Local Guidance for Configuration-Based Multi-Agent Pathfinding

Tomoki Arita[1,2], **Keisuke Okumura**[1,3]

[1]National Institute of Advanced Industrial Science and Technology (AIST), Japan
[2]Keio University, Japan
[3]University of Cambridge, UK
{arita.tomoki, okumura.k}@aist.go.jp

Abstract

Guidance is an emerging concept that improves the empirical performance of real-time, sub-optimal multi-agent pathfinding (MAPF) methods. It offers additional information to MAPF algorithms to mitigate congestion on a *global* scale by considering the collective behavior of all agents across the entire workspace. This global perspective helps reduce agents' waiting times, thereby improving overall coordination efficiency. In contrast, this study explores an alternative approach: providing *local* guidance in the vicinity of each agent. While such localized methods involve recomputation as agents move and may appear computationally demanding, we empirically demonstrate that supplying informative spatiotemporal cues to the planner can significantly improve solution quality without exceeding a moderate time budget. When applied to LaCAM, a leading configuration-based solver, this form of guidance establishes a new performance frontier for MAPF.

Introduction

A holy grail challenge in multi-agent pathfinding (MAPF) research is to develop scalable yet real-time methods for computing collision-free paths with reasonable solution quality. Given the computational intractability of finding optimal solutions (Yu and LaValle 2013; Banfi, Basilico, and Amigoni 2017), the field has increasingly shifted its focus toward efficient suboptimal approaches. Indeed, the past decade has seen the emergence of several powerful algorithms capable of handling hundreds of agents. Notable examples include *PIBT* (Okumura et al. 2022) and *LaCAM* (Okumura 2023b,a), which rapidly and reliably deliver feasible solutions, even in densely populated scenarios where other MAPF algorithms falter.

The evolution of these suboptimal algorithms reveals a new MAPF challenge, particularly in dense setups: planners must now mitigate *congestion* in addition to deriving collision-free motions. As one might intuitively expect, when many agents gather in a specific spatiotemporal region, most agents are forced to wait in place, wasting time, and thus significantly degrading the planning performance.

This awareness leads to a series of studies introducing *guidance* (Zhang et al. 2024). The guidance itself does not derive collision-free paths; rather, it supports other heuristic search methods by providing congestion mitigation information, as like the left-hand side rule in traffic management. Such guidance indeed plays a critical role in achieving high-performance MAPF systems, as we see examples in winning strategies in the MAPF competition (Chan et al. 2024a; Jiang et al. 2024; Yukhnevich and Andreychuk 2025). The exact form of guidance varies from study to study, but it is typically constructed through a *global* consideration of the entire environment and the entire team of agents (Han and Yu 2022; Chen et al. 2024; Kato et al. 2025). Such global guidance also discards time-parametrized information about agent trajectories to ensure they remain trackable.

This study, in contrast, develops the notion of *local* guidance around individual agents. The underlying rationale is that global guidance, at one extreme, can be viewed as a coarse solution concept that relaxes the strict collision constraints of exact solutions, which lie at the other extreme. Local guidance is designed to lie between these two ends of the spectrum: it is more informative than global guidance, incorporating spatiotemporal awareness, yet less computationally demanding than computing fully collision-free, (near-)optimal paths. As a result, it offers a practical compromise that balances planning effort and solution quality.

We instantiate the concept of local guidance using a leading suboptimal MAPF algorithm, LaCAM, which relies on a configuration—the simultaneous positions of all agents—as its planning representation. Empirical results show that this enhancement significantly improves solution quality, outperforming its global counterpart while maintaining real-time responsiveness, typically within a few seconds even for $1,000$ agents. In the most extreme case, we observe a 50% reduction in solution cost compared to the original LaCAM. Moreover, the local guidance enhancement mostly surpasses the state-of-the-art anytime MAPF solver (Okumura 2024). Together, these results further push the frontier of real-time MAPF, promoting the practicality of MAPF in real-world applications. The code is publicly available at https://github.com/allegorywrite/lg_lacam.

