BCBS-AA: Bounded Sub-Optimal Conflict-Based Search for Multi-Agent Path Finding with Asynchronous Actions

Xuemian Wu, Shizhe Zhao, Zhongqiang Ren*

Shanghai Jiao Tong University, Shanghai, China
{xuemian.wu, shizhe.zhao, zhongqiang.ren}@sjtu.edu.cn

Abstract

Multi-Agent Path Finding (MAPF) seeks collision-free paths for multiple agents from their respective start locations to their respective goal locations while minimizing path costs. Many MAPF studies rely on a common assumption of synchronized actions, where the actions of all agents start at the same time and always take a time unit, which may limit the applicability of MAPF planners in practice. To bypass this assumption, various algorithms have been developed to handle asynchronous, non-unit-time actions, ranging from optimal to unbounded sub-optimal algorithms. This paper focuses on bounded sub-optimal algorithms for MAPF with asynchronous actions (MAPF-AA) due to their ability to balance solution quality and runtime efficiency. For MAPF, the recent bounded sub-optimal algorithms can intelligently distribute the sub-optimality bound among the agents based on agent-agent collision to achieve fast planning even with tight bounds. We find that directly adapting these techniques to MAPF-AA can degrade their performance due to asynchronous actions. This paper thus develops new techniques to consider asynchronous actions when distributing the sub-optimality bound among the agents and when selecting nodes for expansion during planning. Results show that our approach achieves up to 60% higher success rates and reduces the number of expansions during planning by up to an order of magnitude compared to existing methods.

1 Introduction

Multi-Agent Path Finding (MAPF) plans collision-free paths for a set of agents moving in a shared environment, which is typically represented as a graph, while minimizing path costs (Stern et al. 2019; Sharon et al. 2015). MAPF has been studied extensively due to its broad applications in logistics, and various MAPF planners were developed. Among them, many planners leverage a "synchronized" assumption that all agents start their next actions at the same time and each action has the same unit-time duration. In real-world deployment, however, agents can have different speeds, edge traversal times can vary, and actions are executed asynchronously, which violates this synchronized assumption. Without this assumption, many planners lose their runtime efficiency or solution quality guarantees, which motivates the research on MAPF variants such as Continuous-Time MAPF (Andreychuk et al. 2022), MAPF with Asynchronous Actions (MAPF-AA) (Ren, Rathinam, and Choset 2021), MAPF with Time Uncertainty (Shahar et al. 2021), and MAPF$_R$ (Walker, Sturtevant, and Felner 2018). The key idea in those variants is: the actions of agents can take different amounts of time, where action durations are continuous and can be any non-negative real value, and as a result, the agents may not start and end each of their actions at the same discrete time steps (as shown in Fig. 1(a)).

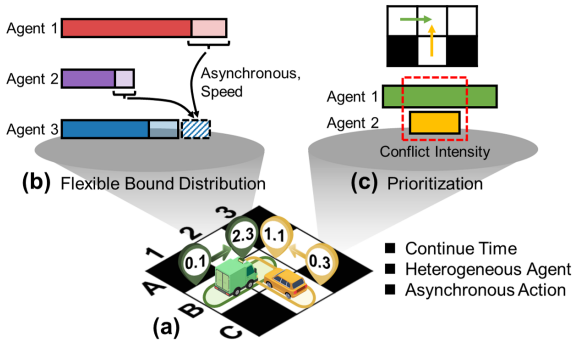

Figure 1: Overview of our method. (a) MAPF-AA exhibits three key characteristics: continuous time, heterogeneous agents, and asynchronous actions. (b) To speed up finding a bounded sub-optimal solution, we design a flexible bound distribution that accounts for asynchronous action and agent speeds. (c) We further introduce conflict intensity to characterize the temporal overlap of conflicts and use it for node prioritization during planning.

This paper focuses on MAPF-AA, which describes agent-agent collision using duration conflicts (Ren, Rathinam, and Choset 2021; Okumura, Tamura, and Défago 2021): when an agent traverses an edge over a time interval, it occupies both end vertices of that edge during that time interval, and no two agents can occupy the same vertex at the same time. Various planners for MAPF-AA have been developed, including optimal planners such as CBS-AA (Wu, Zhao, and Ren 2026), and unbounded sub-optimal planners (Zhou, Zhao, and Ren 2025a,b). However, these optimal planners find optimal solutions yet suffer from limited scalability, and

*Corresponding author.

the unbounded sub-optimal planner handles many agents yet finds highly sub-optimal solutions. This paper is thus interested in developing bounded sub-optimal planners.

To our knowledge, there is no bounded sub-optimal algorithm for MAPF-AA yet. In contrast, for MAPF, various bounded sub-optimal methods have been developed to improve scalability while still providing solution-quality guarantees (Barer et al. 2014; Li, Ruml, and Koenig 2021; Chan et al. 2022, 2025). Most of them follow the Conflict-Based Search (CBS) (Sharon et al. 2015) framework, a two-level search for MAPF, where the high-level detects and resolves agent-agent conflicts by adding constraints to each agent involved in the conflict, and the low-level replans agent's path to satisfy the added constraints by the high-level. These CBS-based bounded sub-optimal methods typically allow a user-specified sub-optimality bound $w \geq 1$ and maintain a lower bound on the optimal solution cost. During the search, they use FOCAL search (Pearl and Kim 1982) to collect all high-level nodes whose costs are within w times this lower bound into a FOCAL list, and expand nodes from this list by prioritizing those with conflict heuristics (e.g., the number of conflicts along the current paths). Beyond conflict heuristics, prior CBS work also reduces search effort by prioritizing which conflicts to resolve (Boyarski et al. 2015) and by using high-level heuristics to guide the search (Li et al. 2019; Felner et al. 2018).

The recent bounded sub-optimal MAPF planners (Chan et al. 2022, 2025) make the bound flexible and can intelligently distribute the bound among the agents based on agent-agent collision to achieve fast planning even when the bound is tight. Properly distributing the bounds among the agents can help keep the newly generated high-level nodes remain in the FOCAL list, so that these new nodes remain eligible for further expansion in subsequent iterations and enable the search to reach a conflict-free solution faster. To distribute the bounds, existing methods (Chan et al. 2025) often count the number of conflicts along the paths and estimate the extra delay needed to satisfy constraints.

