maximum plan duration, there exists a fixed constant $c < \infty$ (determined by σ^*) such that all plans in N^k_Π have duration $\leqslant c$. Therefore, it must hold for all $N^k \in \varsigma$ that $\forall \pi \in N^k_\Pi : \pi_D \leqslant c$, otherwise $\sigma\left(N^k_\Pi\right)$ necessarily grows to eventually violate condition (I).

Finally, we have arrived at our requirement for non-termination: for some fixed $c < \infty$ and trajectory τ^∞, there must exists some plan $\pi \sim \tau^\infty$ with duration $\pi_D \leqslant c$ that is permitted under the constraints accumulated from infinitely many branchings on plans also mapping to τ^∞.

5.3. Solution Completeness

In this section, we prove that CCBS under δ-BR terminates within finite iterations by showing that the non-termination requirement established in Section 5.2 cannot be satisfied, thereby completing our proof by contradiction. This is done by showing that each branching operation reduces the set of feasible execution times for actions in τ^∞ by non-degenerate amounts, eventually making it impossible to construct any plan $\pi \sim \tau^\infty$ with a bounded duration.

The proof is structured in five stages:

5.3.1) Permitted Execution Times: We formalize how constraints produced under δ-BR restrict move execution and vertex occupancy times, establishing the framework for analyzing constraint accumulation.

5.3.2) CSIPP Planning Behavior: We explain how CCBS's underlying path planner, CSIPP, constructs plans, focusing on how it selects execution times from safe intervals.

5.3.3) Non-degenerate Reductions For Move Actions: We show how δ-BR applied to a timed move action (in a move-move or move-wait conflict) reduces feasible move execution times by a non-degenerate amount.

5.3.4) Non-degenerate Reductions For Wait Actions: We show how δ-BR applied to a timed wait action reduces feasible move execution or feasible vertex occupancy times by a non-degenerate amount.

5.3.5) Bounded Plan Impossibility: Using the results from Sections 5.3.3 and 5.3.4, and the fact that all timed actions in a bounded-duration plan must end before a fixed time, we show that the feasible move execution and vertex occupancy times are exhausted after finite iterations of the CCBS search. Therefore, we show that CCBS under δ-BR does not satisfy the non-termination requirement and therefore is guaranteed to terminate.

5.3.1. Permitted Execution Times

To prove that the non-termination requirement cannot be satisfied, we establish a framework to analyze how constraints accumulate and restrict agent movements. At the CT root N^R where the constraint set is empty, $N_C^R = \varnothing$, all agents are permitted to execute all actions at any time $t \geqslant 0$. However, at nodes further down in the CT with non-empty constraint sets, any number of constraints may restrict when agents can execute move actions and occupy vertices. Thus, under the constraints N_C at CT node N, we define for a specific agent i

- $\mathcal{T}_m \subseteq [0, \infty)$ as the set of all times when i is permitted to execute a move action $m \in \mathcal{A}$, and
- $\mathcal{T}_v \subseteq [0, \infty)$ as the set of all times when i is permitted to occupy a vertex $v \in \mathcal{V}$.

Since constraints under the various branching rules restrict move execution and vertex occupancy over time intervals (Definition 5.1), every set $\mathcal{T}_x$ — with x ranging over all moves $m \in \mathcal{A}$ and vertices $v \in \mathcal{V}$ — can be described as the union of a finite number of maximally connected time intervals:

$$\mathcal{T}_x = \bigcup_{h=1}^{N_x} S_x^h, \quad S_x^h = [s_x^h, e_x^h). \tag{11}$$

We refer to S_x^h as a *safe interval* for i to execute a move (when $x = m$) or occupy a vertex (when $x = v$).

5.3.2. CSIPP Planning Behavior

Understanding how CSIPP selects execution times is crucial for proving that constraint additions cause non-degenerate reductions in feasible time intervals. The key insight is that CSIPP, like SIPP [30] which CSIPP is based on, systematically chooses the earliest possible time within a safe interval for each action. This property prevents CSIPP from making arbitrarily small timing adjustments when constraints are added and thereby postponing termination indefinitely.

CSIPP performs an A* [31] search in the vertex-safe interval space. Each search state is represented by a tuple (v, S_v) where agent i is located at $v \in \mathcal{V}$ during the safe interval $S_v = [s_v, e_v) \subseteq \mathcal{T}_v$. Let $t^a \in S_v$ be the time when i arrives at v. Unless state (v, S_v) is a goal state and $e_v = \infty$, agent i must execute a move action to leave v before the end of the safe interval, e_v. For each move action m with $from(m) = v$, agent i can execute m at any time t satisfying

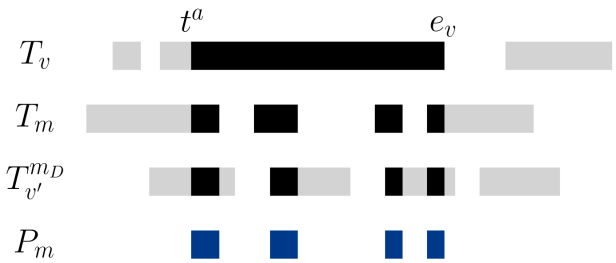


Figure 8: Illustration of the construction of P_m as the intersection between $[t^a, e_v) \subseteq \mathcal{T}_v$, $\mathcal{T}_m$, and $\mathcal{T}_{v'}^{m_D}$.

- $t \in [t^a, e_v)$ (executing m after arriving at v and before it is unsafe to remain there),
- $t \in \mathcal{T}_m$ (executing m when it safe to do so), and
- $t + m_D \in \mathcal{T}_{v'}$ (executing m to arrive at the next node $v' = to(m)$ when it is safe to do so),

In other words, it is safe to execute m at some time $t \in P_m$ where

$$P_m = [t^a, e_v) \cap \mathcal{T}_m \cap \mathcal{T}_{v'}^{m_D} \tag{12}$$

and $\mathcal{T}_{v'}^{m_D} = \{t - m_D \mid t \in \mathcal{T}_{v'}\}$ to compensate for the arrival at v' being m_D after t. By (11), $\mathcal{T}_m$ and $\mathcal{T}_{v'}^{m_D}$ are unions of maximally connected intervals, such that P_m is also a union of maximally connected intervals. Figure 8 illustrates how P_m is formed from the intersection between $[t^a, e_v)$, $\mathcal{T}_m$ and $\mathcal{T}_{v'}^{m_D}$. For every maximally connected interval $[s_m, e_m) \subseteq P_m$, CSIPP creates an edge representing that agent i executes $\langle m, s_m \rangle$, leading to the next state $(v', S_{v'})$ where $S_{v'} \subseteq \mathcal{T}_{v'}$ is the safe interval at v' such that $s_m + m_D \in S_{v'}$.