Preliminaries and Related Work

Problem Definition. This paper focuses on one-shot, classical MAPF formulation (Stern et al. 2019). The system consists of a set of agents $A = \{1, 2, \dots, n\}$ and a graph $G = (V, E)$. All agents act synchronously according to a discrete wall-clock time. At each timestep, each agent can stay in place or move to an adjacent vertex. Vertex and edge

collisions are considered, i.e., two agents cannot occupy the same vertex simultaneously, and two agents cannot swap their occupied vertices within one timestep move. Then, we aim to find a finite action sequence for each agent $i \in A$ to its assigned goal $g_i \in V$ from a given start $s_i \in V$. The solution quality is assessed by *flowtime* (aka. sum-of-costs; SoC), which sums the travel time of each agent until it stops at the target location and no longer moves.

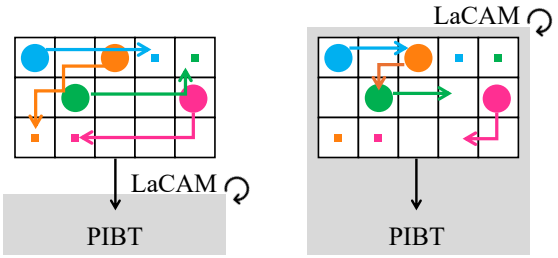
LaCAM *(lazy constraints addition search)* (Okumura 2023b) is a search-based algorithm that finds unbounded suboptimal solutions quickly, leading to large-scale MAPF studies due to its scalability and real-time responsiveness (Shen et al. 2023). Although its planning ability to handle densely populated, complicated instances at scale is remarkable (Okumura 2023a), the resultant solutions are known to be highly suboptimal (Chan et al. 2024b; Tan et al. 2025). This gives rise to two practical challenges: *(i)* finding better initial solutions with an acceptable computational overhead, or *(ii)* accelerating the refinement process to reach optimal solutions from feasible initial ones (Okumura 2024). This study focuses on the first challenge, but we will later demonstrate that improving the initial solutions leads to better final outcomes under the severe time limit.

How LaCAM Works. Let a *configuration* $\mathcal{Q} \in V^n$ be the locations for all agents. For instance, the start configuration is $\mathcal{S} = (s_1, s_2, \dots, s_n)$ and the goal is $\mathcal{G} = (g_1, g_2, \dots, g_n)$. Then, MAPF is reduced to single-agent pathfinding from $\mathcal{S}$ to $\mathcal{G}$ on a graph consisting of configurations. Two configurations are neighboring if any agent can move from one to the other within one timestep, and there are no collisions during this transition. This connectivity defines a graph structure; thus, any search algorithm, such as depth-first search (DFS), can derive a solution to MAPF. Within this scheme, LaCAM is a variant of DFS, but with a lazy successor generation that avoids inspecting all successor configurations at once, which is exponential to the number of agents. This generation process utilizes another MAPF algorithm, termed *configuration generator*, to guide the search towards the goal $\mathcal{G}$ efficiently.

Configuration Generator generates a connected configuration $\mathcal{Q}_{k+1}$ given a configuration $\mathcal{Q}_k$. LaCAM's performance in terms of both speed and quality is heavily influenced by this process as it determines the search direction. Typically, PIBT is employed to prioritize speed.

PIBT *(priority inheritance with backtracking)* (Okumura et al. 2022) is a computationally lightweight function that synthesizes a neighboring configuration given another $\mathcal{Q}$. As fast configuration generation is central to various MAPF situations, PIBT has gained notable popularity in recent years. For each agent i, it uses a sorted list of candidate actions v from $\mathcal{Q}[i]$, termed *preference*, where $v \in \mathsf{neigh}(\mathcal{Q}[i]) \cup \{\mathcal{Q}[i]\}$ and neigh denotes a set of adjacent vertices.

Preference Construction in PIBT originally employs sorting actions in a lexicographic and ascending order with $\langle \mathsf{dist}(v, g_i), \epsilon \rangle$, where dist denotes the shortest path distance on G and ϵ is a random number to break ties. Although this certainly guides agents towards their respective goals, it can also result in greedy, myopic behaviors, which are often accompanied by deadlock, livelock, or congestion. As PIBT gets popularity, numerous methods have been developed to optimize the preference: Monte-Carlo sampling (Okumura 2024), imitation learning (Jiang et al. 2025; Jain et al. 2025), DFS (Gandotra et al. 2025), and tiebreaking (Okumura and Nagai 2025), among others.