Directly transferring these techniques of flexible bound distribution and node prioritization from classical MAPF to MAPF-AA is ineffective because MAPF-AA introduces additional "temporal coupling":

- In MAPF-AA, resolving a conflict often induces a continuous waiting interval whose length depends on both agents' speeds and asynchronous action. The existing methods for MAPF that count the number of conflicts or estimate the delay can lead to inappropriate bound distribution, causing generated high-level nodes to fall outside the FOCAL list, which slows down the search.
- With heterogeneous speeds, the same amount of flexible bound distribution can have very different effects: a fast agent may detour around many conflicts with a small cost increase, while a slow agent may gain little benefit from the same flexible bound distribution.
- For the prioritization of FOCAL, two nodes with the same number of conflicts can differ substantially in how much waiting is needed to resolve them, which impacts whether high-level nodes remain within the FOCAL list.

These challenges make existing flexible bound distribution and FOCAL prioritization rules ineffective for MAPF-AA. We incorporate MAPF-AA characteristics by designing new flexible bound distribution and prioritization methods and fuse them into CBS-AA. We propose BCBS-AA (Bounded CBS-AA), which includes:

- Asynchronous Bound Distribution (ABD), which estimates delay using the time-interval lengths of CBS-AA constraints and distributes the bound accordingly.
- Speed Bound Distribution (SBD), which distributes more bound to faster agents, leveraging their higher detour efficiency under a small cost increase.
- Mixed Bound Distribution (MBD), which combines ABD and SBD and adjusts the maximum allowed bound to avoid generating high-level nodes outside the FOCAL list (as shown in Fig. 1(b)).
- We propose Conflict-Intensity Prioritization (CIP), which introduces conflict intensity measures for node prioritization, based on temporal overlap durations rather than conflict numbers (as shown in Fig. 1(c)).

We evaluate on the MAPF benchmark maps (Stern et al. 2019) under different sub-optimal bounds with ablation studies. Results show that our approach, BCBS-AA (Bounded CBS-AA), achieves up to 60% higher success rates and reduces the number of expansions during planning by up to an order of magnitude than naively adaption of the existing flexible bound distribution methods to MAPF-AA.

2 Problem Formulation

Let $I = \{1, 2, \ldots, N\}$ be the index set of N agents, where each $i \in I$ denote a specific agent. The workspace is represented by an undirected graph $G = (V, E)$, where V is the set of vertices representing reachable locations by the agents, and $E \subset V \times V$ represents the actions that move an agent from one vertex to another adjacent vertex. The travel time of each edge may vary for each agent. Let $\tau^i(u, v) \in \mathbb{R}_{\geq 0}$ denote the travel time for agent i to move from u to v. All agents can wait at a vertex for an arbitrary amount of time.

Let $s^i = (v, t)$ denote the space-time state of agent i, and let $A^i = (s^i_1, s^i_2)$ denote a state transition from s^i_1 to s^i_2. Given $A^i = ((v^i_1, t^i_1), (v^i_2, t^i_2))$, let $\tau(A^i)$ denote the duration of A^i. For any action, $t^i_2 = t^i_1 + \tau(A^i)$. For a *move* action, v^i_2 is adjacent to v^i_1, and $\tau(A^i) = \tau^i(v^i_1, v^i_2)$ is given by input. For a *wait* action, we have $v^i_1 = v^i_2$, and the duration $\tau(A^i) \in \mathbb{R}_{\geq 0}$ is determined by the planner.

We use the same conflict model from existing work (Ren, Rathinam, and Choset 2021; Okumura, Tamura, and Défago 2021), as illustrated below.

Definition 1 (Duration Occupancy)**.** *When agent i performs an action $((v^i_1, t^i_1), (v^i_2, t^i_2))$, both v^i_1 and v^i_2 are occupied by i during the action, which is called Duration Occupancy (DO). Specifically, v^i_1 is occupied at time t^i_1, v^i_2 is occupied at time t^i_2, and both v^i_1, v^i_2 are occupied during (t^i_1, t^i_2). At any time point, a vertex can only be occupied by at most one agent. Multiple agents are in conflict if they both occupy the*

Algorithm 1: CBS-AA

```
1: Ω_c ← ∅, π, g ← LowLevelPlan(Ω_c)
2: Add P_{root,1} = (π, g, Ω_c) to OPEN
3: while OPEN ≠ ∅ do
4:     P = (π, g, Ω_c) ← OPEN.pop()
5:     cft ← DetectConflict(π)
6:     if cft = NULL then return π
7:     Ω ← GenerateConstraints(cft)
8:     for all ω^i ∈ Ω do
9:         Ω' = Ω_c ∪ {ω^i}
10:        π', g' ← LowLevelPlan(Ω')
11:        Add P' = (π', g', Ω') to OPEN
12:    end for
13: end while
14: return failure
```

same vertex for a non-empty time interval, which is referred to as Duration Conflict (DC).

Let V_s, V_g denote the set of start and goal locations of all agents, and $v_s^i \in V_s, v_g^i \in V_g$ denote the start and goal location of agent i respectively. Let $\pi^i(v_s^i, v_g^i) = (s_0^i = (v_s^i, 0), s_1^i, \dots, s_k^i = (v_g^i, t_k^i))$ denote a path from v_s^i to v_g^i. The cost of π^i is t_k^i, denoted as $g(\pi^i)$, which is the time point it reaches v_g^i and can permanently stay after t_k^i without conflict. Multi-Agent Path Finding with Asynchronous Action (MAPF-AA) $P = \langle V, E, V_s, V_g \rangle$ seeks to find a set of conflict-free paths such that (1) each agent $i \in I$ starts at v_s^i and ends at v_g^i; and (2) minimize the sum of costs (SoC) of all agents' paths, i.e., $\min \sum_{i \in I} g(\pi^i)$.

Remark 1 (Duration Settings)**.** *We consider two settings for obtaining $\tau^i(u, v)$ later when presenting our methods.* ***(i) Speed-driven:*** *Each agent has a known speed, and each edge has a length value. The duration $\tau^i(u, v)$ is the ratio of the length over the agent's speed. In this case, a faster agent has a shorter duration for all edges than a slower agent.* ***(ii) General:*** *$\tau^i(u, v)$ are directly provided as input, and can take any positive value for any agents.*

3 Preliminaries

3.1 CBS with Asynchronous Action (CBS-AA)

CBS-AA (Wu, Zhao, and Ren 2026) is a two-level search algorithm that finds an optimal solution for MAPF-AA. Like CBS, CBS-AA (Alg. 1) builds a Constraint Tree (CT), where each CT node maintains a set of paths. The root node contains individually optimal paths computed ignoring other agents. CBS-AA then repeatedly selects and expands the CT node, detects duration conflicts, and branches by adding time-interval constraints. At the low level, CBS-AA replans only the constrained agent with a continuous-time single-agent planner SIPPS-WC (Wu, Zhao, and Ren 2026). CBS-AA returns an optimal solution for any solvable instance.