CSIPP always selects the earliest time s_m in a safe interval $[s_m, e_m) \subseteq P_m$ for i to execute m. This earliest-time selection property is essential for our proof: since the earliest time s_m in $[s_m, e_m)$ is selected, it is not possible to select an earlier time $t' < s_m$ arbitrarily close to s_m for the move action m to be executed, since $t' \notin P_m$.

5.3.3. Non-degenerate Reductions For Move Actions

When δ-BR is applied to a move action $\langle m^i, t^i \rangle \in \pi_i$ in a move-move or move-wait conflict, the motion constraint $\langle i, m^i, [t^i, t') \rangle$ is created. Assuming that π_i was returned from CSIPP, a safe interval $[s_{m^i}, e_{m^i}) \subseteq P_{m^i}$ with $s_{m^i} = t_i$ was identified during the search to construct π_i. We will first show that the constrained interval $[t^i, t')$ is non-degenerate, and then consequently that the new constraint removes either a prefix or the entirety of $[s_{m^i}, e_{m^i})$ from P_{m^i}.

If $\langle m^i, t^i \rangle$ is contained in a move-move conflict, δ-BR forbids the execution of m^i during the unsafe interval $[t^i, t^i_u)$ which is non-degenerate by construction. Thus, in this case we know that the constrained interval $[t^i, t') = [t^i, t^i_u)$ is non-degenerate.

Consider instead that $\langle m^i, t^i \rangle$ is contained in a move-wait conflict $\langle \langle m^i, t^i \rangle, \langle w^j, t^j \rangle \rangle$ with intersection interval $\bar{I}^c = [\bar{t}^c_1, \bar{t}^c_2)$ and collision interval $I^c = [t^c_1, t^c_2)$. δ-BR creates the motion constraint $\langle i, m^i, [t^i, t^i + \delta) \rangle$. The constrained interval $[t^i, t') = [t^i, t^i + \delta)$ is non-degenerate only if δ is bounded away from 0. By (7), δ is the minimum of two positive quantities, $\gamma|\bar{I}^c|$ and $t^j + w^j_D - \bar{t}^c_1$:

- $\gamma|\bar{I}^c|$ is bounded away from zero since $\bar{I}^c$ is non-degenerate and $\gamma \in (0, 1)$ is a fixed, non-zero value.

- From (5) we know that $I^c \subseteq \bar{I}^c$ and $I^c \subseteq [t^j, t^j + w_D^j]$. The former implies that $\bar{t}_1^c \leqslant t_1^c < t_2^c \leqslant \bar{t}_2^c$ where strict inequality holds since I^c is non-degenerate. The latter implies that $t^j \leqslant t_1^c < t_2^c < t_2^{c\prime} \leqslant t^j + t_D^j$ where $t_2^{c\prime} > t_2^c$ is arbitrarily close to t_2^c (to correct for $I^c = [t_1^c, t_2^c)$ not including t_2^c while $[t^j, t^j + t_D^j]$ does include $t^j + t_D^j$). These together give

$$\begin{aligned} \bar{t}_1^c &\leqslant t_1^c < t_2^c < t_2^{c\prime} \leqslant t^j + w_D^j, \\ \bar{t}_1^c &< t^j + w_D^j, \\ 0 &< t^j + w_D^j - \bar{t}_1^c. \end{aligned}$$

  Thus, $t^j + w_D^j - \bar{t}_1^c$ is bounded away from zero.

Since δ is the minimum of two positive quantities which are both bounded away from zero, δ is also bounded away from zero. Therefore, $[t^i, t') = [t^i, t^i + \delta)$ is non-degenerate.

In both cases — when $\langle m^i, t^i \rangle$ is contained in a move-move or move-wait conflict — a constraint $\langle i, m^i, [t^i, t') \rangle$ is created for which we have shown above that $[t^i, t')$ is non-degenerate. This constraint effectively removes $[t^i, t')$ from $\mathcal{T}_{m^i}$. Since $t^i = s_{m^i}$ for some $[s_{m^i}, e_{m^i}) \subseteq P_{m^i}$, by (12), either the entire safe interval $[s_{m^i}, e_{m^i})$ is removed (if $t' \geqslant e_{m^i}$) or a non-degenerate prefix $[s_{m^i}, t')$ is removed from P_{m^i} (if $t' < e_{m^i}$). This is a useful result as it shows that δ-BR applied to any timed move action in a conflict necessarily leads to a non-degenerate time interval being removed from the possible times to execute the move action.

5.3.4. Non-degenerate Reductions For Wait Actions

In this section, we consider the constraints created by δ-BR when applied to a wait action $\langle w^j, t^j \rangle \in \pi_j$ in a move-wait conflict $\langle \langle m^i, t^i \rangle, \langle w^j, t^j \rangle \rangle$ with intersection interval $\bar{I}^c = [\bar{t}_1^c, \bar{t}_2^c)$ and collision interval $I^c = [t_1^c, t_2^c)$. δ-BR creates the vertex constraint $\langle j, v(w^j), [\bar{t}_1^c + \delta, \bar{t}_2^c) \rangle$ and motion constraints $\langle j, m, [\bar{t}_1^c + \delta, \bar{t}_2^c) \rangle$ for each $m \in \mathcal{A} : \mathit{from}(m) = v(w^j)$. It will be shown that there exist two cases: when the overlap between the constrained interval $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ and the wait interval $[t^j, t^j + w_D^j]$ is either singular or non-degenerate. For each case, we will show that either the same conclusions as in Section 5.3.3 can be drawn for the move action $\langle m, t^j + w_D^j \rangle$ following $\langle w^j, t^j \rangle$ in π_j, or a non-degenerate reduction occurs in $\mathcal{T}_{v(w^j)}$.

We begin by showing that the constrained interval $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ is non-degenerate by expanding $\bar{t}_1^c + \delta$ using (7):

$$\begin{aligned} \bar{t}_1^c + \delta &= \bar{t}_1^c + \min\left(\gamma|\bar{I}^c|,\ t^j + w_D^j - \bar{t}_1^c\right) \\ &= \min\left(\bar{t}_1^c + \gamma|\bar{I}^c|,\ \bar{t}_1^c + t^j + w_D^j - \bar{t}_1^c\right) \\ &= \min\left(\bar{t}_1^c + \gamma(\bar{t}_2^c - \bar{t}_1^c),\ t^j + w_D^j\right) \\ &= \min\left((1-\gamma)\bar{t}_1^c + \gamma\bar{t}_2^c,\ t^j + w_D^j\right). \end{aligned} \tag{13}$$

This shows that $\bar{t}_1^c + \delta \leqslant (1-\gamma)\bar{t}_1^c + \gamma\bar{t}_2^c$. Since $0 < \gamma < 1$ is fixed to some value less than 1, it follows that $(1-\gamma)\bar{t}_1^c + \gamma\bar{t}_2^c < \bar{t}_2^c$ and therefore that $\bar{t}_1^c + \delta < \bar{t}_2^c$. Thus, $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ is non-degenerate.

