Dynamic Agent Grouping ECBS: Scaling Windowed Multi-Agent Path Finding with Completeness Guarantees

Tiannan Zhang[1], **Rishi Veerapaneni**[1], **Shao-Hung Chan**[2], **Jiaoyang Li**[1], **Maxim Likhachev**[1]
[1]Carnegie Mellon University
[2]University of Southern California
{tiannanz, rveerapa, jiaoyanl, mlikhach}@andrew.cmu.edu, shaohung@usc.edu

Abstract

Multi-Agent Path Finding (MAPF) is the problem of finding a set of collision-free paths for a team of agents. Although several MAPF methods that solve full-horizon MAPF have completeness guarantees, very few MAPF methods that plan partial paths have completeness guarantees. Recent work introduced the Windowed Complete MAPF (WinC-MAPF) framework, which shows how windowed optimal MAPF solvers (e.g., SS-CBS) can use heuristic updates and disjoint agent groups to maintain completeness even when planning partial paths. A core limitation of WinC-MAPF is that it requires optimal MAPF solvers. Our main contribution is to extend WinC-MAPF by showing how we can use a bounded suboptimal solver while maintaining completeness. In particular, we design Dynamic Agent Grouping ECBS (DAG-ECBS) which dynamically creates and plans agent groups while maintaining that each agent group solution is bounded suboptimal. We prove how DAG-ECBS can maintain completeness in the WinC-MAPF framework and can improve scalability compared to windowed ECBS which does not have completeness guarantees. More broadly, our work serves as a blueprint for designing more MAPF methods that can use the WinC-MAPF framework.

Code — https://github.com/Rishi-V/Windowed-Complete-MAPF
Extended version — https://arxiv.org/abs/2509.15381

1 Introduction

The rise of capable individual robots has increased the prevalence of autonomous teams of robots. A core problem for managing teams of robots moving in a shared environment is Multi-Agent Path Finding (MAPF), which is the problem of finding a set of collision-free paths, one for each agent, to move from its start location to its goal location. In areas with congestion, MAPF is tough as it requires complex reasoning of all agents' actions, which grows exponentially with respect to the number of agents.

However, the past decade of MAPF research has led to significant progress in the field. Historical methods typically employed greedy planning that could scale to dozens of agents without collisions, but failed in congested scenarios (Erdmann and Lozano-Perez 1987; Silver 2005). Current modern methods can find full-horizon paths (i.e., complete paths from start to goal locations) for hundreds of agents in congested scenarios within seconds. Additionally, several methods provide theoretical completeness guarantees, which means that given enough time and memory, these methods are guaranteed to find a solution if it exists (Barer et al. 2014; Li, Ruml, and Koenig 2021; Veerapaneni, Kusnur, and Likhachev 2023; Okumura 2023).

In situations where the planning time is too short (or where congestion is too large), planning a full-horizon path is infeasible. Thus, existing works in these situations focus on "windowed MAPF," which tries to find only collision-free paths for the next W timesteps (Li et al. 2021b; Jiang et al. 2024). Windowed MAPF, therefore, relaxes the planning problem and makes it more tractable. Although windowed MAPF makes it easier to plan, the main drawback in most windowed approaches is that they are theoretically incomplete. In practice, this means windowed solvers get stuck in deadlock (agents stuck waiting for each other to move out of the way) or livelock (agents revisiting the same locations) as their windowed horizon leads to myopic behavior (Li et al. 2021b).

Recently, the Windowed Complete MAPF (WinC-MAPF) framework has shown how an optimal Windowed MAPF solver, which they term "Action Generator" (AG), can be used with heuristic updates and disjoint agent groups to be theoretically complete (Veerapaneni et al. 2025). Their main insight is that by updating the heuristic values of previously seen agent configurations (i.e., joint state, discussed formally in Section 2.4), the optimal windowed MAPF AG will prefer to visit new configurations and, therefore, get out of deadlock/livelock. They require an optimal AG for their proof of completeness and thus create Single-Step CBS as one such AG, and show how it is able to outperform naive windowed CBS across a variety of maps.

Our main contribution is to broaden the WinC-MAPF framework and show how we can use a bounded suboptimal AG while still maintaining completeness guarantees. In particular, we design Dynamic Agent Grouping ECBS (DAG-ECBS), which dynamically creates and plans agent groups. Critically, DAG-ECBS maintains that each group's windowed solution is bounded suboptimal. We then prove how DAG-ECBS can maintain completeness in the WinC-MAPF framework. We empirically show how it can improve

the success rate compared to naive windowed ECBS without completeness guarantees. More generally, DAG-ECBS and our proof serve as a blueprint for designing more MAPF algorithms within the WinC-MAPF framework.

2 Preliminaries

2.1 Problem Formulation

Multi-Agent Path Finding (MAPF) is the problem of finding collision-free paths for a group of N agents $i = 1, \dots, N$, that takes each agent from its start location $\mathcal{C}_i^{\text{start}}$ to its goal location $\mathcal{C}_i^{\text{goal}}$. Time is discretized into timesteps. We define the MAPF problem in the joint configuration space where a configuration at timestep t, denoted as $\mathcal{C}^t$, is the locations of all agents at timestep t, i.e. $\mathcal{C}^t := [\mathcal{C}_1^t, \mathcal{C}_2^t, ..., \mathcal{C}_N^t]$.

In traditional 2D MAPF, the environment is discretized into grid cells. Agents are allowed to move in any cardinal direction or wait in the same location. A valid solution is a sequence of collision-free configurations $\mathcal{C}^{0:T} := \{\mathcal{C}^0, ..., \mathcal{C}^T\}$ where $\mathcal{C}_i^0 = \mathcal{C}_i^{\text{start}}$ and $\mathcal{C}_i^T = \mathcal{C}_i^{\text{goal}}$ for all agents i, and T is the length of the sequence. Critically, agents must avoid static obstacles as well as vertex collisions (when $\mathcal{C}_i^t = \mathcal{C}_{j\neq i}^t$) and edge collisions (when $\mathcal{C}_i^t = \mathcal{C}_{j\neq i}^{t+1} \wedge \mathcal{C}_i^{t+1} = \mathcal{C}_j^t$) for all timestep t. We define the agent transition cost function as $c(\mathcal{C}_i^t, \mathcal{C}_i^{t+1}) = 1$ unless the agent is resting at its goal (where $c(\mathcal{C}_i^t, \mathcal{C}_i^{t+1}) = 0$ if $\mathcal{C}_i^t = \mathcal{C}_i^{t+1} = \mathcal{C}_i^{\text{goal}}$). We focus on finding a solution $\mathcal{C}^{0:T}$ that minimizes the typical "sum-of-cost" objective $\sum_{i=1}^{N} \sum_{t=0}^{T-1} c(\mathcal{C}_i^t, \mathcal{C}_i^{t+1})$.

Windowed MAPF simplifies the problem by only resolving collisions for the first W timesteps, after which agents are assumed to move along their individual paths (ignoring collisions). Windowed MAPF relies on interleaving partial planning and execution to reach the final goal locations. The windowed MAPF objective is to plan $\mathcal{C}^{0:W}$ which tries to minimize $\sum_{i=1}^{N} (\sum_{t=0}^{W-1} c(\mathcal{C}_i^t, \mathcal{C}_i^{t+1}) + h_i^{BD}(\mathcal{C}_i^W))$ where $h_i^{BD}(\mathcal{C}_i^W)$ is the optimal single-agent cost (computed via the backward Dijkstra (BD) algorithm) for agent i to reach its goal from $\mathcal{C}_i^W$ ignoring other agents. Computing the single-agent backward Dijkstra's heuristic is cheap and commonly used in current MAPF methods (Okumura et al. 2022; Li, Ruml, and Koenig 2021).