For a vertex v, CBS-AA defines three action types: $\texttt{IN}(v) = \{A^i \mid v_2^i = v\}$, $\texttt{OUT}(v) = \{A^i \mid v_1^i = v\}$, and $\texttt{WAIT}(v) = \{A^i \mid v_1^i = v_2^i = v\}$. If agent i goes through v, it performs $A_I^i \in \texttt{IN}(v)$, $A_W^i \in \texttt{WAIT}(v)$, and $A_O^i \in \texttt{OUT}(v)$ in order (where wait time is zero if there is no waiting). Conflicts are classified into three types according to the involved actions: IN–IN, OUT–IN, WAIT–IN. Each type induces different constraints, defined over time intervals rather than discrete time steps. To efficiently resolve these conflicts, CBS-AA introduces Constraints on Multiple Actions (CMA), a method for resolving conflicts by propagating constraints to multiple actions using DO. For example, consider an IN–IN conflict $\langle A^i, A^j, v \rangle_I$ between agents i and j, where $A^i = ((v_1^i, t_1^i), (v_2^i, t_2^i))$, $A^j = ((v_1^j, t_1^j), (v_2^j, t_2^j))$ and $v = v_2^i = v_2^j$. Let $\tau_{in}^i, \tau_{out}^i$ (resp., $\tau_{in}^j, \tau_{out}^j$) denote the minimum IN and OUT travel times at v for agent i (resp., j), i.e., $\tau_{in}^i(v) = \min_{e=(u,v) \in E}(\tau^i(u, v))$ and $\tau_{out}^i(v) = \min_{e=(v,u) \in E}(\tau^i(v, u))$. To permit agent j's current action, CMA adds a time-interval constraint $\langle i, * \to v_2^i, [t_1^i,\ t_1^j + \tau_{in}^j + \tau_{out}^j) \rangle_m$ to agent i for forbidding it from starting *any* IN action ($\to$) from any vertex ($*$) into the conflict vertex v_2^i within the interval $[t_1^i,\ t_1^j + \tau_{in}^j + \tau_{out}^j)$. On the other hand, to permit agent i's current action, CMA adds another time-interval constraint $\langle j, * \to v_2^j, [t_1^j,\ t_1^i + \tau_{in}^i + \tau_{out}^i) \rangle_m$ to agent j. These interval constraints explicitly incorporate the agents' asynchronous action when entering and leaving the conflict vertex.

At the low-level, CBS-AA introduces SIPPS-WC, a single-agent planner that explicitly accounts for soft conflicts (when the time interval of the agent's action overlaps with another agent's time interval), which may occur during waiting inside a safe interval. To briefly summarize, a state is $s = (v, t, t_h, c_v^w)$, where c_v^w counts the soft conflicts incurred by waiting at v from t to t_h, and each state also maintains $c(s)$, the number of soft conflicts accumulated along the path to s. Because soft constraints can split a safe interval, multiple states may be generated for the same safe interval with different t and c_v^w. Duplicate detection uses dominance on states with the same (v, t_h, c_v^w): s dominates s' if $s.t \le s'.t$ and $c(s) \le c(s')$; dominated states are pruned, and a better (non-dominated) partial path for a previously expanded state is reinserted into OPEN for possible re-expansion.

3.2 Bounded Sub-Optimal Variants of CBS

Bounded sub-optimal variants of CBS, such as ECBS (Barer et al. 2014) and EECBS (Li, Ruml, and Koenig 2021), trade off optimality for efficiency and guarantee a solution within a sub-optimality bound $w \ge 1$. For each CT node N, the low-level search for agent i maintains the minimum f-value in its $\texttt{OPEN}_L$ list, denoted $f_{\min}^i(N)$, which is a lower bound on the optimal path cost of agent i subject to constraints. The low-level uses FOCAL search and forms $\texttt{FOCAL}_L = \{n \in \texttt{OPEN}_L \mid f(n) \le w \cdot f_{\min}^i(N)\}$, and selects a node from $\texttt{FOCAL}_L$ using a conflict heuristic (e.g., number of conflicts) to guide the search. Let $lb^i(N) = f_{\min}^i(N)$ denote the (best-known) lower bound on the minimum cost of the path that satisfies constraints of agent i. The lower bound of CT node N is then $LB(N) = \sum_{i \in I} lb^i(N)$. Let $\mathcal{N}$ denote the set of all currently maintained CT nodes. The global lower bound is $LB = \min_{N \in \mathcal{N}} LB(N)$, which is a lower bound on the optimal solution cost of the entire problem.

Definition 2 (Globally bounded sub-optimal frontier)**.** *A CT*

node N is globally bounded sub-optimal iff $\frac{\texttt{cost}(N)}{LB} \leq w$, where $\texttt{cost}(N)$ is the cost of node N. The set of globally bounded sub-optimal CT nodes is referred to as the globally bounded sub-optimal frontier.

Accordingly, the high-level FOCAL list is $\texttt{FOCAL}_H = \{N \in \mathcal{N} \mid \texttt{cost}(N) \leq w \cdot LB\}$, i.e., the globally bounded sub-optimal frontier, and nodes in $\texttt{FOCAL}_H$ are prioritized by a conflict heuristic (e.g., number of conflicts). By combining the FOCAL framework with conflict heuristics, bounded sub-optimal CBS variants significantly improve scalability while guaranteeing that the returned solution cost is at most w times the optimal cost.