Next, we show that $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ necessarily overlaps with the wait interval $[t^j, t^j + w_D^j]$. To do so, we combine two conditions. First, I^c is assumed in Section 3.2 to be non-degenerate, and by (5) we have $I^c \subseteq [t^j, t^j + w_D^j]$. Thus, it follows that

$$t^j < \bar{t}_2^c. \tag{14}$$

Second, (13) shows that $\bar{t}_1^c + \delta \leqslant t^j + w_D^j$. These two conditions — $t^j < \bar{t}_2^c$ and $\bar{t}_1^c + \delta \leqslant t^j + w_D^j$ — together are sufficient to guarantee overlap between $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ and $[t^j, t^j + w_D^j]$ by at least a singular value. We now consider each of the cases of singular or non-degenerate overlap:

- **Singular:** This implies that $\bar{t}_1^c + \delta = t^j + w_D^j$. In this case, w^j cannot be the final infinite wait action in j's plan: $\bar{t}_1^c < \infty$ (since collisions occur at finite times) and $\delta < \infty$ (since $|\bar{I}^c| < \infty$, see (7)). Thus, the left-hand-side is $< \infty$ implying that the right-hand side containing w_D^j is also $< \infty$. Since w^j does not have infinite duration and is therefore not the final infinite wait action, there exists a subsequent move action $\langle m, t^j + w_D^j \rangle \in \pi_j$ with $\mathit{from}(m) = v(w^j)$. δ-BR creates a constraint forbidding m from being executed during the non-degenerate interval $[\bar{t}_1^c + \delta, \bar{t}_2^c) = [t^j + w_D^j, \bar{t}_2^c)$. The same reasoning and conclusions as in Section 5.3.3 for move actions in move-move and move-wait conflicts applies: a non-degenerate prefix or the entirety of some safe interval $[s_m, e_m) \subseteq P_m$ with $s_m = t^j + w_D^j$ identified during the CSIPP search is removed from P_m.

- **Non-degenerate:** We know that $[t^j, t^j + w_D^j] \subseteq \mathcal{T}_{v(w^j)}$ else the timed wait action $\langle w^j, t^j \rangle$ would not have been returned by CSIPP. This means that the non-degenerate overlap $[\bar{t}_1^c + \delta, \bar{t}_2^c) \cap [t_j, t^j + w_D^j]$ is also a subset of $\mathcal{T}_{v(w^j)}$. In other words, this tells us that a non-degenerate part of $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ exists in $\mathcal{T}_{v(w^j)}$. δ-BR creates a vertex constraint $\langle j, v(w^j), [\bar{t}_1^c + \delta, \bar{t}_2^c) \rangle$ and thereby removes $[\bar{t}_1^c + \delta, \bar{t}_2^c)$ from $\mathcal{T}_{v(w^j)}$. Thus, a non-degenerate interval is removed from $\mathcal{T}_{v(w^j)}$.

5.3.5. Bounded Plan Impossibility

Consider a CT node $N \in \varsigma$ mapping to τ^∞, meaning that a plan $\pi \in N_\Pi$ with $\pi \sim \tau^\infty$ is branched on. To satisfy the non-termination condition (I), $\pi_D \leqslant c$ for some fixed $c < \infty$ which requires that every timed action in π (except the final infinite wait action, which we will momentarily address specifically) ends before c. For simplicity, we suffice with the less restrictive requirement that all timed move actions must *begin* before c. In other words, under the constraints N_C and $\pi_D \leqslant c$ it holds that

$$\forall m \in \tau^\infty : P_m \subseteq [0, c) \tag{15}$$

and

$$\forall v \in \mathbf{v} : \mathcal{T}_v \subseteq [0, c) \tag{16}$$

where $\mathbf{v} = \bigcup_{m \in \tau^\infty} \{\mathit{from}(m), \mathit{to}(m)\}$ contains all vertices visited along τ^∞.

Before proceeding, we address the case of δ-BR being applied to an infinite wait action $\langle w_n^j, t_n^j \rangle \in \pi_j$ in a move-wait conflict $\langle \langle m^i, t^i \rangle, \langle w_n^j, t_n^j \rangle \rangle$, with $\pi_j \sim \tau^\infty$. Let this conflict have intersection interval $\bar{I}^c$. δ-BR creates a vertex constraint $\langle j, v(w_n^j), [\bar{t}_1^c + \delta, \bar{t}_2^c) \rangle$ and several motion constraints (which are not necessary to consider for this analysis). Thus, to satisfy the vertex constraint and also follow a plan mapping to τ^∞ which ends with waiting at $v(w_n^j)$ indefinitely, agent j can only arrive at $v(w_n)$ from time $\bar{t}_2^c$ and later. By (14) we know that $t_n^j < \bar{t}_2^c$. Consequently, the timed move action $\langle m_{n-1}^j, t_{n-1}^j \rangle \in \pi_j$ preceding $\langle w_n^j, t_n^j \rangle$ is no longer valid since it leads to the agent arriving at $v(w_n)$ at t_n^j, that is, before $\bar{t}_2^c$. Instead, m_{n-1}^j can only occur at some time later than $\bar{t}_2^c - m_{n,D}^j$. Since m_{n-1}^j is the last move action in τ^∞, this branching leads to a non-degenerate increase in the duration of all plans $\pi' \sim \tau^\infty$.

In Section 5.3.3, we showed that when δ-BR is applied to a move action $\langle m, t \rangle \in \pi$, then either a non-degenerate prefix or the entirety of a maximally connected interval $[s_m, e_m) \subseteq P_m$ is removed from P_m. Similarly, it was shown in Section 5.3.4 that when δ-BR is applied to a wait action $\langle w, t \rangle \in \pi$, then one of two cases occur. In the first case, the same constraining as when δ-BR is applied to a move action occurs for the move action following $\langle w, t \rangle$ in π. In the second case, a non-degenerate reduction of $\mathcal{T}_{v(w)}$ occurs.

From the above, at every branching there occurs a non-degenerate reduction in one of the finite safe interval sets P_m or $\mathcal{T}_v$ (with $m \in \tau^\infty$ and $v \in \mathbf{v}$) or a non-degenerate increase in the duration of all valid plans $\pi' \sim \tau^\infty$ (when a final infinite wait action is branched on). Therefore, after a finite number of iterations, either one of the safe interval sets P_m or $\mathcal{T}_v$ will be depleted or the duration of valid plans mapping to τ^∞ will grow beyond the fixed $c < \infty$. In either case, eventually — after a finite number of branchings — no plan $\pi \sim \tau^\infty$ exists that can satisfy both the accumulated constraints from these branchings and the plan duration bound $\pi_D < c$.

To conclude, the requirement of non-termination which was established in Section 5.2 cannot be satisfied by CCBS using δ-BR. Therefore, termination is guaranteed after a finite number of iterations.