In terms of notation, subscripts always denote agent indices. Functions using $\mathcal{C}$ without subscripts are intuitively defined as the sum over agents, e.g., $c(\mathcal{C}^t, \mathcal{C}^{t+1}) := \sum_{i=1}^{N} c(\mathcal{C}_i^t, \mathcal{C}_i^{t+1})$. Finally, we use the shorthand notation $c(\mathcal{C}^0, \mathcal{C}^W) := \sum_{i=0}^{W-1} c(\mathcal{C}^i, \mathcal{C}^{i+1})$ to denote the cost of a windowed path $\mathcal{C}^{0:W}$.

2.2 Related MAPF Works

There are a plethora of full-horizon MAPF methods that have been proposed. Our work builds on Conflict-Based Search (CBS) (Sharon et al. 2015) and Enhanced Conflict-Based Search (ECBS) (Barer et al. 2014). CBS decomposes MAPF into an optimal high-level search over "Constraint Tree" nodes with space-time constraints, and an optimal low-level single-agent search satisfying constraints. ECBS replaces these optimal searches with bounded suboptimal focal searches (Pearl and Kim 1982).

Dynamically Grouping Agents: Several existing works dynamically group and replan agents that collide. Independence Detection in Operator Decomposition (Standley 2010) is an early work that grouped and replanned agents that collided. Meta-Agent CBS (MA-CBS) (Sharon et al. 2012) specifically groups agents in CBS that repeatedly collide into meta-agents and then replans these agents using a joint-state-space MAPF solver such as A* or EPEA* (Goldenberg et al. 2014). Building on MA-CBS, Nested ECBS (NECBS) (Chan et al. 2022) combines the meta-agent merging strategy with bounded-suboptimal search. Instead of using optimal joint-state-space solvers for meta-agents, NECBS employs ECBS to resolve internal conflicts within meta-agents.

Windowed MAPF: There are a variety of MAPF works that utilize windowed planning. Windowed Hierarchical Cooperative A* (Silver 2005) is a windowed variant of Hierarchical Cooperative A* which uses Prioritized Planning (Erdmann and Lozano-Perez 1987). Bounded Multi-Agent A* (Sigurdson et al. 2018) proposes that each agent runs its own limited horizon real-time planner considering other agents as dynamic obstacles. Rolling Horizon Conflict Resolution (RHCR) applies a planning horizon for lifelong MAPF planning and replans paths at repeated intervals (Li et al. 2021b). The winning submission (Jiang et al. 2024) to the Robot Runners competition, due to the tight planning time constraint, leveraged windowed planning of PIBT (Okumura et al. 2022) with LNS (Li et al. 2021a). All these methods mention that deadlock or livelock can occur and is a problem that reduces the success rate in instances with congestion.

Planning and Improving while Executing (Zhang et al. 2024) is a recent work that attempts to quickly generate an initial full plan using LaCAM (Okumura 2023) and then refines it during execution using LNS2 (Li et al. 2022). However, if a complete plan cannot be found, the method resorts to using the best partial path available, making it incomplete.

Recently, the Windowed Complete MAPF framework was proposed, which proved completeness for windowed optimal MAPF solvers (Veerapaneni et al. 2025). After discussing relevant single-agent heuristic search methods, we will revisit this work in more detail.

2.3 Related Single Agent Works

Real-Time Heuristic Search (RTHS) is a problem formulation in single-agent search where the agent is required to iteratively plan and execute partial paths (since it does not have enough time to plan a full path). The key innovation in single-agent RTHS is that the agent updates (increases) the heuristic values of visited states. RTHS algorithms prove how this update prevents deadlock/livelock and enables completeness (Korf 1990; Koenig and Likhachev 2006; Rivera, Baier, and Hernández 2013).

2.4 Windowed Complete MAPF

Our work directly extends the Windowed Complete MAPF (WinC-MAPF) framework. Due to the several moving parts

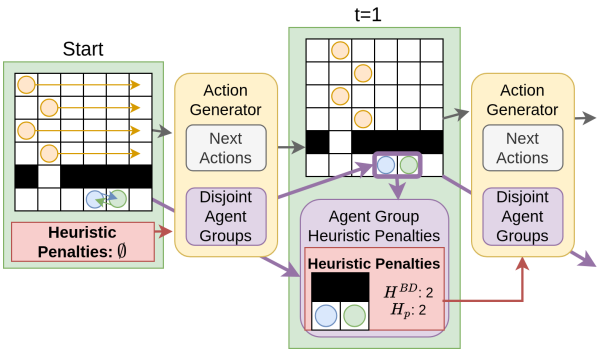


Figure 1: The iterative planning and execution loop of WinC-MAPF with heuristic penalties (see Heuristic Update section below) and disjoint agent groups, figure taken from Veerapaneni et al. (2025).

of WinC-MAPF and the page limit, we provide a reduced summary of the work. Readers unfamiliar with WinC-MAPF are strongly encouraged to read the original paper, which has in-depth and robust explanations and examples.

The WinC-MAPF framework leverages ideas from RTHS to enable completeness for optimal windowed MAPF planners, which they term "Action Generator" (AG). Specifically, given an initial configuration $\mathcal{C}^0$, they call the AG which returns the windowed solution $\mathcal{C}^{0:W*}$ ending at $\mathcal{C}^{W*}$ that optimally minimizes $\sum_{i=1}^{N}(\sum_{t=0}^{W-1} c(\mathcal{C}_i^t, \mathcal{C}_i^{t+1}) + h_i^{BD}(\mathcal{C}_i^W))$, i.e., $\mathcal{C}^{W*}$ is the best windowed configuration. Fig. 1 depicts an overview of the WinC-MAPF framework.

Heuristic Update: Given the optimal windowed solution $\mathcal{C}^{0:W}$, they update the heuristic value of configuration $\mathcal{C}^0$

$$h(\mathcal{C}^0) \leftarrow U(\mathcal{C}^0, \mathcal{C}^W) := \max(h(\mathcal{C}^0), c(\mathcal{C}^0, \mathcal{C}^W) + h(\mathcal{C}^W)), \quad (1)$$

where the heuristic $h(\mathcal{C})$ for all configurations are initially set to $\sum_{i=1}^{N} h_i^{BD}(\mathcal{C}_i)$ and gradually increases as agents visit configurations. For notional convenience, they denote $h(\mathcal{C}) = h^{BD}(\mathcal{C}) + h_p(\mathcal{C})$, where a "heuristic penalty" term $h_p(\mathcal{C}) \geq 0$ denotes an increase in heuristic over the default backward Dijkstra's value due to the update function U.

Disjoint Agent Groups: The main drawback with this approach is that the number of configurations $\mathcal{C}$ grows exponentially with the number of agents. Thus, escaping a local minima (e.g., deadlock/livelock in our case) can require updating the heuristic of an impractical number of configurations. Thus, WinC-MAPF introduces the idea of disjoint agent groups, which essentially decomposes a windowed MAPF problem into sets of independent agent groups. Instead of updating the heuristic of the entire configuration $\mathcal{C}$, they update the heuristics for $\mathcal{C}_{Gr_i}$ where $\mathcal{C}_{Gr_i}$ is the configuration of a group of agents Gr_i. In Fig. 1, this corresponds to detecting and updating the heuristic for the conflicting blue and green agents' grouped configuration as opposed to the entire joint configuration.