(a) global guidance (b) local guidance

Figure 1: Concept of global and local guidance with LaCAM. Goals for agents are marked with colored boxes. The gray-shaded regions correspond to configuration generation.

Guidance is one of these attempts in PIBT's preference optimization, which encourages agents to follow the congestion mitigation heuristic to optimize the traffic flow *globally*. Various approaches exist to construct guidance, such as handcrafted (Wang, Botea et al. 2008; Yukhnevich and Andreychuk 2025), heuristic search (Han and Yu 2022; Chen et al. 2024; Kato et al. 2025), or blackbox optimization (Zhang et al. 2024). To improve LaCAM, we are also interested in better preferences with guidance, but *locally*, i.e., mitigating congestion surrounding each agent in each configuration. Figure 1 sketches this difference between global and local ones and how to integrate them within the search. The remainder of this paper crystallizes this notion with empirical evidence.

Method

This study follows the guidance construction based on heuristic search (Han and Yu 2022; Chen et al. 2024; Okumura 2024), which is known to effectively improve representative MAPF algorithms, including LaCAM, without the need for prior preparation (e.g., offline data collection). Specifically, we utilize Alg. 1 to construct agent-wise paths Φ starting from a given configuration $\mathcal{Q}$ towards the goal configuration $\mathcal{G}$. Alg. 1 adopts *windowed* planning, where the planning horizon is limited by w timesteps ahead, which is known to be a technique for attaining computational efficiency (Silver 2005; Li et al. 2021b). In the pseudocode, $\{c, c_T\} : V \times V \mapsto \mathbb{R}$ represent the cost functions; c is for the stage cost while c_T is for the terminal cost.

Observe that the structure abstracted in Alg. 1 captures various MAPF techniques by adjusting the cost definition. For instance, seminal prioritized planning (PP) (Erdmann and Lozano-Perez 1987; Silver 2005) is considered to be a type of Alg. 1 with an infinite window size and hard constraints for goals and collisions that must never be violated.

Algorithm 1: Guidance Construction

input: configuration $\mathcal{Q} \in V^n$, goals $\mathcal{G} = (g_1, g_2, \ldots, g_n) \in V^n$
output: $\Phi \in V^{n \times (w+1)}$ **params:** $w \in \mathbb{N}_{>0}$

1: initialize Φ
2: **while** termination condition not met **do**
3: **for** $i \in A$ **do**
4: $\Phi[i] \leftarrow \underset{\pi \in \Pi}{\arg\min} \sum_{t=0}^{w-1} c(\pi[t], \pi[t+1]) + c_T(\pi[w], g_i)$
Π : paths from $Q[i]$ on G with length $w+1$
5: **return** Φ

These hard constraints are represented by infinite costs when they are not met. LNS2 (large neighborhood search) (Li et al. 2022) is another instantiation of Alg. 1, which starts from collision-containing infeasible paths and gradually resolves them by repairing paths for a subset of the entire team. LNS2 uses soft collision constraints, which allow for collisions during replanning but are negatively incentivised.

We leverage this widely-used algorithmic structure to construct the local guidance. *It should mitigate congestion locally while remaining computationally efficient*, given that it is invoked at each search iteration of LaCAM. For this purpose, we employ windowed planning, where w typically ranges from 5 to 20. This approach naturally focuses on each agent's local situation while avoiding excessive computational overhead. Congestion mitigation is achieved through carefully designed soft collision constraints. In what follows, we first describe how to exploit the guidance paths, and then specify each component of Alg. 1.

Exploitation. Once the guidance paths Φ have been obtained, a design choice must be made regarding how to incorporate them into PIBT. We follow the latest LaCAM implementation (Okumura 2024), which uses the global guidance. Specifically, PIBT uses a scoring

$$\langle \mathsf{Ind}\big[\Phi[i][1] \neq v\big], \mathsf{dist}(v, g_i), \epsilon \rangle \tag{1}$$

for the preference construction, where Ind is an indicator function that returns one only if the condition is true, otherwise zero. This encourages agents to follow the guidance paths while minimizing the computational overhead required to extract the guidance information. Note that this preserves LaCAM's completeness guarantee for MAPF.