5.4. Soundness and Solution Completeness

Based on the results from the previous sections, we finalize our proof that CCBS using δ-BR is sound and solution complete.

Theorem 5.2 (Sound and solution complete)**.** *CCBS using δ-BR is sound and solution complete.*

Proof. δ-BR was shown in Section 5.1 to be sound, which means by Theorem 3.2 that CCBS using δ-BR is sound. Additionally, Section 5.3 proved that CCBS using δ-BR terminates within finite iterations. With these soundness and termination guarantees, Theorem 3.3 establishes that CCBS using δ-BR is solution complete. □

6. Experimental Evaluation

We perform experiments with the publicly available CCBS using the IBR (CCBS-IBR), and an otherwise identical CCBS using the proposed δ-BR (CCBS-δ-BR). Both versions are available in our repository[3] along with details in Appendix A regarding necessary modifications to the original code base. A counterexample is introduced in Section 6.1 for which CCBS-δ-BR produces an optimal solution and CCBS-IBR returns a sub-optimal solution, along with explanations as to why this occurs. Thereafter, we compare the solution quality and runtime for the two methods on benchmark problems in Section 6.2. Finally, in Section 6.3 we perform an ablation study on the parameter γ used in δ-BR, motivating the use of $\gamma = 0.9$ unless stated otherwise. We use circular agents with radius $\sqrt{2}/4$ that traverse edges in straight lines at constant speed 1. All experiments are run on a 2025 Mac Studio, 16-core M4 Max CPU, 64 GB RAM, macOS Sequoia (15.3).

6.1. A Counterexample

The example in Figure 9 contains four agents, $i = 1, .., 4$, each with a start vertex $S(i)$ and goal vertex $G(i)$. All agents are circular with radius $r = \sqrt{2}/4$ and traverse edges at constant

[3] https://github.com/Adcombrink/Optimal-Continuous-CBS.

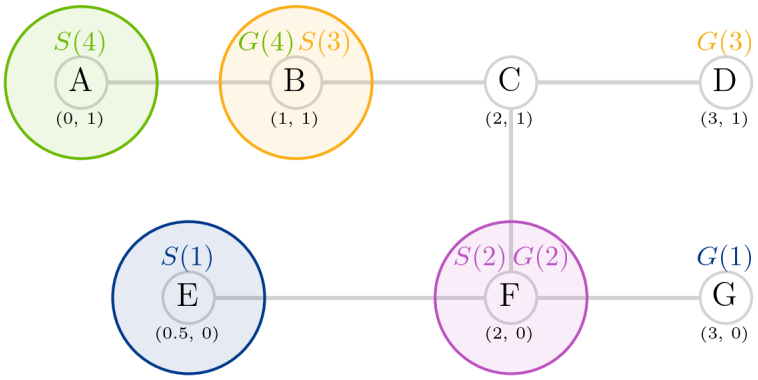


Figure 9: A MAPF$_R$ problem with four agents with radius $r = \sqrt{2}/4$. Each agent $i = 1, .., 4$ is shown at its start vertex $S(i)$ and must reach its goal vertex $G(i)$ without colliding with any other agent.

speed 1 in a straight line. Agent 1 starts at vertex $S(1) = E$ and must pass vertex F to reach its goal $G(1) = G$, therefore, agent 2 with $S(2) = G(2) = F$ must move out of the way. The solution found by both CCBS-IBR and CCBS-δ-BR is for agent 2 to move to C to allow agent 1 to pass. However, agent 2 must avoid a collision with agent 3 which passes vertex C on its way to its goal vertex D. So, there are two options for the interaction between agents 2 and 3 at C:

1. agent 2 waits for agent 3, in turn meaning that agent 1 must wait for agent 2, or
2. agent 3 waits for agent 2, meaning that agent 4 must wait for agent 3.

Due to the difference in length of edges EF and AB, the time that agent 4 must wait for agent 3 is larger than the time that agent 1 must wait for agent 2. Therefore, these two possible solutions have different sum-of-costs, and only the one where agent 2 waits for agent 3 is optimal.

We now examine how CCBS-IBR handles this problem. At the CT root N^R, with $N^R_C = \varnothing$, each agent's initial plan is generated without collision avoidance:

$$N^R_\Pi = \begin{Bmatrix} \pi_1 = \langle\langle EF, 0\rangle\langle FG, 1.5\rangle\langle G, 2.5, \infty\rangle\rangle \\ \pi_2 = \langle\langle F, 0, \infty\rangle\rangle \\ \pi_3 = \langle\langle BC, 0\rangle\langle CD, 1\rangle\langle D, 2, \infty\rangle\rangle \\ \pi_4 = \langle\langle AB, 0\rangle\langle B, 1, \infty\rangle\rangle \end{Bmatrix}$$

where $\langle VU, t\rangle$ denotes a timed move action from V to U starting at time t, and $\langle V, t_1, t_2\rangle$ is a timed wait action at vertex V from t_1 to t_2. Two conflicts appear in N^R_Π: $\langle\langle EF, 0\rangle, \langle F, 0, \infty\rangle\rangle$ and $\langle\langle FG, 1.5\rangle, \langle F, 0, \infty\rangle\rangle$, both between agents 1 and 2. CCBS-IBR branches on the first conflict, with intersection interval $\bar{I}^c = [0.793, 1.5)$. This yields the constraints $\langle 1, EF, [0, \infty)\rangle$ (forbidding agent 1 from ever traversing EF) and $\langle 2, F, \bar{I}^c\rangle$ (forbidding agent 2 from occupying F during $\bar{I}^c$). The first constraint is infeasible, as agent 1 must traverse EF to reach its goal G. Thus, agent 2 must vacate F before $t = 0.793$. However, the earliest time that agent 2 can traverse FC while allowing agent 3 to pass first is at $t = 1$, which is not possible due to the constraint. Therefore, all valid solutions where agent 2 waits for agent 3 are removed from the search. Consequently,

agent 3 must wait for agent 2 to complete its detour via C. The resulting solution is:

$$\Pi = \begin{Bmatrix} \pi_1 = \langle\langle EF, 0\rangle\langle FG, 1.5\rangle\langle G, 2.5, \infty\rangle\rangle \\ \pi_2 = \langle\langle FC, 0\rangle\langle C, 1, 1.5\rangle\langle CF, 1.5\rangle\langle F, 0, \infty\rangle\rangle \\ \pi_3 = \langle\langle B, 0, 1.5\rangle\langle BC, 1.5\rangle\langle CD, 2.5\rangle\langle D, 3.5, \infty\rangle\rangle \\ \pi_4 = \langle\langle A, 0, 1.2\rangle\langle AB, 1.2\rangle\langle B, 2.2, \infty\rangle\rangle \end{Bmatrix}.$$

We now examine how CCBS-δ-BR handles the same conflict. The conflict $\langle\langle EF, 0\rangle, \langle F, 0, \infty\rangle\rangle$ with intersection interval $\bar{I}^c = [0.793, 1.5))$ yields

$$\delta = \min\left(0.5 \cdot |[0.793, 1.5)|, \infty\right) = 0.354$$

and the following set of constraints:

- Agent 1: motion constraint $\langle 1, EF, [0, 0.354)\rangle$,
- Agent 2: vertex constraint $\langle 2, F, [1.17, 1.5)\rangle$ and motion constraints $\langle 2, FE, [1.17, 1.5)\rangle$, $\langle 2, FC, [1.17, 1.5)\rangle$, $\langle 2, FG, [1.17, 1.5)\rangle$.