Definition 1 (Disjoint Agent Groups)**.** *Given a configuration transition $\mathcal{C}^0 \rightarrow \mathcal{C}^W$, and set of disjoint agent groups $\{Gr_i\}$, each disjoint agent group Gr_i satisfies the property that for each agent j with transition $\mathcal{C}_j^0 \rightarrow \mathcal{C}_j^W$ in Gr_i, there cannot exist another agent in a different group Gr_k that blocked agent j from picking a better path.*

Conceptually, this means that each group's decision to make $\mathcal{C}_{Gr_i}^0 \rightarrow \mathcal{C}_{Gr_i}^W$ is independent of agents in other groups. Given a transition from $\mathcal{C}^0 \rightarrow \mathcal{C}^W$ and a set of disjoint agent groups $\{Gr_i\}$, we apply the update defined in Eq. 1 to each group transition, i.e., $h(\mathcal{C}_{Gr_i}^0) \leftarrow U(\mathcal{C}_{Gr_i}^0, \mathcal{C}_{Gr_i}^W)$. This significantly improves efficiency by increasing the heuristic value of multiple configurations/groups at once. Given these group heuristic values, we compute the heuristic of a joint configuration $h(\mathcal{C})$ via the sum of each group's heuristic, i.e. $h(\mathcal{C}) = \sum_i h(\mathcal{C}_{Gr_i}) = h^{BD}(\mathcal{C}) + \sum_i h_p(\mathcal{C}_{Gr_i})$ for disjoint agent groups $\{Gr_i\}$ at the configuration.

A core component of WinC-MAPF is detecting/computing disjoint agent groups. Instead of computing disjoint groups where agents are not interacting, it is easier to determine coupled agent groups where they could be interacting.

Definition 2 (Coupled Agents)**.** *Given a configuration transition $\mathcal{C} \rightarrow \mathcal{C}^W$, an agent i is coupled with j if j prevents i from choosing a better path or vice-versa.*

Note that coupled agents must be in the same disjoint agent group Gr. Importantly, a disjoint agent group does not need to solely consist of coupled agents, i.e., it is allowed for extra non-coupled agents (e.g., agents independent of all others) to be in a disjoint group.

Action Generator Requirements: Combining everything together, the WinC-MAPF framework proves completeness with an optimal windowed MAPF Action Generator which (1) finds $\arg\min_{\mathcal{C}^W} c(\mathcal{C}^0, \mathcal{C}^W) + h(\mathcal{C}^W)$, and (2) computes disjoint agent groups for $\mathcal{C}^0 \rightarrow \mathcal{C}^W$. The WinC-MAPF framework iteratively calls the AG to obtain a windowed plan, updates the heuristic of each disjoint agent group's configuration, executes the plan, and repeats (Fig. 1).

Single-Step CBS: The WinC-MAPF paper also introduces Single-Step CBS (SS-CBS), which is a single-step optimal windowed AG that is able to incorporate heuristic penalties and compute disjoint agent groups. SS-CBS computes coupled agents (which form disjoint agent groups) by grouping together agents that shared a conflict/constraint during the process of finding the windowed solution. However, incorporating heuristic penalties is trickier as Veerapaneni et al. (2025) prove that they cannot be naively handled in CBS. Conceptually, SS-CBS resolves the problem by delaying incurring a heuristic penalty via a "heuristic" conflict and constraint in the high-level search. A heuristic conflict occurs when agents match a group configuration with a non-zero $h_p(\mathcal{C}_{Gr})$. SS-CBS then allows agents to try to avoid this heuristic penalty by replanning and avoiding $\mathcal{C}_{Gr}$, or forces agents to stay at $\mathcal{C}_{Gr}$ and incur the penalty.

The main restriction with WinC-MAPF is the requirement for an optimal windowed planner. In particular, although SS-CBS has a better success rate than naive windowed CBS, SS-CBS starts timing out when dealing with tougher congestion.

Proof of Completeness: The key to proving completeness in WinC-MAPF is showing that $h(\mathcal{C}_{Gr_i}) \leq h^*(\mathcal{C}_{Gr_i})$ (where

$h^*(\mathcal{C}_{Gr_i})$ is the optimal cost for Gr_i at $\mathcal{C}_{Gr}$ to reach the goal) holds even when updating the heuristic values, i.e., the heuristic of each disjoint agent group configuration $\mathcal{C}_{Gr_i}$ is always admissible. If that holds, then $h(\mathcal{C})$ satisfies

$$h(\mathcal{C}) := \sum_i h(\mathcal{C}_{Gr_i}) \leq \sum_i h^*(\mathcal{C}_{Gr_i}) \leq h^*(\mathcal{C}) \quad (2)$$

for a set of disjoint agent groups Gr_i that cover all agents.

Given this fact, proving completeness follows standard single-agent real-time heuristic search logic. The main intuition is that if the agents never reach the goal, they must be infinitely stuck cycling through a finite set of states. Across every cycle, we can show that at least one heuristic value must increase. This means that the heuristic values in the cycle are getting arbitrarily high, which contradicts heuristic values being admissible. Thus, the agents cannot be stuck in a loop and must eventually reach their goals. Readers are encouraged to read Veerapaneni et al. (2025) for more details about WinC-MAPF in general, and specifically their Appendix B.1 for a formal proof.

3 Dynamic Agent Grouping ECBS

The WinC-MAPF framework requires an *optimal* Action Generator, which severely limits the range of solvers that can be used. This section first describes why naively trying to use a bounded suboptimal solver like ECBS will not satisfy the proof for completeness due to individual group's heuristics violating w-suboptimality (for the duration of this paper, w denotes a suboptimality weight while W denotes the planning window size). Then we describe our solution, Dynamic Agent Grouping ECBS (DAG-ECBS), which maintains that each group's heuristic is w-bounded, and show how it is complete within the WinC-MAPF framework.

Issue with Naïvely Using Bounded-Suboptimal Search: Conceptually, the WinC-MAPF proof in Section 2.4 relies on the fact that the heuristic $h(\mathcal{C})$ remains bounded throughout the planning and update procedure (i.e., Eq. 2). This is done by showing that $h(\mathcal{C}_{Gr}) \leq h^*(\mathcal{C}_{Gr})$. If we want to use a bounded suboptimal search, we can extend/re-use this proof if we can show that $h(\mathcal{C}) \leq w \cdot h^*(\mathcal{C})$ (i.e., is w-admissible), which we can do if we show that each group's heuristic w-admissible, i.e. $h(\mathcal{C}_{Gr}) < w \cdot h^*(\mathcal{C}_{Gr})$. Unfortunately, we show here that using a bounded-suboptimal solver such as ECBS does *not* maintain this.

Applying a bounded-suboptimal solver such as ECBS with a single factor w over all agents to the WinC-MAPF framework ensures only that the total cost is within w of optimal; individual groups can still be worse than w times their own optimum. For example, if two groups have optimal costs 10 and 40, a global $w = 2$ bound accepts any solution costing up to 100. A solution costing 30 and 45 for the two groups (total 75) is globally acceptable, yet the first group violates its local w-bound of 20, breaking the completeness proof when heuristics are updated. This unfortunately means that we cannot use the existing proof of completeness from WinC-MAPF. Further attempts on our end for alternative proofs of completeness were unfruitful.