3.3 Flexible Bound (Flex) Distribution

Flex Distribution (Chan et al. 2022, 2025) is a bounded sub-optimal method for CBS solvers that relaxes per-agent low-level thresholds by redistributing the flexible bound across agents. When branching a CT node $\hat{N}$ to a child node N and replanning a specific agent i, the low-level threshold is increased using the bound contributed by other agents, while preserving the overall sub-optimality bound. The threshold of the low-level search for agent i is

$$\theta^i = w \cdot \max\{f^i_{\min}(N),\, lb^i(\hat{N})\} + \Delta^i \tag{1}$$

where Δ^i is the bound contributed by other agents. Intuitively, Δ^i gives agent i more flexibility during replanning to take detours and avoid conflicts, while the overall solution is still kept within the bounded sub-optimal guarantee. To distribute the bound, several strategies were proposed:

Greedy-based Flex Distribution (GFD) lets $\Delta^i_{\max}$ denote the maximum allowed flexible bound for agent i

$$\Delta^i_{\max} = \sum_{j \neq i} (w \cdot lb^j(\hat{N}) - c^j(\hat{N})) \tag{2}$$

where $c^j(\hat{N})$ is the path cost of agent j in CT node $\hat{N}$. GFD distributes the maximum allowed flexible bound to agent i, i.e., $\Delta^i = \Delta^i_{\max}$. Prior work (Chan et al. 2025) observed that this greedy rule may spend the entire flexible bound when replanning, making the generated child nodes no longer lie on the globally bounded sub-optimal frontier. This motivates more conservative and structured bound distribution rules.

Conflict-based Flex Distribution (CFD) distributes the flexible bound proportionally to the number of conflicts an agent is involved in. If $\Delta^i_{\max} \geq 0$, the distributed bound is $\Delta^i = \rho^i \cdot \Delta^i_{\max}$, where the conflict ratio $\rho^i = \frac{X^i}{X}$, X^i is the number of conflicts involving agent i and X is the total number of conflicts in the current CT node. If $\Delta^i_{\max} < 0$, the method falls back to GFD.

Delay-based Flex Distribution (DFD) distributes part of the bound according to the estimated delay caused by the constraints on the replanned agent. The delay-based bound is $\Delta^i_d = \min\left\{\Delta^i_{\max}, \sum_{\psi \in \Psi^i(N)} d_\psi\right\}$, where d_ψ is the estimated delay required to satisfy constraint ψ. DFD assumes that satisfying one constraint incurs a delay of one timestep. The remaining bound is then distributed proportionally to the conflict ratio ρ^i, resulting in $\Delta^i = \Delta^i_d + \rho^i(\Delta^i_{\max} - \Delta^i_d)$.

Mixed-strategy Flex Distribution (MFD) combines CFD and DFD while ensuring bounded sub-optimality. MFD first computes the flexible bound Δ^i using DFD. The bound is accepted only if the resulting node remains globally bounded sub-optimal, i.e., $w \cdot lb^i(\hat{N}) + \Delta^i + \sum_{j \neq i} c^i(N) \leq w \cdot LB$. If the condition is violated, MFD falls back to CFD. If the condition still does not hold, MFD reduces the bound by recomputing $\Delta^i_{\max}$ using the CT node N_F with $LB(N_F) = LB$ or setting $\Delta^i = 0$, ensuring that the generated CT node remains globally bounded sub-optimal.

Overall, these flexible bound distribution rules bias the distribution to keep the high-level nodes on the globally bounded sub-optimal frontier whenever possible, so they remain eligible for further branching in subsequent iterations and enable the search to reach a conflict-free solution faster.

4 Method

Similar to extending CBS to ECBS, we extend CBS-AA by introducing focal search at both high- and low-levels, yielding ECBS-AA. We incorporate the characteristics of MAPF-AA into bounded sub-optimal CBS techniques, and then modify Alg. 1 at three key points. First, the flexible bound distribution (Secs. 4.1–4.3) is inserted right before each replanning call (Line 10). Second, the high-level node selection (Line 4) is replaced with bounded sub-optimal focal search, prioritizing CT nodes inside the focal list by conflict intensity (Sec. 4.4). Third, we replace the low-level planner in *LowLevelPlan* (Line 10) with bounded sub-optimal focal search, prioritized by conflict intensity (Sec. 4.5).

4.1 Asynchronous Bound Distribution (ABD)

The CMA method in CBS-AA (Sec. 3.1) captures temporal interactions between asynchronous actions. When two agents execute actions that overlap in time and cause a conflict, CBS-AA generates constraints that forbid an agent from executing specific actions within a time interval.

Formally, a CMA constraint for agent i can be written as $\langle i, A^i, [t_s, t_e)\rangle$, which forbids agent i from starting action A^i within the interval $[t_s, t_e)$. The interval encodes the time window in which executing the action would cause a collision due to the temporal overlap between the actions of two agents. Such intervals naturally incorporate the speeds of agents, action durations, and asynchronous execution.

We leverage the CMA intervals to estimate the delay caused by constraints when replanning an agent. For a CT node N that replans agent i, we collect all CMA constraints $\psi \in \Psi^i(N)$ and compute

$$\Delta^i = \begin{cases} \min\{\Delta^i_{\max}, \sum_{\psi \in \Psi^i(N)} |I_\psi|\}, & \Delta^i_{\max} \geq 0, \\ \Delta^i_{\max}, & \text{otherwise.} \end{cases} \tag{3}$$

I_ψ denotes the forbidden time interval of constraint ψ.

The existing DFD for MAPF estimates delay using simple heuristics, where the delay for satisfying a constraint is approximated as one timestep. In contrast, MAPF-AA involves asynchronous actions, where conflicts arise from temporal overlaps between actions with different durations. CMA constraints explicitly encode this information. By incorporating the interval information of CMA constraints,

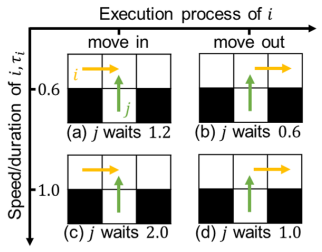

Figure 2: The waiting time required for agent j to yield to agent i varies with agent i's travel time τ_i and its execution process (moving in or moving out of the conflict vertex). (a)-(b) When agent i is fast ($\tau_i = 0.6$), if i is moving in to the conflict vertex, j needs to wait 1.2; if i is moving out, j needs to wait 0.6. (c)-(d) When agent i is slower ($\tau_i = 1.0$), the corresponding waiting times increase to 2.0 for moving in and 1.0 for moving out.

ABD distributes the bound that better reflects the actual temporal impact of conflicts.

For example, in Fig. 2, when yielding to a slower agent, the resulting interval is longer and more bound should be distributed. Similarly, when an agent has just entered the conflicting vertex, resolving the conflict may require a larger delay. These temporal effects are captured by CMA intervals. By estimating delays using CMA constraint intervals, ABD explicitly considers the temporal coupling among the agents and distributes the bound accordingly. Moreover, since CMA intervals are determined by the actual travel times, ABD can be applied to both the speed-driven setting and the general setting described in Remark 1.