Unlike the constraints produced by CCBS-IBR, these restrictions exclude only infeasible behaviors while preserving feasible solutions — including the case where agent 2 waits until after agent 3 passes vertex C. The resulting solution is:

$$\Pi = \begin{Bmatrix} \pi_1 = \langle\langle R, 0, 0.5\rangle\langle EF, 0.5\rangle\langle FG, 2\rangle\langle G, 2.5, \infty\rangle\rangle \\ \pi_2 = \langle\langle F, 0, 1\rangle\langle FC, 1\rangle\langle CF, 2\rangle\langle F, 3, \infty\rangle\rangle \\ \pi_3 = \langle\langle BC, 0\rangle\langle CD, 1\rangle\langle D, 2, \infty\rangle\rangle \\ \pi_4 = \langle\langle AB, 0\rangle\langle B, 1, \infty\rangle\rangle \end{Bmatrix}$$

where agent 2 waits at F until agent 3 has traversed C.

The performance using each branching rule is reported in Table 1. There, we see that CCBS-δ-BR yields a solution with lower sum-of-costs, makespan, and computation time than CCBS-IBR. Consequently, this provides additional evidence to that in [29] and Section 4 that CCBS-IBR is not sound. Moreover, the theoretical results in Section 5 ensure that the solution found by δ-BR has optimal sum-of-costs. However, we further strengthen this claim by validating the optimality of CCBS-δ-BR's solution by constructing a tailor-made Satisfiability Modulo Theory (SMT) model for this specific problem and solving it using the SMT solver Z3 [33], confirming after roughly 26 hours of computation that the solution returned by δ-BR is indeed optimal. Details regarding this validation can be found in Appendix B, and animations of CCBS-IBR and CCBS-δ-BR's solutions can be found in the provided repository.

Table 1: Performance comparison between the CCBS-IBR and CCBS-δ-BR on the example in Figure 9.

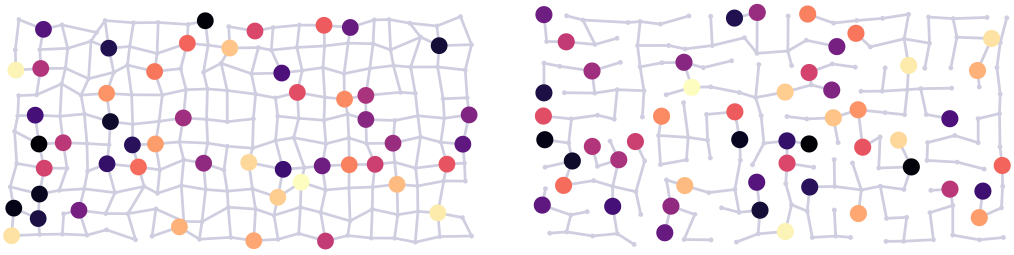
Method	**Sum-of-Costs**	**Makespan**	**Computation Time**
CCBS-δ-BR	9.000	3.000	1.0 ms
CCBS-IBR	10.707	3.500	7.1 ms

6.2. Benchmark Comparison

We compare CCBS-IBR and CCBS-δ-BR on the same 2^k-*neighborhood gridmaps* and *roadmaps* benchmark sets from [21], and additionally on our own randomly generated *gridlike roadmaps*. The benchmarking scheme is the same as in [21]: for each map, a *scenario* defines a list of start and goal vertex pairs. We then create a problem with n agents using the first n start and goal vertex pairs. For each map and scenario, we begin by solving for $n = 2$, and repeatedly incrementing n until a problem cannot be solved within a time limit of 30 seconds.

The 2^k-neighborhood gridmaps — originally introduced in [8] — define a grid of traversable and blocked cells. Each traversable cell is connected to its 2^k neighborhood, provided that a traversal can be done without colliding with a blocked cell. We test with the same parameters as in [21]: $k = 1, ..., 5$ with each of the maps *Den520d*, *Warehouse*, *Rooms*, and *Empty16x16*. The roadmaps are based on the overall shape of Den520d with three levels of density: *Sparse*, *Dense*, and *Mega-dense*. We refer to [21] for additional details regarding these maps.

In addition to the gridmaps and roadmaps, we introduce gridlike roadmaps which are generated with a desired average degree over the map vertices. Two example maps with average degrees of 2.0 and 3.5 are shown in Figure 10. By lowering the average degree, the map becomes more constrained in the number of paths between two vertices. The likelihood of agent interactions consequently increases. These maps are generated from a 20–by–10 vertex grid with spacing 1 by applying zero-mean gaussian noise with standard deviation 0.15 to the vertex positions, and randomly removing edges until a desired average vertex degree is achieved. Edges are only removed if the resulting graph remains connected.



(a) Average vertex degree 3.5.

(b) Average vertex degree 2.0.

Figure 10: Two gridlike roadmaps with different average degrees and agent start positions.

On the gridmap and roadmap benchmark sets, we find no difference in solution quality between CCBS-IBR and CCBS-δ-BR; both methods produce equivalent solutions. However, on the gridlike roadmap benchmark set we find on 11 problems that CCBS-δ-BR produces solutions with a lower sum-of-costs than CCBS-IBR, and equivalent solutions on all remaining problems. Although these 11 problems represent only a fraction of the 15 462 problems that both methods could solve within the time limit, it shows that CCBS-IBR's sub-optimality issues do not only present themselves in tailor-made problems. These results also hint at the difficulty of producing benchmark problems that fully test the limits of these algorithms. Additionally, the 11 problems that trigger CCBS-IBR to return sub-optimal solutions are on maps with an average node degree distribution toward the lower end of the tested range: 1 with average degree 2.1, 3 with average degree 2.2, 4 with average degree 2.4, and 3 with average degree 2.8. This is perhaps unsurprising, as CCBS-IBR's unsoundness is due to how it handles move-wait conflicts; with fewer path options due to a lower average degree, the likelihood that the available paths include a vertex