Algorithm 1: Dynamic Agent Grouping ECBS

Parameters: Current configuration $\mathcal{C}^0$, heuristic penalties HPs

Output: w-suboptimal windowed solution $\mathcal{C}^{1:W}$, disjoint agent groups disjointGroups

1: $\mathcal{C}^{1:W} \leftarrow \emptyset$ ▷ *Initialize empty windowed path*
2: activeGroups $\leftarrow \{\{a\} \mid a \in \mathcal{A}\}$ ▷ *Singleton groups*
3: disjointGroups $\leftarrow \{\}$
4: **while** activeGroups $\neq \emptyset$ **do**
5: $Gr \leftarrow$ activeGroups.pop() ▷ *Current group to plan*
6: $\mathcal{C}^{1:W}_{Gr}, Grs_{int} \leftarrow$ WinCG-ECBS($\mathcal{C}^0_{Gr}$, $\mathcal{C}^{1:W}_{\text{disjointGroups}}$, HPs)
7: **if** $Grs_{int} \neq \emptyset$ **then** ▷ *Coupled with other agents*
8: merge Gr with groups $Gr' \in Grs_{int}$ and reinsert into activeGroups. Remove Gr' from disjointGroups.
9: **else if** $\mathcal{C}^{1:W}_{Gr} = \emptyset$ **then** ▷ *Failed to find a solution*
10: **return** FAILURE
11: **else** ▷ *Not coupled with other agents*
12: disjointGroups.insert(Gr) ▷ *Tentatively a disjointGroup*
13: **return** $\mathcal{C}^{1:W}$, disjointGroups

3.1 Dynamic Agent Grouping ECBS

Dynamic Agent Grouping ECBS (DAG-ECBS) is a bounded suboptimal search that computes disjoint agent groups and returns a w-suboptimal solution per agent group. DAG-ECBS does this by planning agent groups *independently* instead of planning across all groups at once. The idea of planning groups of agents is shared with Independence Detection (Standley 2010), and in the context of CBS is similar to MA-CBS (Sharon et al. 2012) and NECBS (Chan et al. 2022) which plan groups of agents. The purpose of grouping agents in MA-CBS and NECBS is to speed up the search. However, in our case we group agents as we need to compute a w-suboptimal solution for each disjoint agent group.

Conceptually, DAG-ECBS essentially runs ECBS on individual agent groups (starting with each agent in its own group) and iteratively merges and replans colliding groups of agents. Algorithm 1 describes DAG-ECBS's main procedure of planning groups of agents in more detail. We maintain a queue of active groups "activeGroups" which is initialized with singleton groups (i.e., we assign each agent its own individual group initially). We then proceed to plan each group Gr in activeGroups (lines 4-6) by calling the WinCG-ECBS function. WinCG-ECBS finds a bounded suboptimal windowed solution for the specific agents in Gr or returns a group of other agents that Gr is coupled with.

This process means that DAG-ECBS starts with singleton groups but iteratively merges and replans groups that interact with each other. The final result is a set of disjoint agent groups and their collision-free paths, with the guarantee that each group's solution is w-suboptimal.

3.2 Windowed Complete Group ECBS

Windowed Complete Group ECBS (WinCG-ECBS) is similar to calling regular windowed ECBS on a group of agents

Gr, except for the following changes. First, and identical to Single-Step CBS in WinC-MAPF, WinCG-ECBS incorporates heuristic penalties and constraints to return bounded suboptimal solutions that incorporate changes in heuristics. Second, we modify the high-level and low-level focal search as we maintain that $h(\mathcal{C})$ is w-admissible. This modification is required for the proof of completeness. Third, WinCG-ECBS detects if agents in Gr are coupled with other agents. If potential coupled agents are detected, WinCG-ECBS early terminates (before finding a solution) and returns the coupled agent groups Grs_{int} which will be merged together with Gr afterwards (lines 7-8).

Conceptually (but notationally imprecise), focal search (Pearl and Kim 1982) maintains w-suboptimality by maintaining a lower bound $c + h$ (given an admissible h) and multiplying the quantity by w for the focal threshold. We however maintain that our h is w-admissible, so we instead keep track of and base our focal threshold on $w \cdot c + h$ to maintain that our solution is w-suboptimal.

We now proceed to define the focal search mathematically. For notational convenience, n denotes a Constraint Tree (CT) node where conditioning on n (which is denoted as $|n$) represents that n's constraints must be satisfied. Additionally, since we want $h(\mathcal{C})$ to be w-admissible, we initialize $h(\mathcal{C})$ to $w \cdot h^{BD}(\mathcal{C})$. We thus redefine the heuristic penalty $h_p(\mathcal{C})$ to be the added residual, i.e., $h(\mathcal{C}) = h_p(\mathcal{C}) + w \cdot h^{BD}(\mathcal{C})$. We re-emphasize that conceptually our focal threshold is based $w \cdot c + h$, with our h initialized by a w-admissible value.

Low-Level Search: The original ECBS uses a focal search consisting of an "Anchor" queue and a "Focal" queue in the low-level search for planning single agent paths. The priority function used in Anchor is the standard A* priority (Eq. 3). Eq. 4 describes which states v are allowed in the Focal queue (e.g., which states agent i can go to while maintaining the solution path is w-suboptimal).

$$\text{Anchor}_i\text{: } F_1(v|n) := c(\mathcal{C}_i^0, v|n) + h_i^{BD}(v) \tag{3}$$

$$\text{Focal}_i\text{: } \{v|c(\mathcal{C}_i^0, v|n) + h_i^{BD}(v) \leq w \cdot \min_{v \in \text{Anchor}_i} F_1(v|n)\} \tag{4}$$

The low-level search in WinCG-ECBS in DAG-ECBS requires a subtle change to the Focal criteria. In particular, Eq. 5's sole change is weighting $h_i^{BD}(v)$ by w.

$$\text{Focal}_i\text{: } \{v|c(\mathcal{C}_i^0, v|n) + w \cdot h_i^{BD}(v) \leq w \cdot \min_{v \in \text{Anchor}_i} F_1(v|n)\} \tag{5}$$

The low-level search of WinCG-ECBS only avoids collisions with agents in the group Gr and ignores other agents.

High-Level Search: The original ECBS also uses a focal search for the high-level constraint search, with a similar Anchor queue and Focal queue. The priority function $F_2(n)$ used in Anchor is described in Eq. 6 where $\min_{v \in \text{Anchor}_i} F_1(v|n)$ can be interpreted as the lower bound of a path for agent i that satisfies CT node n's constraints. Eq. 7 describes which nodes are allowed in the Focal queue where $\mathcal{C}^W$ is the end of the partial solution for node n'.

$$\text{Anchor: } F_2(n) := \sum_i \min_{v \in \text{Anchor}_i} F_1(v|n) \tag{6}$$

$$\text{Focal: } \{n'|c(\mathcal{C}^0, \mathcal{C}^W) + h^{BD}(\mathcal{C}^W) \leq w \cdot \min_{n \in \text{Anchor}} F_2(n)\} \tag{7}$$

Conceptually, WinCG-ECBS in DAG-ECBS requires also keeping track of heuristic constraints/penalties that increase the heuristic of the group. Eq. 8 includes $h_p(n)$ which shows how we incorporate heuristic penalties according to n's constraints. Eq. 9 shows how this value is used to determine the modified Focal threshold.

$$\text{Anchor: } F_3(n) := h_p(n) + w \cdot \sum_i \min_{v \in \text{Anchor}_i} F_1(v|n) \tag{8}$$

$$\text{Focal: } \{n'|c(\mathcal{C}^0, \mathcal{C}^W) + w \cdot h^{BD}(\mathcal{C}^W) + h_p(n) \leq \min_{n \in \text{Anchor}} F_3(n)\} \tag{9}$$

Group Detection: WinCG-ECBS detects when the group Gr it is planning for might be coupled with other agents not in Gr. In WinCG-ECBS's high-level search, every time a high-level CT node is expanded, WinCG-ECBS first checks vertex, edge, and heuristic conflicts with other agents not in Gr. If conflicts exist, WinCG-ECBS early returns the conflicting agent group's without completing the search.