Cost and Path Planning (Alg. 1). The cost design is core for achieving effective guidance. Inspired from the studies on path-based global guidance (Han and Yu 2022; Chen et al. 2024; Okumura 2024), we use a lexicographic cost:

$$c(\pi[t], \pi[t+1]) := \langle 1 + \alpha \cdot \mathsf{Ind}[\chi > 0], \chi \rangle, \tag{2}$$

$$c_T(\pi[w], g_i) := \langle \mathsf{dist}(\pi[w], g_i), 0 \rangle, \tag{3}$$

where χ denotes the number of collisions for $(\pi[t], \pi[t+1])$ with other paths in Φ. The hyperparameter $\alpha \in \mathbb{R}_{\geq 0}$ determines the importance of collisions; our implementation sets $\alpha = 3$ by default, based on the pilot study. For agent i at configuration $\mathcal{Q}$, this will find the first fragment of the path from $\mathcal{Q}[i]$ to g_i with the fewer collisions within the window, while avoiding overly conservative motion. Such a path can be quickly identified by space-time A* (Silver 2005). This is because the search space is limited by windowed planning, which enables the memory needed for A* to be pre-allocated.[1]

Plan Iteration (Alg. 1). As Alg. 1 plans paths sequentially, the first planned agent is less constrained by others in terms of collision penalty, while the last planned agent is heavily constrained. To resolve this unfairness, we iterate the path planning at Lines 3–4 several times. Empirically, we found that a few iterations, e.g., twice, are sufficient to obtain effective guidance. Meanwhile, repeated planning for the entire team adds non-negligible computational overhead. Therefore, as explained next, if available, we reuse the previous guidance paths to approximate the first iteration.

Initialization (Alg. 1). As LaCAM adopts DFS, the search within LaCAM explores configurations in a connected sequence. For example, if the current search iteration generates $\mathcal{Q}_{k+1}$ from $\mathcal{Q}_k$, the previous iteration should be for $\mathcal{Q}_{k-1}$, which is connected to $\mathcal{Q}_k$. Utilizing this structure, we initialize the guidance Φ_k for $\mathcal{Q}_k$ by the previous result Φ_{k-1} for $\mathcal{Q}_{k-1}$ with Alg. 2, at Alg. 1 in Alg. 1. This procedure shifts the guidance path for agent i by one timestep if i is at the intended location of the previous guidance. This initialization allows us to omit the plan iteration, which results in significant time savings.

Algorithm 2: Initialization

input: configuration $\mathcal{Q}_k$, previous guidance Φ_{k-1}
output: Φ_k (each path is initialized as empty)

1: **for** $i \in A$ **do**
2: **if** $\mathcal{Q}_k[i] = \Phi_{k-1}[i][1]$ **then**
3: **for** $t = 0, \ldots, w-1$ **do**
4: $\Phi_k[i][t] \leftarrow \Phi_{k-1}[i][t+1]$
5: **return** Φ_k

Agent Order (Alg. 1). Similar to PP, whose performance can vary depending on the order of the agents (Wu, Bhattacharya, and Prorok 2020), the agent order at Alg. 1 in Alg. 1 also affects the resultant guidance quality, especially when the number of plan iterations is limited. For the construction of guidance Φ_k, we sort the agent list A in descending order according to the number of collisions in the previous guidance Φ_{k-1}. The rationale is to relax the unfairness between agents, as discussed in the plan iteration.

Global Guidance Consideration. It is possible to incorporate global guidance into local guidance construction to mitigate congestion on a global scale. Assume that the global guidance Ψ is provided, and it takes the form of agent-wise paths similar to our local guidance, i.e., $\Psi[i] \in V^m$, as seen in (Han and Yu 2022; Chen et al. 2024; Okumura 2024). Then, we modify the cost function for agent i,

[1] An alternative to space-time A* is SIPP (Phillips and Likhachev 2011), a fast single-agent pathfinding method for dynamic environments. We tested SIPP, but did not achieve its speed benefits over space-time A*.