4.2 Speed Bound Distribution (SBD)

In MAPF-AA, for the speed-driven setting as mentioned in Remark 1, when agents' speeds are very different, the costs of individually optimal paths can vary significantly across agents even on the same map. This makes the effect of distributing the same amount of bound highly uneven: a given amount of bound may be negligible for a slow agent but sufficient for a fast agent to take meaningful detours and avoid multiple conflicts.

As shown in Fig. 3, consider two agents i (slow) and j (fast) with individually optimal costs $c^i = 6$ and $c^j = 0.6$, respectively, and bound sub-optimal factor $w = 1.5$. Distributing the maximum allowed bound to the slow agent i may still be insufficient to enable a detour that resolves conflicts. In contrast, distributing the bound to the fast agent j can allow it to detour substantially with its own speed, bypass the conflict, and yield a conflict-free solution while staying within the sub-optimal bound.

Motivated by this observation, we propose SBD, which biases the flexible bound distribution toward faster agents. Let v^i denote the speed of agent i, and let $v_{\max} = \max\{v^k \mid k \in I\}$ be the maximum speed among all agents. Given the maximum allowed bound $\Delta^i_{\max}$, SFD distributes

$$\Delta^i = \begin{cases} \frac{v^i}{v_{\max}} \cdot \Delta^i_{\max}, & \Delta^i_{\max} \geq 0, \\ \Delta^i_{\max}, & \text{otherwise.} \end{cases} \tag{4}$$

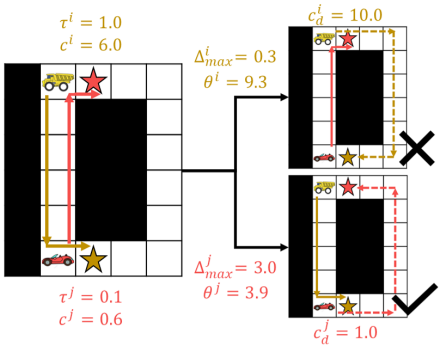

Figure 3: Agent i is slow with $\tau^i = 1.0$ and current path cost $c^i = 6.0$, while agent j is fast with $\tau^j = 0.1$ and current path cost $c^j = 0.6$. The sub-optimal bound is $w = 1.5$. According to Eq. (1) and (2), the maximum allowed bound for i is $\Delta^i_{\max} = 0.3$, yielding a low-level cost threshold $\theta^i = 9.3$; however, a detour path for i has cost $c^i_d = 10.0$ and thus cannot be accepted. In contrast, the maximum allowed bound for j is $\Delta^j_{\max} = 3.0$ with threshold $\theta^j = 3.9$, and a detour path of cost $c^j_d = 1.0$ is feasible, enabling the fast agent to detour and avoid conflicts under the same bound.

Our scaling rule never increases the bound beyond the maximum allowed amount: when $\Delta_{\max,i} \geq 0$, we have $\frac{v_i}{v_{\max}} \in (0, 1]$ and thus $\Delta_i \leq \Delta_{\max,i}$. Therefore, any bounded sub-optimality guarantee that relies on Δ_i being upper-bounded by $\Delta_{\max,i}$ remains valid. When $\Delta_{\max,i} < 0$, we follow the standard fallback and set $\Delta_i = \Delta_{\max,i}$ to maintain feasibility of the bound along the CT branch.

4.3 Mixed Bound Distribution (MBD)

A key challenge of flexible bound distribution is that over-distributing the bound may increase the cost of a child CT node so that it becomes no more globally bounded sub-optimal and falls outside the globally bounded sub-optimal frontier, making it unlikely to be expanded. Therefore, like MFD, MBD explicitly targets the globally bounded sub-optimal frontier and reduces the bound when necessary, while still distributing the bound effectively to resolve conflicts under MAPF-AA.

Given a parent CT node $\hat{N}$ and a child CT node N that replans agent i, we first compute a candidate bound Δ^i and test whether using it would keep the child node globally bounded sub-optimal. If not, we reduce the bound by tightening the maximum allowed bound $\Delta^i_{\max}$ before re-computing Δ^i.

We compute two candidates, ABD Δ^i_{A} (Eq. (3)) and SBD Δ^i_{S} (Eq. (4)). The two candidates provide complementary information. SBD depends only on the speed, so the scaling ratio $\frac{v^i}{v_{\max}}$ is static during the search, while ABD depends on the CMA constraints and thus changes with the temporal conflict along the CT branch. We select $\Delta^i_{\mathrm{M}} = \max\{\Delta^i_{\mathrm{A}}, \Delta^i_{\mathrm{S}}\}$. Intuitively, we start by distributing the bound according to speed. If the estimated asynchronous delay exceeds the speed-based distribution, it indicates that the agent is involved in substantial temporal coupling, and we thus distribute more bound using ABD.

We accept Δ^i_{M} only if using it keeps the child node globally bounded sub-optimal. Using the same sufficient check as in MFD, we require $w \cdot lb^i(\hat{N}) + \Delta^i_{\text{M}} + \sum_{j \neq i} c^j(N) \leq w \cdot LB$. If the condition holds, we use Δ^i_{M} to relax the low-level threshold of agent i. Otherwise, like MFD, we reduce the bound by tightening the maximum allowed bound and recompute Δ^i_{M}.

Specifically, we replace $\Delta^i_{\max}$ with a tighter value computed using the CT node N_F that attains the global lower bound, i.e., $LB(N_F) = LB$: $\tilde{\Delta}^i_{\max} = w \sum_{j \neq i} lb^j(N_F) - \sum_{j \neq i} c^j(N)$. If the recomputed maximum allowed bound is positive and smaller than the previous one (i.e., $0 < \tilde{\Delta}^i_{\max} < \Delta^i_{\max}$), we replace $\Delta^i_{\max}$ with $\tilde{\Delta}^i_{\max}$ and recompute Δ^i_{M}; otherwise, we set $\Delta_i = 0$. This hierarchical reduction aims to keep child nodes on the globally bounded sub-optimal frontier and eligible for expansion as much as possible, while still leveraging the bound to resolve conflicts.

By combining (i) a static speed-based prior and (ii) a dynamic asynchronous delay estimate from CMA, MBD distributes the bound in a way that is both speed-aware and conflict-aware under MAPF-AA. Moreover, the globally bounded sub-optimal check prevents over-distribution that would push CT nodes beyond the globally bounded sub-optimal frontier, improving search focus and efficiency.