We highlight that an initial implementation of DAG-ECBS separated the path generation and group coupling detection. Specifically, WinCG-ECBS originally only returned a solution for Gr, which was then afterwards checked for conflicts with other agents. However this version was noticeably slower as WinCG-ECBS spent non-trivial time finding collision free paths that were later re-grouped and replanned. Thus this version of early terminating if agents are potentially grouped in intermediate WinCB-ECBS high-level nodes (even if that node may not have led to the solution) was more effective.

3.3 DAG-ECBS within WinC-MAPF

We directly use DAG-ECBS as an Action Generator within the WinC-MAPF framework of iterative planning, execution, and heuristic updates, and prove in the next section how we get completeness guarantees. Specifically, given a configuration $\mathcal{C}^0$, we use DAG-ECBS to return the next windowed solution $\mathcal{C}^{1:W}$ and disjoint agent groups. We then move the agents one step, update the heuristic of $\mathcal{C}_{Gr}^0$ for each disjoint agent group Gr via Eq. 1 (i.e. $h(\mathcal{C}_{Gr}^0) \leftarrow U(\mathcal{C}_{Gr}^0, \mathcal{C}_{Gr}^W)$), and repeat this process.

We emphasize that DAG-ECBS dynamically groups agents at every planning and execution iteration. This means that agents can be not grouped together for the first few time steps if they are initially far apart, then grouped when they move near each other, and then afterwards be not grouped together if they move apart.

One subtlety for readers familiar with single-agent RTHS is that we use suboptimality different than single-agent RTHS. Most single-agent RTHS methods use suboptimality to increase the rate of learning / heuristic updates. On the other hand, we use suboptimality to speed up the windowed planner, not to increase the heuristic update.

3.4 Sketch of Proof of Completeness

The core of DAG-ECBS's modifications is to show that each group's heuristic remains w-admissible after the heuristic update equation is applied. As mentioned earlier, the intuition is that if h is w-admissible, then $w \cdot c + h$ gives a w-admissible bound. The formal proof is given in the appendix.

With w-admissible heuristics and the finite state space, the standard RTHS / WinC-MAPF argument applies: the search cannot loop indefinitely (as this contradicts our proof of the heuristics always being w-admissible), so DAG-ECBS will not get stuck in a cycle and will eventually find a solution if one exists. We note that this proof enables designing other bounded-suboptimal windowed MAPF solvers / action generators and is more general than just DAG-ECBS.

One last subtlety is that since DAG-ECBS returns a windowed path, we also update the heuristic values of the intermediate configurations $\mathcal{C}_{G_r}^{1:W-1}$ based on $\mathcal{C}_{G_r}^{W}$ via a slightly modified Eq. 1, which we detail in the appendix still retains the updated heuristic value is w-admissible.

4 Experimental Results

We evaluate DAG-ECBS and windowed ECBS on standard benchmark maps (Stern et al. 2019) with windows $W = \{1, 2, 4, 8, 16, 32\}$ and suboptimalities $w = \{1, 1.5, 2, 3, 5\}$. We also include Single-Step CBS (SS-CBS) from the WinC-MAPF paper which is an optimal one-step solver with completeness guarantees. These methods interleave planning a windowed path, executing the first step, and repeating, with a 1-minute timeout summed across all planning iterations. Finally, we include full-horizon ECBS which just plans once in the very beginning. Fig 2 shows results across different maps, parameters, and metrics.

We note we did not compare against state-of-the-art full-horizon methods (e.g., EECBS (Li, Ruml, and Koenig 2021) or LaCAM (Okumura 2023)) for two main reasons. First, the contribution of this paper is advancing windowed planning with completeness guarantees. Thus, the main comparison is with prior windowed planning with completeness (SS-CBS) or to test against the naive version without guarantees (windowed ECBS) to evaluate the empirical impact of theoretical guarantees. Second, most MAPF research has focused on full-horizon methods, and thus current state-of-the-art full-horizon methods have better success rates than Fig 2.

Scalability: Figure 2a depicts the scalability of DAG-ECBS, ECBS, and SS-CBS as we vary the window size (x-axis) and suboptimality w (lines). Note that SS-CBS is an optimal solver with window size one and thus only appears as one point (red). First, comparing DAG-ECBS (solid lines) to SS-CBS, we see that DAG-ECBS outperforms SS-CBS, highlighting the benefit of incorporating suboptimality into the search. Interestingly, DAG-ECBS with $w = 1$ and $W = 1$ outperforms SS-CBS (where they technically should find equivalent solutions). We surmise this is because planning agent groups helps performance (similar to NECBS (Chan et al. 2022)). Second, observing ECBS (faded lines), we notice how performance is most tightly coupled with the window size; performance is poor for small window sizes and increases as the window size increases. The poor performance at small window sizes across all maps reveals how ECBS get stuck in deadlock/livelock in those situations. DAG-ECBS's superior performance show how incorporating heuristic updates reduces these issues.

Taking a step back, we see that DAG-ECBS consistently outperforms ECBS across all window sizes in maps random-32-32-20 and warehouse-10-20-10-2-1, but not for larger window sizes in ht_chantry and den520d. This is because the random and warehouse map have more congestion with a small number of agents that DAG-ECBS is able to overcome (and that ECBS gets stuck in). The other two maps are larger where congestion maily occurs with larger number of agents where DAG-ECBS starts to timeout while ECBS is able to make progress.

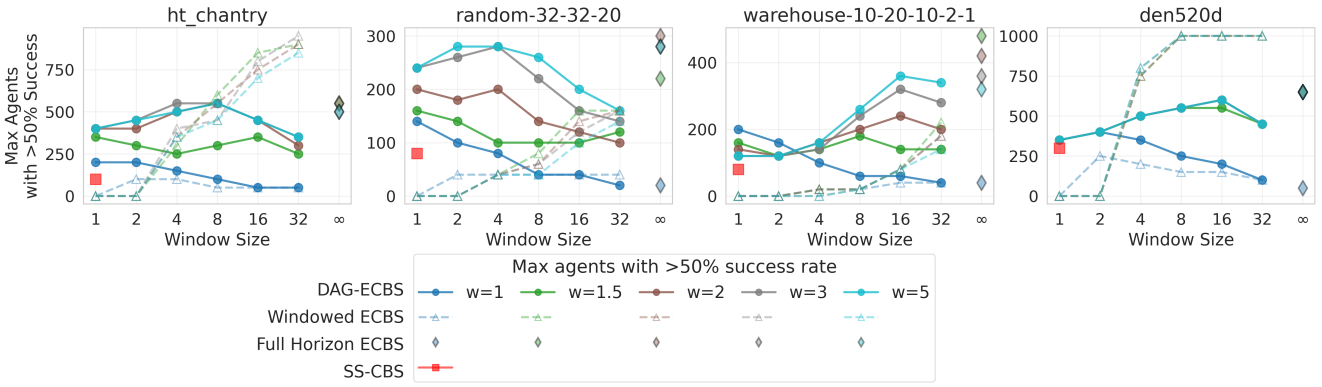
Varying Window Size Analysis: Figure 2b plots the maximum iteration runtime in seconds across each instance (failures included) and the average cost per agent (failures omitted). The maximum iteration runtime represents the longest planning time required across all planning and execution iterations within a single instance. We plot the maximum runtime as the per-iteration runtime distribution was heavily skewed and the few planning iterations that took the longest dominated the total planning time. Figure 3 includes the 25th-75th percentile information demonstrating this.