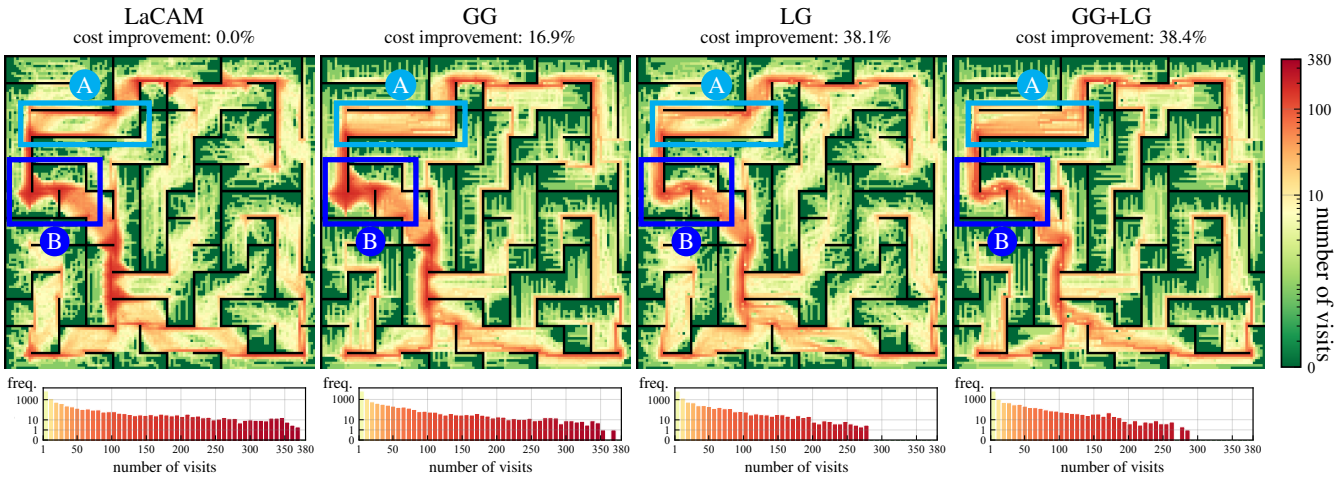


Figure 2: Vertex usage in LaCAM solutions, where congestion is implied by warm colors. The same start-goal instance comprising 1,000 agents on *maze-128-128-10* is used across different solvers. The cost improvement percentage is shown at the top, with LaCAM as the baseline. The bottom shows the distribution among 14,818 vertices of how many times each vertex is used by any agent. In the middle, area-A shows the effect of global guidance, while area-B for local guidance.

Eq. (2), to reflect the global guidance as

$$c(\pi[t], \pi[t+1]) := \langle 1 + \alpha \cdot \mathsf{Ind}[\chi > 0], \delta(\pi[t+1]), \chi \rangle. \tag{4}$$

The new interim term $\delta(v)$ measures the distance from $\Psi[i]$, which is computed by lazy breadth-first search upon query. The terminal cost, $c_T(\cdot)$, remains unchanged except for a dimension match. The updated cost design incentivizes agents to adhere to global guidance while prioritizing the mitigation of local congestion.

Integration with Swap. The *swap* technique is an important technique for stabilizing LaCAM's planning performance (Okumura 2023a). It reverses the PIBT preference for specific agents, enabling those who need to exchange locations within narrow corridors to escape local deadlocks. Local guidance can co-exist with the swap technique: if the swap situation is identified for an agent, we simply discard the guidance term in Eq. (1) and use the reverse scoring $\langle 0, -\mathsf{dist}(v, g_i), \epsilon \rangle$.

Time Complexity of Guidance Construction. Running w-windowed space-time A* on a graph $G = (V, E)$ has a time complexity of $O(w|E| + w|V|\log(w|V|))$, given that Dijkstra's algorithm on the time-expanded graph of G with horizon w exhibits the same order. The local guidance construction executes this process for n agents, m times, resulting in an overhead of $O(mn(w|E| + w|V|\log(w|V|)))$. As demonstrated in the experiments, $m = 1$ is typically sufficient for solution improvement, with the plan initialization (Alg. 2) operating in $O(nw)$. Assuming G is a four-connected grid, typically used in MAPF benchmarks where $O(|E|) = O(|V|)$, the practical overhead for local guidance construction is thus derived as $O(nw|V|\log(w|V|))$.