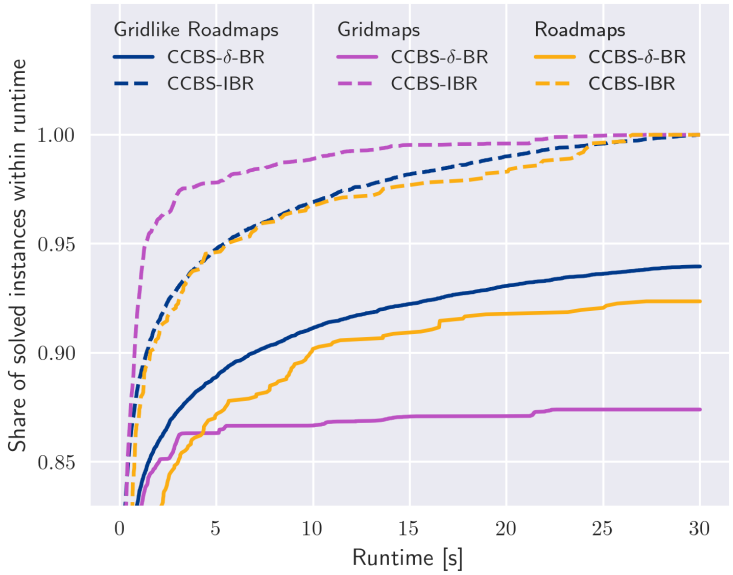


Figure 11: The number of solved problems within a given runtime, for each method on each benchmark set, normalized by the method with most solved problems.

with a stationary agent increases. Over these 11 problems, CCBS-IBR returns mean 0.243 and maximum 0.569 higher sum-of-costs than CCBS-δ-BR. Finally, we note that it is possible for CCBS-IBR— being unsound – to remove all solutions to a problem and therefore never terminate. Such cases are not reflected in our results as we only compare cases where both CCBS-IBR and CCBS-δ-BR return a solution.

We compare the runtime of CCBS-IBR and CCBS-δ-BR in Figure 11 by measuring the number of problems each method is able to solve within a given time limit. To account for different benchmark set sizes, values are normalized to the number of problems CCBS-IBR solves within 30 seconds. Across all benchmark sets, CCBS-IBR is able to solve more problems within the given time limit. This advantage stems from CCBS-IBR's more aggressive handling of move-wait conflicts. While this strategy sometimes removes solutions from the search, it often retains at least one optimal solution while reducing the size of the search space. Consequently, CCBS-IBR generally requires fewer node expansions than CCBS-δ-BR, as illustrated in Figure 12. The difference is most pronounced on gridlike roadmaps, followed by roadmaps, suggesting that move-wait conflicts occur more frequently in these environments compared to gridmaps. However, CCBS-IBR's aggressive pruning comes with trade-offs. In certain cases, as shown above, all optimal solutions are eliminated, leading to sub-optimal results. This means that we are effectively comparing an optimal method (CCBS-δ-BR) to a sub-optimal method (CCBS-IBR). Furthermore, problems where all solutions are eliminated by CCBS-IBR are not reflected in these results, since CCBS-IBR fails to terminate in such cases and contributes to neither the runtime comparison nor the solution quality evaluation above.

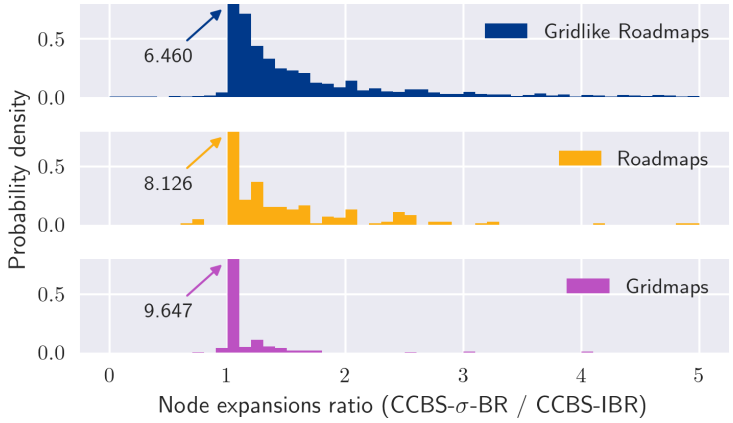


Figure 12: Probability density histogram of the increase in the number of high-level node expansions using CCBS-δ-BR compared to CCBS-IBR. The y-axes are limited to maximum 0.8; the height of the $(1.0, 1.1)$-bins are shown specifically.

6.3. Ablation Study on Parameter γ

The effect of the parameter γ — used to compute δ in CCBS-δ-BR (see Definition 5.1) — on solver runtime is investigated by solving a set of benchmark problems with $\gamma \in \{0.1, 0.3, ..., 0.9\}$ and recording the number of problems solved within a given time limit. The benchmark set contains gridlike roadmaps: for every average degree value $2.0, 2.1, \ldots, 3.5$, we generate 5 maps, each with 5 scenarios. To avoid selection bias, this benchmark is separate from the gridlike roadmap set used in Section 6.2. The results are reported in Figure 13, showing the number of problems that can be solved within 30 seconds for each value of γ and average degree value. We note two things: first, larger γ values lead to more problems being solved. Therefore, on these gridlike roadmaps, larger values of γ reduce the solver runtime. Second, we see that more problems can be solved within the given time limit on maps with a larger average vertex degree. This is likely due to the larger number of possible paths that an agent can take from its start to goal vertex, making these problems less constrained. At these higher average vertex degree values, the different between the performance using different γ values becomes more prominent when compared to lower vertex degrees.

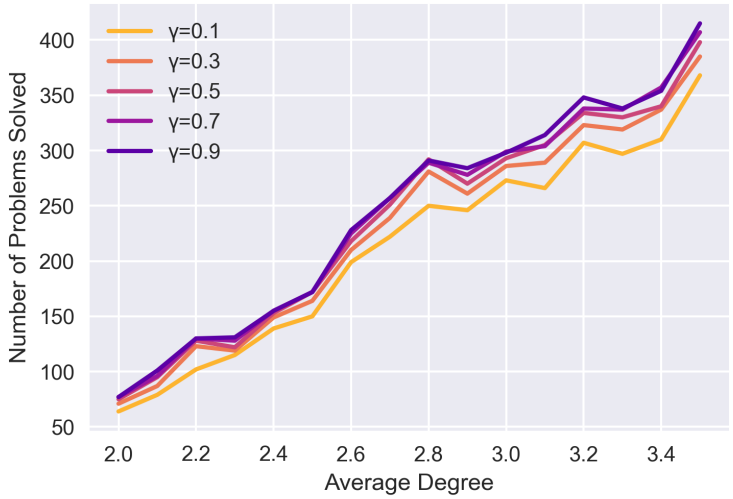


Figure 13: The number of problems solved within a 30 second time limit for various values of γ, shown over a range of average vertex degrees on gridlike roadmaps.