Each line denotes a different window size for DAG-ECBS and ECBS with suboptimality $w = 2$ (SS-CBS is omitted as it only work with $w = 1$). Comparing different colored lines, we see that increasing the window size increases the per-iteration runtime for both ECBS and DAG-ECBS, although it seems to affect the runtime of DAG-ECBS more. Combined with Fig 2a, we see that DAG-ECBS fails due to runtime issues.

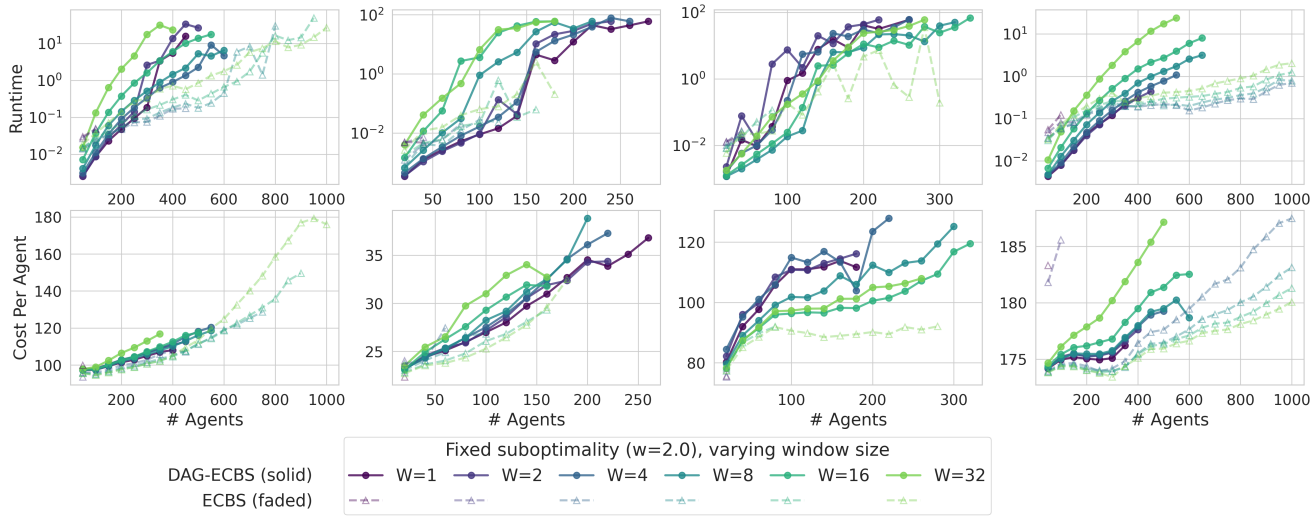
The effect of window size on cost varies across maps. In warehouse, larger window sizes reduce costs, while in the other three maps, they increase costs. We conceptualize two different explanations that appear to play out in different situations. First, a larger window size enables avoiding myopic decisions and therefore decreases overall solution length. Second, since we are employing suboptimality ($w = 2$), a larger window size means that at every planning iteration, we can compute a more suboptimal path and increases overall solution cost.

Varying Suboptimality Analysis: Fig 2c keeps the window sized fixed ($W = 4$) and varies the suboptimality (lines) of DAG-ECBS and ECBS. First, observing the runtimes of DAG-ECBS, we see that increasing the suboptimality consistently reduces DAG-ECBS's per-iteration runtime. This boost in performance, especially in comparison to an optimal solver ($w = 1$), highlights the importance of being able to use a bounded suboptimal solver for larger window sizes.

ECBS seems to have a slightly runtime story. Like DAG-ECBS, ECBS with $w = 1$ (blue triangle line) times out fast. However, unlike DAG-ECBS, ECBS with $w > 1$ all have similar runtimes (as opposed to DAG-ECBS where different $w > 1$ have differing runtimes). We hypothesize this is due to the low-level focal queue. In particular, ECBS and DAG-ECBS have the same focal threshold value, but

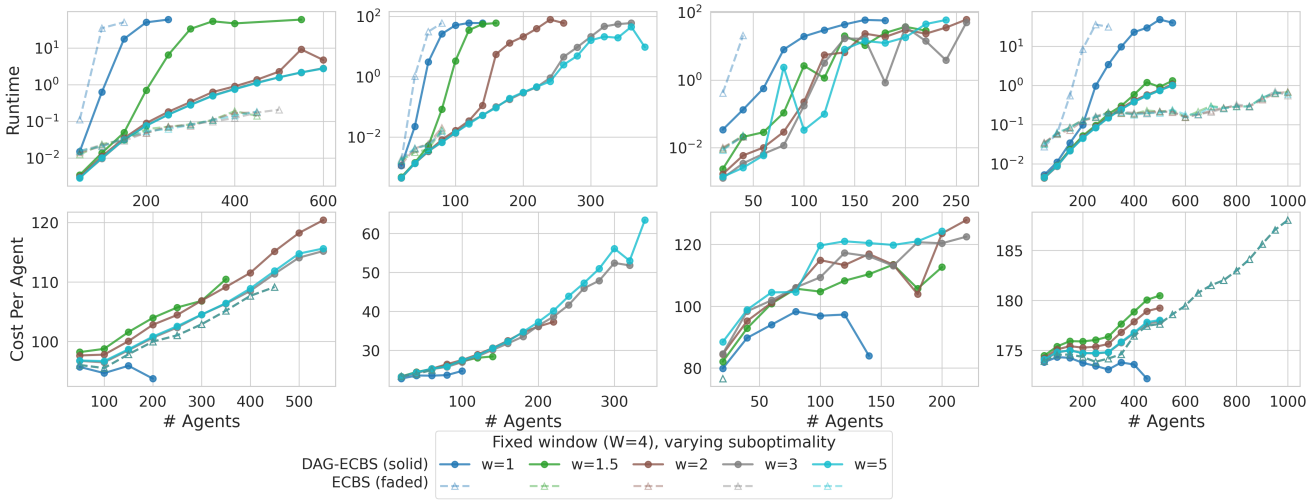


(a) The maximum number of agents DAG-ECBS, windowed ECBS, and SS-CBS can scale to with a $> 50\%$ success rate.



(b) The effect of varying the window size W in DAG-ECBS and windowed ECBS for suboptimality $w = 2$.

hi



(c) The effect of varying suboptimality w in DAG-ECBS and windowed ECBS for fixed window $W = 4$.

Figure 2: Evaluating DAG-ECBS and windowed ECBS across various suboptimalities (w), window sizes (W), and maps.

DAG-ECBS includes a $w \cdot h^{BD}(\mathcal{C}_i)$ term on the left hand side of the inequality (Eq. 5) while ECBS does not (Eq. 4). Since windowed ECBS is planning a partial path, this means many nodes can satisfy the focal threshold. On the flip end, there are less nodes in DAG-ECBS's focal queue. Analyzing the solution cost seems to verify this hypothesis. For DAG-ECBS, we see that increasing the suboptimality consistently increases the solution cost on all 4 maps. However for ECBS with $w > 1$, all their solution costs seem to be similar.

5 Conclusion

The Windowed Complete MAPF (WinC-MAPF) framework shows how to use windowed planning while retaining completeness guarantees. However, WinC-MAPF required an optimal windowed solver, restricting the types of useable solvers and limiting the scalability of the framework.

Our main contribution is extending the framework by showing how we can use a bounded suboptimal windowed solver, Dynamic Agent Grouping ECBS (DAG-ECBS), that maintains solution guarantees in the WinC-MAPF framework. The key idea in DAG-ECBS is to maintain that each groups heuristic is w-admissible by planning agent groups independently, with agent groups getting merged and replanned if they conflict.