Evaluation

Experiments were conducted on a laptop equipped with Intel Ultra 9 185H and 62 GB of RAM. We use "random" scenarios from the MAPF benchmark (Stern et al. 2019), consisting of 25 instances for each map and each number of agents (up to $1,000$), which is commonly used to evaluate MAPF methods. Unless specified, the planner's time limit is set to 30 s, following (Okumura 2023a); otherwise, the planner is considered unsuccessful in solving the instance. This section compares the following methods:

- **LaCAM** denotes a basic implementation without any guidance terms, aligned with (Okumura 2023a).
- **GG** uses global guidance in Eq. (1). The guidance is constructed with *space utilization optimization (SUO)*, as described in (Han and Yu 2022; Okumura 2024).[2] This method precomputes time-independent, less-congested paths for each agent, which reduces the effort required for collision resolution during the planning stage.
- **LG** uses the local guidance in Eq. (1). Unless noted, LG reuses the previous guidance information and refines it only once. The window size is set to $w = 20$, and the collision penalty $\alpha = 3$.
- **LG+GG** uses both global and local terms with Eq. (4).
- **LNS2** (Li et al. 2022) is another leading unbounded suboptimal MAPF algorithm, which falls into the same category as LaCAM for the solution quality guarantee.

All LaCAM-based methods use the swap technique.

Qualitative Comparison

We first show the effect of guidance visually, using a highly constrained environment *maze-128-128-10*, in which agents

[2]We also tested the traffic flow-based method (Chen et al. 2024), but found that SUO with LaCAM produced superior results.