7. Conclusions

This work set out to resolve the recently revealed long-standing gap between the theory and practice of Continuous-time Conflict Based Search (CCBS) for optimal Multi-Agent Path Finding in continuous time (MAPF_R). We revisited the MAPF_R formulation, introduced a new analytical framework for CCBS-style algorithms, and analyzed the publicly available reference implementation of CCBS. The framework provided simple, sufficient conditions for soundness and solution completeness, revealing that the reference implementation violates these conditions — thereby supporting recent counterexamples that challenge the validity of CCBS.

To address this, we proposed a new branching rule (δ-BR) and proved that CCBS using δ-BR satisfies all conditions: it preserves all solutions and guarantees termination within a finite number of iterations on any solvable MAPF_R problem. Thus, CCBS using δ-BR is both sound and solution complete for sum-of-costs, makespan, and any other objective function that is strictly monotonically increasing with respect to the maximum agent arrival time. To our knowledge, this represents the first method for MAPF_R with the same guarantees as Conflict Based Search for the discrete-time MAPF problem. We illustrated the difference between standard CCBS and CCBS using δ-BR with a counterexample, showing that standard CCBS returns a worse solution than CCBS with δ-BR, thereby showing that standard CCBS is not sound. Furthermore, we compared the two approaches on benchmark problems. While standard CCBS often finds solutions faster due to its more aggressive pruning of the search space, this comes at the cost of discarding all optimal solutions in some cases. In contrast, our branching rule guarantees solution optimality without sacrificing termination.

Finally, we show with our provided implementation that δ-BR can be adopted as a drop-in replacement in existing CCBS implementations. Moreover, the analytical framework and non-termination criterion introduced in this work provide analytical tools for reasoning about CCBS-like solvers and their extensions, thereby providing a rigorous foundation for the next generation

of continuous-time multi-agent path finding algorithms.

8. CRediT Authorship Contribution Statement

Alvin Combrink: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft. **Sabino Francesco Roselli:** Software, Writing - Review & Editing, Supervision. **Martin Fabian:** Writing - Review & Editing, Supervision, Funding acquisition.

9. Declaration of Competing Interest

Alvin Combrink reports financial support was provided by Sweden's Innovation Agency. Sabino Francesco Roselli reports financial support was provided by Sweden's Innovation Agency. Martin Fabian reports financial support was provided by Sweden's Innovation Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

10. Acknowledgments

We gratefully acknowledge the Vinnova project CLOUDS (Intelligent algorithms to support Circular soLutions fOr sUstainable proDuction Systems), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Appendix A. Implementation Details

Our implementation of CCBS-IBR[4] closely follows that of the publicly available CCBS implementation[5] from [21]. However, minor modifications have been made to ensure a fair comparison between CCBS-IBR and CCBS-δ-BR[6]. For more details than what we describe here, we refer the reader to the provided repositories.

First, we do not use any *high-level heuristics* or *disjoint splitting* for CCBS-IBR or CCBS-δ-BR, as these are not the focus of this work. Therefore, ensuring compatibility of CCBS-δ-BR with these additional improvements are left for future work. We do however enable *Cardinal Constraints*.

Second, in the implementation from [21], move-wait conflict detection is done using the same method as for move-move conflicts: a binary interval search with a specified precision. CCBS-δ-BR, however, is highly sensitive to small numerical differences between the detection of move-wait conflicts and the computation of $\bar{I}^c = [\bar{t}_1^c, \bar{t}_2^c)$. Specifically, for a move-wait conflict $\langle\langle m^i, t^i\rangle, \langle w^j, t^j\rangle\rangle$ for which $\delta = t^j + w_D^j - \bar{t}_1^c$ (see Definition 5.1), the resulting constraint interval using in the constraints on j is $[\bar{t}_1^c + \delta, \bar{t}_2^c) = [t^j + w_D^j, \bar{t}_2^c)$. This interval overlaps with j's wait action time interval $[t^j, t^j + w_D^j]$ by only a singular time instant, $t^j + w_D^j$. Thus, if the constraint interval is subject to numerical imprecision such that it is computed to $[t^j + w_D^j + \epsilon, \bar{t}_2^c)$ for some

[4] https://github.com/Adcombrink/Optimal-Continuous-CBS/tree/originalCCBS
[5] https://github.com/PathPlanning/Continuous-CBS
[6] https://github.com/Adcombrink/Optimal-Continuous-CBS

small $\epsilon > 0$, then the resulting constraints using this interval will not forbid $\langle w^j, t^j \rangle$ and therefore the detected move-wait conflict may occur again. Our solution to this is straight-forward: we use $\bar{I}^c$ for both constraint creation and move-wait collision detection. An alternative could be to use a fixed precision value as additional margin on the constraints, however, this could be regarded as a form of discretization which we avoid. Besides, since $\bar{I}^c$ is computed analytically (under the assumption that edges are traversed in straight lines, as in all experiments), the collision detection is not subject to a precision value and is therefore more precise. Moreover, for the case of general motion functions, CCBS-δ-BR's numerical sensitivity is easily overcome by using the same computations for collision detection as for constraint creation (such as by pre-computing intersection intervals with, e.g., a binary interval search). Therefore, these modifications do not require agents to only traverse edges in straight lines. As a consequence of using $\bar{I}^c$ in the collision detection for CCBS-δ-BR, the same collision detection is used in CCBS-IBR to avoid any differences in the underlying problem that these two methods solve.

Finally, the computation of $\bar{I}^c$ is re-implemented for more robustness to numerical imprecision, particularly for near-tangent collisions where the collision interval is singular or close to singular.

Appendix B. Validating the Optimality of CCBS-δ-BR's Solution

We validated the optimality of CCBS-δ-BR's solution to the counterexample in Section 6.1 using a tailor-made Satisfiability Modulo Theory (SMT) model with the SMT solver Z3 [33]. After roughly 26 hours of computation, the model returned an equivalent solution with sum-of-costs 9.0, thereby confirming that CCBS-δ-BR's solution is an optimal solution. The exact implementation and results can be found in our repository[7].

The model requires specifying an exact number of timed actions that each agent executes. Thus, the question arises: how many timed actions must be set to ensure that the returned solution is optimal with respect to the MAPF$_R$ problem? In the following, we will establish that any solution with a given sum-of-costs requires at most a certain number of timed move actions; if any more timed move actions are required by a single agent, then the sum of costs of such a solution is necessarily higher than the given sum-of-costs. We find for this specific problem that any solution requiring more than 5 timed move actions necessarily has a sum-of-costs > 9.0. Considering that agents may also execute timed wait actions, we find that the lower bound on the number of timed actions is 11. Furthermore, we motivate that there does not exists an upper bound to the number of timed actions. Thus, with these results, it holds that the retained solution (which is equivalent to the solution from CCBS-δ-BR) is optimal.