4.4 CIP for High-Level

In bounded sub-optimal CBS solvers, the high-level typically orders CT nodes inside FOCAL using conflict-based heuristics such as the number of conflicts, where a conflicting pair contributes 1 and a non-conflicting pair contributes 0. However, in MAPF-AA, conflicts differ not only in number but also in intensity. We define the *intensity* of a conflict as the estimated waiting time needed to resolve it. Intuitively, nodes with smaller conflict intensities are more likely to remain within the globally bounded sub-optimal frontier after branching. Consider a high-level node N with paths $\pi(N)$. For any pair of agents (i, j), let $c^{ij}(N)$ denote the earliest conflict along their paths, and let $\Omega^{ij}(N) = \{\omega^i, \omega^j\}$ be the two CMA constraints generated for resolving $c^{ij}(N)$, where $\omega^i : \langle i, A^i, [t^i_s, t^i_e)\rangle$ and $\omega^j : \langle j, A^j, [t^j_s, t^j_e)\rangle$. We use the interval length as an estimate of the waiting time induced by the constraint, i.e., $\delta^i(c^{ij}) = t^i_e - t^i_s$, $\delta^j(c^{ij}) = t^j_e - t^j_s$. Since resolving a conflict can be achieved by constraining either agent, we take the smaller of the two as the estimated waiting time for conflict c^{ij}:

$$\Delta(c^{ij}) = \min\{\delta^i(c^{ij}),\ \delta^j(c^{ij})\}. \tag{5}$$

We call it "estimated" because an agent may alternatively resolve the conflict by detouring instead of waiting exactly $\Delta(c^{ij})$; nevertheless, the CMA intervals provide a continuous-time measure of the temporal coupling between the two asynchronous actions.

For a CT node N, we compute its conflict intensity score by summing up the estimated waiting times over all earliest pairwise conflicts:

$$H_{\text{int}}(N) = \sum_{i \neq j} \Delta(c^{ij}(N)). \tag{6}$$

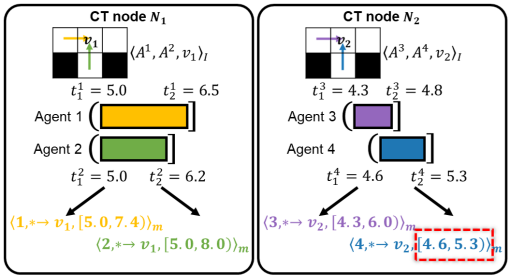

Figure 4: Two CT nodes N_1 and N_2 have an IN−IN conflict at vertices v_1 and v_2, respectively. By Eqs. (5) and (6), the conflict intensity are $H_{\text{int}}(N_1) = 2.4$, and $H_{\text{int}}(N_2) = 0.7$. Our CIP expands N_2 before N_1.

Intuitively, $H_{\text{int}}(N)$ captures not only how many conflicts the node has, but also the expected cost impact of resolving these conflicts via CMA intervals.

As shown in Fig. 4, we prioritize CT nodes in FOCAL based on $H_{\text{int}}(N)$ from the minimum to the maximum. By doing so, the search favors CT nodes with fewer conflicts and smaller conflict intensity, which are more likely to stay on the globally bounded sub-optimal frontier after branching and thus reach a bounded sub-optimal solution faster.

4.5 CIP for Low-Level

The low-level planner is based on the SIPPS-WC in CBS-AA (Wu, Zhao, and Ren 2026). We notice that, under the same number of soft conflicts, the impact can differ: a path that conflicts with a fast agent often requires less additional delay to become conflict-free than a path that conflicts with a slow agent. Therefore, treating every overlap as an identical "+1" can be too coarse for prioritizing low-level states in MAPF-AA.

We introduce an intensity measure for soft conflicts that reflects the continuous-time overlap length. Let the agent i's partial path from the start to state s occupy vertex v over an interval $I^i(v) = [t^i_s, t^i_e]$ (under DO), and let another agent j occupy the same vertex v over interval $I^j(v) = [t^j_s, t^j_e]$. We define the overlap time as

$$\text{ov}(I^i(v), I^j(v)) = I^i(v) \cap I^j(v). \tag{7}$$

For a state s, we define its soft-conflict intensity as the cumulative overlap time along the path from the start to s:

$$\mathcal{I}(s) = \sum_{\text{vertices } v \text{ on the path to } s} \ \sum_{j \neq i} \text{ov}\big(I^i(v), I^j(v)\big). \tag{8}$$

This quantity captures both (i) how many overlaps exist and (ii) how long these overlaps last in continuous time.

As shown in Fig. 5, instead of selecting the state with the smallest number of soft conflicts, we select the state with the smallest soft-conflict intensity $\mathcal{I}(s)$, by replacing the soft-conflict counter $n(s)$ in SIPP-WC with $\mathcal{I}(s)$. This encourages bounded sub-optimal low-level plans whose temporal overlaps with other agents are smaller. In subsequent high-level iterations, resolving such low-intensity overlaps

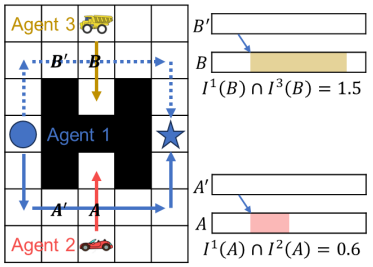

Figure 5: When replanning agent 1, two candidate paths have the same cost but incur different soft-conflict intensities due to different overlap durations with other agents. Using Eqs. (7) and (8), overlap at vertex A for $I^1(A) \cap I^2(A) = 0.6$, and overlap at vertex B for $I^1(B) \cap I^3(B) = 1.5$. Our CIP selects the path with smaller intensity, i.e., the one that conflicts with agent 2.

typically requires a smaller cost increase, making the resulting child CT nodes more likely to remain on the globally bounded sub-optimal frontier and enabling the search to reach a conflict-free solution faster.