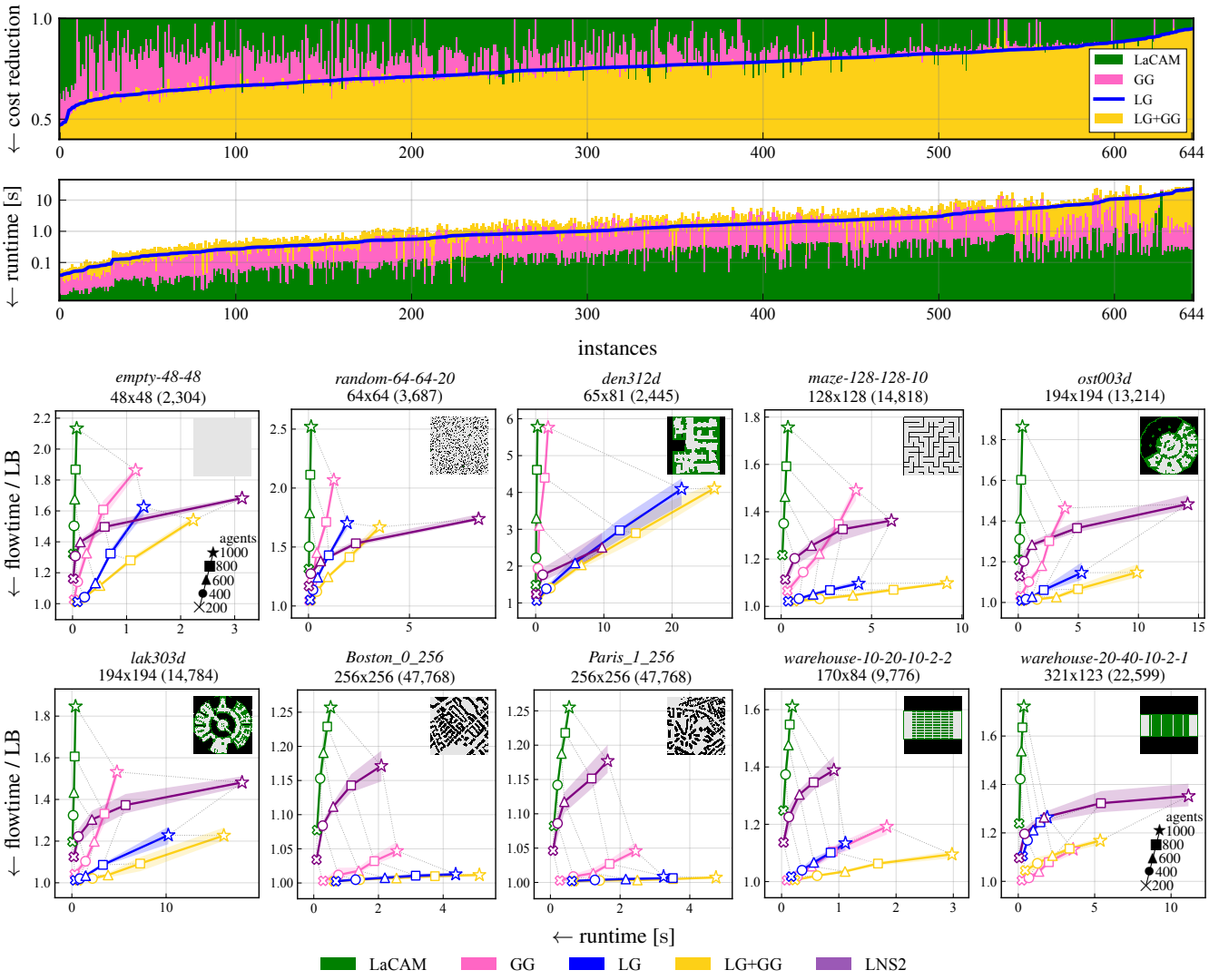


Figure 3: Performance evaluation of suboptimal MAPF methods, assessed by flowtime and runtime. *Top:* Instance-wise comparison. From the MAPF benchmark, we extracted a total of 644 instances across five agent scales $\{200, 400, 600, 800, 1000\}$, selecting five instances per setting from 32 different grids. "cost reduction" is calculated using the LaCAM scores as a reference. The instances are sorted by the LG scores. *Bottom:* Map-wise analysis with varying numbers of agents from 200 to 1000. Each subfigure reports the average over the solved instances among 25 cases per setting. Note that LaCAM-based solvers successfully solved all instances within the 30 s time limit, whereas LNS2 failed on *den312d* when $|A| \in \{800, 1000\}$. Flowtime scores are normalized by a trivial lower bound $\sum_{i \in A} \mathrm{dist}(s_i, g_i)$. The number of vertices for each grid is shown with parentheses.

can easily become congested, making it challenging to find near-optimal solutions. Figure 2 contrasts four LaCAM's solutions: one without guidance, with global guidance (GG), local guidance (LG), or both (LG+GG).

GG can provide coarse, time-independent information about congestion mitigation, resulting in diversified vertex usage in its solution, particularly in area-A. This comes with a cost reduction from the original LaCAM. Meanwhile, it fails to mitigate congestion in area-B, where most agents have a unique shortest start-goal path segment.

This is where LG stands out. It quantitatively and qualitatively smooths out agent flow by providing more detailed, spatiotemporal cues to mitigate local congestion in bottleneck regions (e.g., area-B). Indeed, LG discourages the planner from overusing specific vertices, resulting in a 38% reduction in cost. LG+GG inherits features from both LG and GG, resulting in slightly better performance than LG. More examples appear in the Appendix.

Quantitative Comparison

We next evaluate the guidance effect quantitatively through various maps. As summarized in Fig. 3, overall, the proposed LG substantially reduces the solution cost. With a few exceptions, this reduction exceeds that of GG. Although LG

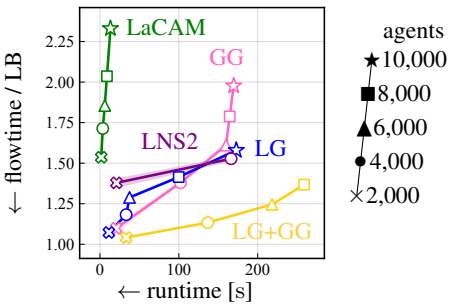


Figure 4: Scalability test on *warehouse-20-40-10-2-2*.