Let d_i^{LB} denote an optimistic lower bound on the duration of a plan taking agent i from $S(i)$ to $G(i)$, which is obtained by not considering potential collisions with other agents:

$$d_1^{LB} = 2.5, \qquad d_2^{LB} = 0, \qquad d_3^{LB} = 2.0, \qquad d_4^{LB} = 1.0.$$

For solution Π, let n be the maximum number of timed move actions contained in any single plan $\pi \in \Pi$. Since the minimum move action duration over all move actions in the counterexample is 1.0, a plan with n move actions has duration $\geqslant n$. The sum-of-costs lower bound SOC_Π^{LB}

[7] https://github.com/Adcombrink/Optimal-Continuous-CBS

for Π occurs when one agent executes n moves while others follow their optimistically shortest-duration plans:

$$SOC_{\Pi}^{LB} = \min_{i=1,..4} \left\{ n + \sum_{j \in \{1,..,4\} \setminus \{i\}} d_j^{LB} \right\} = \min \begin{Bmatrix} n + d_2^{LB} + d_3^{LB} + d_4^{LB} = n + 3.0 \\ n + d_1^{LB} + d_3^{LB} + d_4^{LB} = n + 5.5 \\ n + d_1^{LB} + d_2^{LB} + d_4^{LB} = n + 3.5 \\ n + d_1^{LB} + d_2^{LB} + d_3^{LB} = n + 4.5 \end{Bmatrix} = n + 3.0.$$

From this, if any solution has sum-of-costs < 9.0, then $n + 3.0 < 9.0$, implying $n \leqslant 5$. Since consecutive wait actions can be merged, an agent executing n move actions needs at most $n + 1$ wait actions. Therefore, the maximum number of timed actions per agent is $n+n+1 = 2n+1 = 11$. Moreover, since wait actions can be split arbitrarily, enforcing that each agent executes exactly 11 timed actions still allows for an optimal solution to be found.

References

[1] D. Harabor, R. Hechenberger, T. Jahn, Benchmarks for pathfinding search: Iron harvest, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 15, 2022, pp. 218–222.

[2] P. R. Wurman, R. D'Andrea, M. Mountz, Coordinating hundreds of cooperative, autonomous vehicles in warehouses, AI Magazine 29 (1) (2008) 9–9.

[3] S. Varambally, J. Li, S. Koenig, Which mapf model works best for automated warehousing?, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 15, 2022, pp. 190–198.

[4] K. Brown, O. Peltzer, M. A. Sehr, M. Schwager, M. J. Kochenderfer, Optimal sequential task assignment and path finding for multi-agent robotic assembly planning, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2020, pp. 441–447.

[5] R. Morris, C. S. Pasareanu, K. S. Luckow, W. Malik, H. Ma, T. S. Kumar, S. Koenig, Planning, scheduling and monitoring for airport surface operations., in: AAAI Workshop: Planning for Hybrid Systems, 2016, pp. 608–614.

[6] J. Yu, S. LaValle, Structure and intractability of optimal multi-robot path planning on graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 27, 2013, pp. 1443–1449.

[7] O. Salzman, R. Stern, Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, 2020, pp. 1711–1715.

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

[9] H. Ma, Graph-based multi-robot path finding and planning, Current Robotics Reports 3 (3) (2022) 77–84.

[10] M. Erdmann, T. Lozano-Perez, On multiple moving objects, Algorithmica 2 (1) (1987) 477–521.

[11] D. Silver, Cooperative pathfinding, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, Vol. 1, 2005, pp. 117–122.

[12] G. Wagner, H. Choset, M*: A complete multirobot path planning algorithm with performance bounds, in: 2011 IEEE/RSJ international Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 3260–3267.

[13] G. Sharon, R. Stern, M. Goldenberg, A. Felner, The increasing cost tree search for optimal multi-agent pathfinding, Artificial Intelligence 195 (2013) 470–495.

[14] G. Sharon, R. Stern, A. Felner, N. R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artificial Intelligence 219 (2015) 40–66.

[15] G. Sharon, R. Stern, A. Felner, N. Sturtevant, Meta-agent conflict-based search for optimal multi-agent path finding, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 3, 2012, pp. 97–104.

[16] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, E. Shimony, ICBS: The improved conflict-based search algorithm for multi-agent pathfinding, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 6, 2015, pp. 223–225.

[17] M. Barer, G. Sharon, R. Stern, A. Felner, Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 5, 2014, pp. 19–27.

[18] J. Lim, P. Tsiotras, Cbs-budget (cbsb): A complete and bounded suboptimal search for multi-agent path finding, Artificial Intelligence (2025) 104349.

[19] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, S. Koenig, Lifelong multi-agent path finding in large-scale warehouses, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 11272–11281.

[20] M. Tang, Y. Li, H. Liu, Y. Chen, M. Liu, L. Wang, Mgcbs: An optimal and efficient algorithm for solving multi-goal multi-agent path finding problem (2024). arXiv:2404.19518.

[21] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, R. Stern, Multi-agent pathfinding with continuous time, Artificial Intelligence 305 (2022) 103662.

[22] T. Kolárik, S. Ratschan, P. Surynek, Multi-agent path finding with continuous time using sat modulo linear real arithmetic (2023). arXiv:2312.08051.

[23] P. Surynek, Continuous multi-agent path finding via satisfiability modulo theories (smt), in: International Conference on Agents and Artificial Intelligence, Springer, 2020, pp. 399–420.

[24] T. T. Walker, N. R. Sturtevant, A. Felner, Clique analysis and bypassing in continuous-time conflict-based search, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 17, 2024, pp. 152–160.

[25] W. J. Tan, X. Tang, W. Cai, Robust multi-agent pathfinding with continuous time, in: Proceedings of the International Conference on Automated Planning and Scheduling, Vol. 34, 2024, pp. 570–578.

[26] K. Yakovlev, A. Andreychuk, R. Stern, Optimal and bounded suboptimal any-angle multi-agent pathfinding, in: 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2024, pp. 7996–8001.

[27] A. Combrink, S. F. Roselli, M. Fabian, Prioritized planning for continuous-time lifelong multi-agent pathfinding (2025). arXiv:2503.13175.

[28] M. Kulhan, P. Surynek, Multi-agent path finding for indoor quadcopters, in: International Conference on Practical Applications of Agents and Multi-Agent Systems, Springer, 2023, pp. 409–414.

[29] A. Li, Z. Chen, M. Vered, D. Harabor, Revisiting conflict based search with continuous-time (2025). arXiv:2501.07744.

[30] M. Phillips, M. Likhachev, SIPP: Safe interval path planning for dynamic environments, in: 2011 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 5628–5635.

[31] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.

[32] D. Atzmon, R. Stern, A. Felner, G. Wagner, R. Barták, N.-F. Zhou, Robust multi-agent path finding, in: Proceedings of the International Symposium on Combinatorial Search, Vol. 9, 2018, pp. 2–9.

[33] L. De Moura, N. Bjørner, Z3: An efficient SMT solver, in: C. R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer, 2008, pp. 337–340.