4.6 Analysis

Theorem 1. *ECBS-AA with ABD and/or SBD (and their mixed strategy, MBD) is complete and bounded sub-optimal.*

Proof. If $\Delta_{\max,i} < 0$ holds, we follow the proof as in (Chan et al. 2025), since the rule falls back to $\Delta_i = \Delta_{\max,i}$. Thus, we focus on the case where $\Delta_{\max,i} \geq 0$. For ABD, we distribute $\Delta^i = \min\{\Delta^i_{\max}, \Delta^i_d\}$, where Δ^i_d is the delay estimate computed from CMA constraint intervals. Hence $\Delta^i \leq \Delta^i_{\max}$ holds trivially. For SBD, we distribute $\Delta^i = \frac{v^i}{v_{\max}}\Delta^i_{\max}$. Since $v^i \leq v_{\max}$, we have $\frac{v^i}{v_{\max}} \in (0, 1]$, and therefore $\Delta^i \leq \Delta^i_{\max}$. For MBD, we first compute a candidate bound distribution from ABD and SBD and choose $\Delta^i = \max\{\Delta^i_{\text{A}}, \Delta^i_{\text{S}}\}$. Because both candidates are individually upper-bounded by $\Delta^i_{\max}$ for $\Delta^i_{\max} \geq 0$, their maximum also satisfies $\Delta^i \leq \Delta^i_{\max}$. If the globally bounded sub-optimal check fails, the algorithm reduces the effective maximum allowed bound and recomputes Δ^i, or sets $\Delta^i = 0$, which again ensures $\Delta^i \leq \Delta^i_{\max}$.

Therefore, in all cases, the distributed bound never exceeds the maximum allowed bound. By Theorem 2 in (Chan et al. 2025), each generated CT node N satisfies $\texttt{cost}(N) = \sum_{i \in I} c^i(N) \leq w \cdot \sum_{i \in I} lb^i(N) = w \cdot LB(N)$, and the high-level expands only globally bounded sub-optimal CT nodes. Hence, the search is complete and the returned solution is bounded sub-optimal. □

5 Experimental Results

We evaluate our methods on four benchmark maps (Stern et al. 2019): `warehouse-10-20-10-2-1`, `room-64-64-8`, `maze-128-128-10`, and `brc202d`. For each map, we test 25 available random cases (starts and goals). The runtime limit is set to 120 seconds per instance. We adopt the speed-driven setting (Remark 1): agent speeds are randomized between 1 and 20, and the travel time is determined by the edge length (a unit) and the speed of agent i.

Our baseline is ECBS-AA, which extends CBS-AA to the bounded sub-optimal setting in the same spirit as ECBS extends CBS. On top of ECBS-AA, we compare different bound distribution methods under focal weights $w \in \{1.02, 1.05\}$. Besides, we look into the internal running status of algorithms in `brc202d` map with $w = 1.05$ to better understand the search.

To validate the effectiveness of CIP, we run ablations on `brc202d` with $w = 1.05$. Specifically, we test MBD with: (i) CIP enabled only in the low-level, and (ii) enabled in both the low-level and the high-level. We report the success rate and the number of expanded nodes to quantify the impact on search efficiency.

5.1 Comparison for Bound Distribution

Fig. 6 compares the success rates of ECBS-AA (Baseline) and ECBS-AA equipped with different flexible bound distribution strategies under $w = 1.02$ and $w = 1.05$. Overall, the flexible bound distribution substantially improves the success rate across all maps and agent numbers, with the largest gains observed on `maze-128-128-10` and `brc202d`, where the baseline degrades quickly as the number of agents increases. The largest gain in success rate is observed on `maze-128-128-10` with $w = 1.05$. Compared to the existing method, MBD achieves up to an improvement of 64% in success rate (MBD: 80% vs. GFD: 16%).

Among all strategies, our proposed ABD and SBD consistently outperform GFD, indicating that directly distributing the full bound is less effective in MAPF-AA. Compared to CFD and DFD (originally designed for classical MAPF), ABD and SBD further increase the success rate on MAPF-AA instances, showing the benefit of incorporating MAPF-AA specific properties into the flexible bound distribution.

The advantage of ABD and SBD comes from explicitly accounting for asynchronous actions. By distributing the bound in a way that better matches the temporal coupling of conflicts (ABD) and the detour efficiency of fast agents (SBD), the algorithm finds solutions faster under the given bound. Moreover, by combining ABD and SBD, MBD achieves the highest success rates and consistently outperforms all other flexible bound distribution methods across maps and agent numbers.

Fig. 7(a) and (c) report, on `brc202d` with $w = 1.05$, the GB ratio[1] and the number of expanded high-level nodes for different flexible bound distribution strategies. Compared to GFD, ABD expands substantially fewer CT nodes, because it distributes the bound based on CMA constraint intervals rather than spending the entire bound. This yields a more appropriate bound for resolving conflicts while keeping child nodes on the globally bounded sub-optimal frontier, so they

[1]Following (Chan et al. 2025), a generated child node N is counted iff $\texttt{cost}(N) \leq w \cdot LB$, where $LB = \min_{N' \in \mathcal{N}} LB(N')$ is the current global lower bound; the GB ratio is the fraction of generated child nodes that satisfy this test.

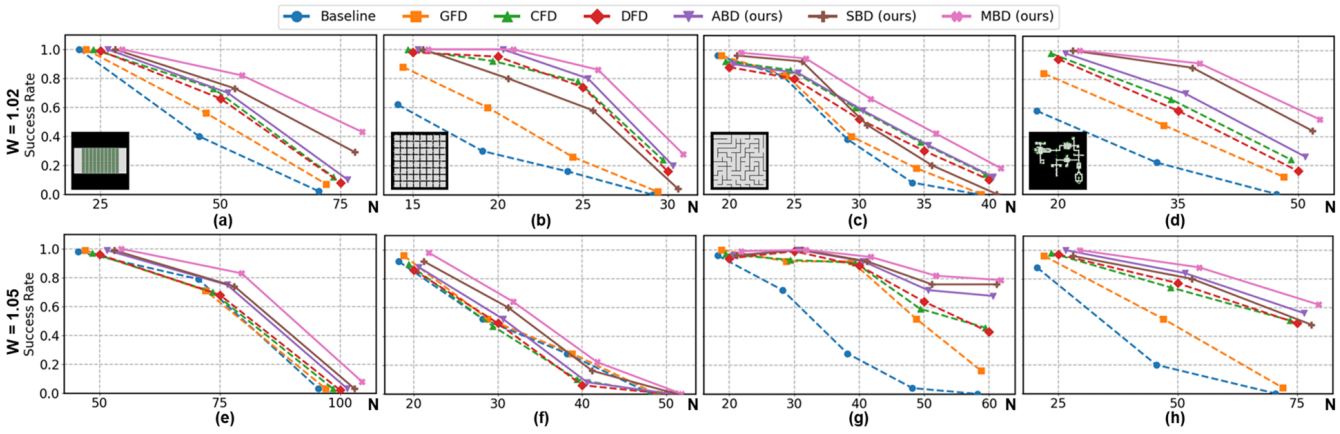

Figure 6: (a)-(d): Success rates of $w = 1.02$. (e)-(h): Success rates of $w = 1.05$. N is the number of agents.