We highlight that the idea of grouping agents via conflicts and planning agent groups is broadly applicable to existing suboptimal MAPF algorithms. In fact, Alg 1 can be viewed as a template by replacing WinCG-ECBS with other algorithms in Line 6. Thus, DAG-ECBS serves as a template for converting suboptimal MAPF algorithms into Action Generators in the WinC-MAPF framework. We empirically show that DAG-ECBS improves the scalability of WinC-MAPF compared to the single-step optimal equivalent. We additionally show that DAG-ECBS consistently outperforms windowed ECBS (without completeness guarantees) for small window sizes or in congested maps.

We see two main avenues for future work. First, we imagine adapting other more advanced bounded suboptimal MAPF algorithms like EECBS (Li, Ruml, and Koenig 2021) to follow our technique / template (e.g., creating a DAG-EECBS). Second, our dynamic grouping is reminiscent of decentralized multi-agent algorithms where agents near each other communicate (e.g., are grouped) while agents far apart do not. Thus, even though we implement DAG-ECBS as a centralized solver, we imagine that a decentralized version of DAG-ECBS is feasible.

A Proof of Completeness

We formally prove how the DAG-ECBS maintains completeness in the WinC-MAPF framework. The core component is proving that the heuristic values of each group is w-admissible even with heuristic updates.

Theorem 1. *For every disjoint agent group, the heuristic maintained by* DAG-ECBS *is always* w-admissible*, i.e.,* $h(\mathcal{C}_{Gr}) \leq w \cdot h^*(\mathcal{C}_{Gr})$.

Proof. **Base case.** All possible group heuristics $h(\mathcal{C}_{Gr})$ are w-admissible as they are initially set to $w \cdot h^{BD}(\mathcal{C}_{Gr}) = w \cdot \sum_{i \in Gr} h_i^{BD}(\mathcal{C}_i)$ (Section 3.2). Note that $h_i^{BD}(\mathcal{C}_i)$ is admissible as it is the individual agent's cost to reach the goal ignoring other agents.

Inductive step. We assume $h(\mathcal{C}_{Gr}) \leq w \cdot h^*(\mathcal{C}_{Gr})$ for all possible group configurations before planning. Then, DAG-ECBS is called and returns a set of disjoint agent groups and windowed paths. From the inductive hypothesis, we assume that existing h is admissible for all group configurations. Each agent group's windowed path ending in $\mathcal{C}_{Gr}^W$ is the result of calling WinCG-ECBS where the solution node n^{sol} contained the windowed solution. We show then that the following holds:

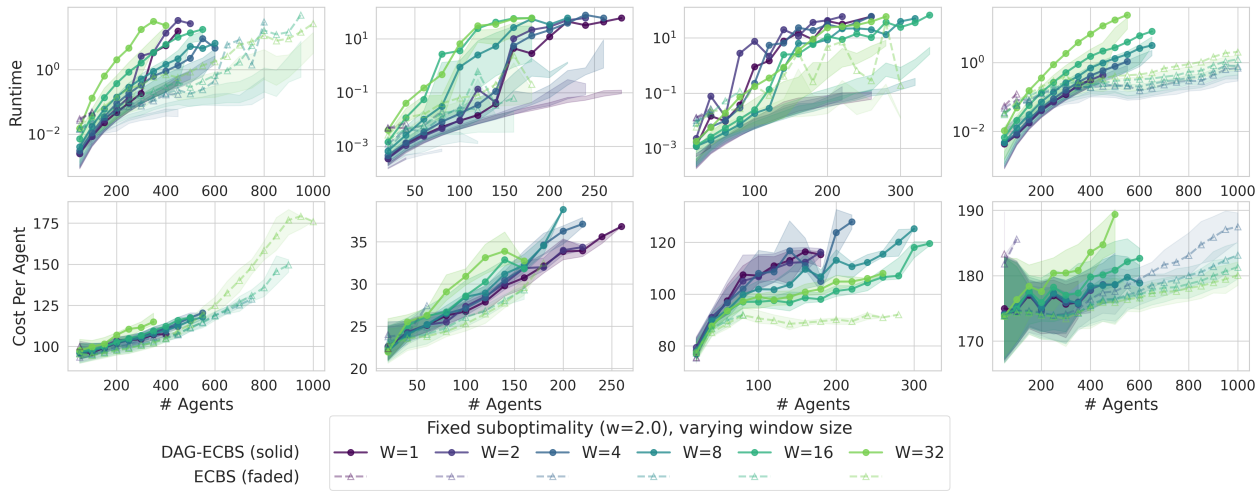
$$
\begin{aligned}
&c(\mathcal{C}_{Gr}^0, \mathcal{C}_{Gr}^W) + h(\mathcal{C}_{Gr}^W) = \\
&c(\mathcal{C}_{Gr}^0, \mathcal{C}_{Gr}^W) + w \cdot h^{BD}(\mathcal{C}_{Gr}^W) + h_p(n^{sol}) \leq \\
&\min_{n \in \text{Anchor}} F_3(n) = \min_{n \in \text{Anchor}} h_p(n) + w \cdot \sum_{i \in Gr} \min_{v \in \text{Anchor}_i} F_1(v|n) \leq \\
&\min_{n \in \text{Anchor}} h_p(n) + w \cdot \sum_{i \in Gr} F_1(\mathcal{C}_i^W|n) = \\
&\min_{n \in \text{Anchor}} h_p(\mathcal{C}_{Gr}^W) + \sum_{i \in Gr} w \cdot c^*(\mathcal{C}_i^0, \mathcal{C}_i^W|n) + w \cdot h_i^{BD}(\mathcal{C}_i^W) = \\
&\min_{n \in \text{Anchor}} w \cdot c^*(\mathcal{C}_{Gr}^0, \mathcal{C}_{Gr}^W|n) + h^*(\mathcal{C}_{Gr}^W) \leq w \cdot h^*(\mathcal{C}_{Gr}^0)
\end{aligned}
\tag{10}
$$

where $\mathcal{C}_i^W \leftarrow \arg\min_{v \in \text{Anchor}_i} F_1(v|n)$ since each agent is constrained to be at $\mathcal{C}_i^W$ due to the heuristic penalty constraint.