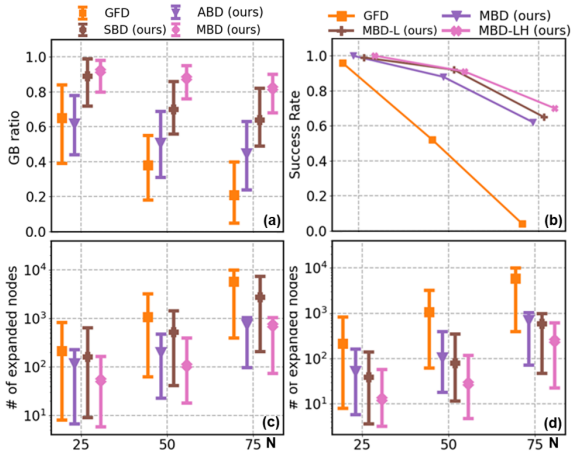

Figure 7: (a), (c): GB ratio and number of expanded high-level nodes for different methods on `brc202d` with $w = 1.05$. (b), (d): Success rate and the number of expanded high-level nodes for CIP on `brc202d` with $w = 1.05$. N is the number of agents.

remain eligible for further branching and the search reaches a conflict-free solution with fewer high-level expansions.

SBD achieves a higher GB ratio than ABD. By distributing more bound to faster agents, SBD enables detours with a smaller cost increase, so generated child nodes are more likely to stay on the globally bounded sub-optimal frontier while conflicts are reduced quickly. In congested instances, detouring only fast agents becomes insufficient; this is reflected by more high-level expansions at $N = 75$, where slow agents also need detours.

MBD combines ABD and SBD. By adjusting the maximum allowed bound when necessary, it achieves the highest GB ratio, so its child nodes are more likely to stay on the globally bounded sub-optimal frontier and remain eligible for further expansion. MBD behaves like SBD early in the search (few constraints), distributing more bound to fast agents for detours; as CMA constraints accumulate and delay estimates grow, it shifts toward ABD, distributing bound to agents with larger delays. This avoids over-focusing on fast agents in dense instances and reduces high-level expansions overall.

5.2 Comparison for Prioritization

Fig. 7(b) and (d) evaluate, on `brc202d` with $w = 1.05$, the impact of CIP on the success rate and number of expanded high-level nodes. We compare (i) MBD without CIP, (ii) MBD-L, which enables CIP only in the low-level, and (iii) MBD-LH, which enables it in the low-level and high-level.

Adding CIP at the low level improves success rates and slightly reduces high-level expansions by preferring bounded sub-optimal paths with smaller soft-conflict intensity. Enabling CIP at both levels yields further gains: MBD-LH achieves the highest success rate and reduces the number of expanded high-level nodes more significantly (about an order of magnitude vs. existing methods). This is because the high level prioritizes nodes with smaller total conflict intensity within the same FOCAL bound, and together with the low-level prioritization it distinguishes cases with similar conflict counts by their overlap durations, keeping more generated nodes on the globally bounded-suboptimal frontier and finding conflict-free solutions with fewer expansions.

6 Conclusion and Future Work

We proposed new flexible bound distributions and node prioritization methods for MAPF-AA to find bounded sub-optimal solutions. Experiments show our approach, BCBS-AA (Bounded CBS-AA), can achieve higher success rates and fewer high-level expansions under tight bounds than existing methods. Future work will exploit CBS-AA's different conflict types induced by asynchronous actions, since their resolutions can incur different cost increases and thus call for conflict-type-aware flexible bound distribution.

7 Acknowledgments

This work was supported by the Natural Science Foundation of China under Grant 62403313, and the Natural Science Foundation of Shanghai under Grant 24ZR1435900.

References

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

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of the international symposium on combinatorial Search*, volume 5, 19–27.

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

Chan, S.-H.; Li, J.; Gange, G.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. Flex distribution for bounded-suboptimal multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9313–9322.

Chan, S.-H.; Phan, T.; Li, J.; and Koenig, S. 2025. New Mechanisms in Flex Distribution for Bounded Suboptimal Multi-Agent Path Finding. In *Proceedings of the International Symposium on Combinatorial Search*, volume 18, 47–55.

Felner, A.; Li, J.; Boyarski, E.; Ma, H.; Cohen, L.; Kumar, T. S.; and Koenig, S. 2018. Adding heuristics to conflict-based search for multi-agent path finding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 28, 83–87.

Li, J.; Felner, A.; Boyarski, E.; Ma, H.; and Koenig, S. 2019. Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. In *IJCAI*, volume 2019, 442–449.

Li, J.; Ruml, W.; and Koenig, S. 2021. Eecbs: A bounded-suboptimal search for multi-agent path finding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 12353–12362.

Okumura, K.; Tamura, Y.; and Défago, X. 2021. Time-Independent Planning for Multiple Moving Agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13): 11299–11307.

Pearl, J.; and Kim, J. H. 1982. Studies in semi-admissible heuristics. *IEEE transactions on pattern analysis and machine intelligence*, (4): 392–399.

Ren, Z.; Rathinam, S.; and Choset, H. 2021. Loosely synchronized search for multi-agent path finding with asynchronous actions. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9714–9719. IEEE.

Shahar, T.; Shekhar, S.; Atzmon, D.; Saffidine, A.; Juba, B.; and Stern, R. 2021. Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research*, 70: 923–954.

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

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

Walker, T. T.; Sturtevant, N. R.; and Felner, A. 2018. Extended Increasing Cost Tree Search for Non-Unit Cost Domains. In *IJCAI*, 534–540.

Wu, X.; Zhao, S.; and Ren, Z. 2026. Conflict-Based Search for Multi Agent Path Finding with Asynchronous Actions. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Zhou, S.; Zhao, S.; and Ren, Z. 2025a. Loosely Synchronized Rule-Based Planning for Multi-Agent Path Finding with Asynchronous Actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 14763–14770.

Zhou, S.; Zhao, S.; and Ren, Z. 2025b. LSRP*: Scalable and Anytime Planning for Multi-Agent Path Finding with Asynchronous Actions (Extended Abstract). In *Proceedings of the International Symposium on Combinatorial Search*, volume 18, 275–276.