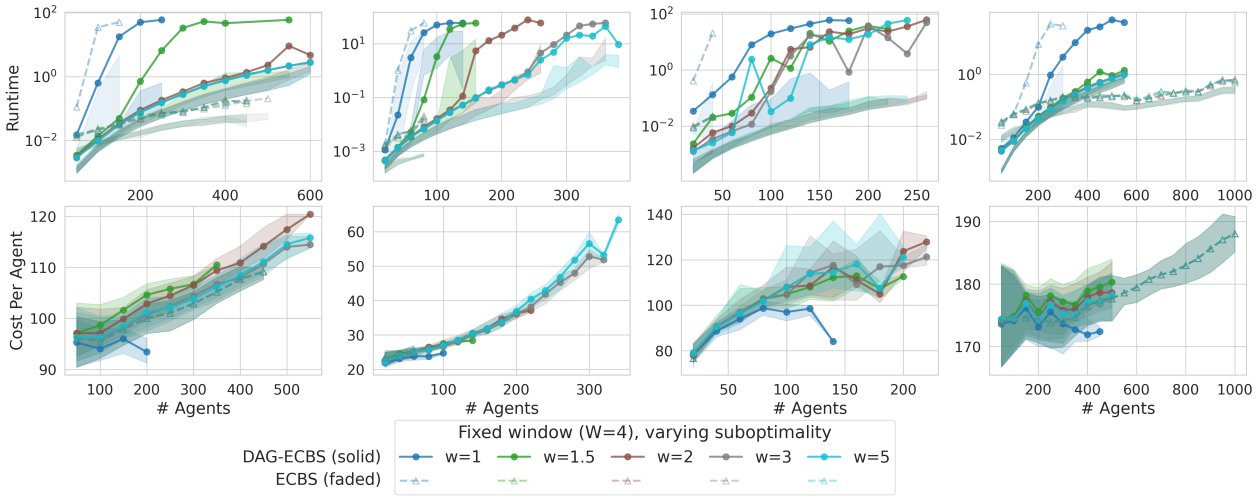
The first to second row uses the definition of heuristic penalties (Section 3.2). The second to third row uses the focal threshold in WinCG-ECBS (Eq. 9). The third row substitutes using Eq. 8. The third to fourth row substitutes the minimum with $\mathcal{C}_i^{*W}$ where $\mathcal{C}_i^{*W}$ denotes the optimal path to $\mathcal{C}_i^W$. This holds as the optimal path or a prefix of it will exist as a vertex in Anchor_i as Anchor_i searches over all paths (so the minimum of Anchor_i is smaller than it). The fourth to fifth row substitutes using Eq. 3 and can use the optimal costs $c(\mathcal{C}_i^0, \mathcal{C}_i^W|n)$ using the definition of $\mathcal{C}_i^{*W}$ defined earlier. The fifth to sixth row equality uses the definition of heuristic penalty (Section 3.2) and the inductive hypothesis, and the last inequality holds as the Anchor in WinCG-ECBS admits the optimal solution. □

Given this fact, we can then reuse the standard Real-Time Heuristic Search / WinC-MAPF argument as described in Section 3.4. To our knowledge, this is the first time a focal search has been used in RTHS, so this proof/methodology may be of relevance to single-agent RTHS researchers.

One last subtlety is that since DAG-ECBS returns a windowed path, we also update the heuristic values of the intermediate configurations $\mathcal{C}_{Gr}^t$ for $t \in [1, W-1]$ based on the path to $\mathcal{C}_{Gr}^W$. Specifically, we update using $h(\mathcal{C}_{Gr}^t) \leftarrow \max(h(\mathcal{C}_{Gr}^t), U(\mathcal{C}_{Gr}^0, \mathcal{C}_{Gr}^W) - w \cdot c(\mathcal{C}_{Gr}^0, C_{Gr}^t))$ which we can prove is w-admissible if $U(\mathcal{C}_{Gr}^0, \mathcal{C}_{Gr}^W)$ is w-admissible (and hence retains overall completeness guarantees) by rearranging terms. In the following proof we drop the group subscript



(a) The effect of varying the window size W in DAG-ECBS and windowed ECBS for suboptimality $w = 2$.



(b) Evaluating DAG-ECBS and windowed ECBS across various suboptimalities (w), window sizes (W), and maps.

Figure 3: Same as Figure 2 except with additional shaded region denoting 25-75th percentiles. The solid line in Runtime denotes maximum runtime across all iterations and scenes, while the solid line in cost per agent denotes mean cost across scenes. Note that the maximum per-iteration runtimes are significantly larger than most of the per-iteration runtime distribution.

for notational simplicity:

$$
\begin{aligned}
&U(\mathcal{C}^0, \mathcal{C}^W) \leq w \cdot h^*(\mathcal{C}^0) \\
&U(\mathcal{C}^0, \mathcal{C}^W) \leq w \cdot [c^*(\mathcal{C}^0, \mathcal{C}^t) + h^*(\mathcal{C}^t)] \\
&U(\mathcal{C}^0, \mathcal{C}^W) - w \cdot c^*(\mathcal{C}^0, \mathcal{C}^t) \leq w \cdot h^*(\mathcal{C}^t) \\
&U(\mathcal{C}^0, \mathcal{C}^W) - w \cdot c(\mathcal{C}^0, \mathcal{C}^t) \leq w \cdot h^*(\mathcal{C}^t) \qquad (11)
\end{aligned}
$$

The first line is the result of Eq. 10. The transition to the second line is via h^* being consistent (an optimal heuristic is always consistent). The transition to the third line re-arranges terms. The final line uses the fact that $c^*(...) \leq c(...)$ by definition, so subtracting by a larger value will still retain the inequality.

B Additional Plots

Figure 3 shows the same data as Figure 2 along with 25th and 75th percentile information.

Acknowledgments

This work is in part supported by the National Science Foundation (NSF) under grant numbers 2328671 and 2441629, as well as a gift from Amazon.

References

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 *Seventh Annual Symposium on Combinatorial Search*.

Chan, S.-H.; Li, J.; Harabor, D.; Stuckey, P. J.; Gange, G.; Cohen, L.; and Koenig, S. 2022. Nested ECBS for Bounded-Suboptimal Multi-Agent Path Finding. *Proceedings of the International Conference on Automated Planning and Scheduling*, 79–87.

Erdmann, M.; and Lozano-Perez, T. 1987. On multiple moving objects. *Algorithmica*, 2(1): 477–521.

Goldenberg, M.; Felner, A.; Stern, R.; Sharon, G.; Sturtevant, N.; Holte, R. C.; and Schaeffer, J. 2014. Enhanced partial expansion A. *Journal of Artificial Intelligence Research*, 141–187.

Jiang, H.; Zhang, Y.; Veerapaneni, R.; and Li, J. 2024. Scaling Lifelong Multi-Agent Path Finding to More Realistic Settings: Research Challenges and Opportunities. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, 234–242.

Koenig, S.; and Likhachev, M. 2006. Real-time adaptive A. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 281–288.

Korf, R. E. 1990. Real-time heuristic search. *Artificial Intelligence*, 42(2): 189–211.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021a. Anytime multi-agent path finding via large neighborhood search. In *International Joint Conference on Artificial Intelligence 2021*, 4127–4135. Association for the Advancement of Artificial Intelligence (AAAI).

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9): 10256–10265.

Li, J.; Ruml, W.; and Koenig, S. 2021. EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14): 12353–12362.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. S.; and Koenig, S. 2021b. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11272–11281.

Okumura, K. 2023. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10): 11655–11662.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 103752.

Pearl, J.; and Kim, J. H. 1982. Studies in Semi-Admissible Heuristics. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 392–399.

Rivera, N.; Baier, J. A.; and Hernández, C. 2013. Weighted real-time heuristic search. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 579–586.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2012. Meta-Agent Conflict-Based Search For Optimal Multi-Agent Path Finding. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search (SoCS)*, 97–104.

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

Sigurdson, D.; Bulitko, V.; Yeoh, W.; Hernández, C.; and Koenig, S. 2018. Multi-Agent Pathfinding with Real-Time Heuristic Search. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8.

Silver, D. 2005. Cooperative pathfinding. In *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, 117–122.

Standley, T. 2010. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI conference on artificial intelligence*, 173–178.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Bartak, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Symposium on Combinatorial Search (SoCS)*, 151–158.

Veerapaneni, R.; Kusnur, T.; and Likhachev, M. 2023. Effective integration of weighted cost-to-go and conflict heuristic within suboptimal CBS. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11691–11698.

Veerapaneni, R.; Saleem, M. S.; Li, J.; and Likhachev, M. 2025. Windowed MAPF with Completeness Guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 23323–23332.

Zhang, Y.; Chen, Z.; Harabor, D.; Bodic, P. L.; and Stuckey, P. J. 2024. Planning and Execution in Multi-Agent Path Finding: Models and Algorithms. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34(1): 707–715.