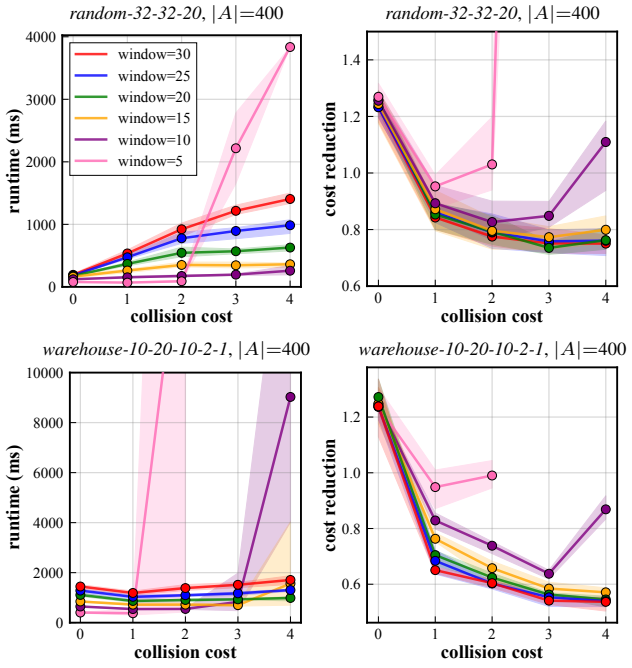


Figure 5: Effect of the collision cost α and window size w. "cost reduction" represents the ratio from LaCAM.

incurs higher computational overhead than GG—due to the guidance reconstruction at every configuration generation—the additional runtime remains within a few seconds in most cases, even with $1,000$ agents. We argue that this overhead is a worthwhile trade-off for the marked improvement in solution quality. Notably, LG outperforms LNS2 in most scenarios, delivering better solutions in less time. These results suggest that local guidance establishes a new Pareto frontier in MAPF, balancing speed and solution quality.

Broadly, LG achieves compelling performance, while GG surpasses LG in *warehouse-20-40-10-2-1*. This is because GG effectively streamlines agent traffic flow, particularly in workspaces with many narrow corridors, by suppressing wasteful back-and-forth movements within them (see also the Appendix, showing the vertex-usage heatmap like Fig. 2). In contrast, results from the other maps indicate that mitigating local congestion is more effective in general. Combining LG and GG compensates for each other's shortcomings and yields the best solution quality among all tested solvers. However, this comes at the cost of additional computation for both global and local guidance, and the performance gain over LG alone remains modest. Developing more effective integration of global and local guidance remains an important direction for future work.

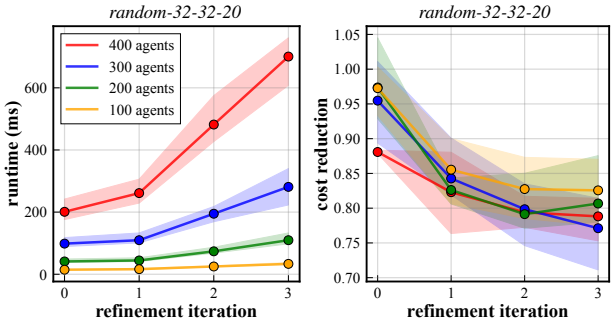


Figure 6: Effect of the plan iteration and initialization. "0" plans the paths from scratch without further refinement. "1" uses the path cache and refines it once, and so on.

Scalability. Figure 4 evaluates the scalability of LG with up to $10,000$ agents on the warehouse map, under an extended time limit of $300\,\mathrm{s}$. While the fast construction of GG at this scale has been reported as challenging (Okumura 2024), LG effectively reduces the solution cost (by approximately 30%) with moderate planning time, remaining significantly faster than both GG and LNS2. These results suggest that the notion of local guidance is effective not only for practical problem sizes, as shown in Fig. 3, but also remains effective in ultra-large-scale MAPF scenarios.

Design Justification

Collision Cost and Window Size. The coefficient α used in Eq. (2) to penalize collisions affects both solution quality and the runtime, together with window size w, as illustrated in Fig. 5, which shows two typical cases. There is a sweet spot; overweighting collisions may worsen the resulting solutions because agents are encouraged to be conservative to avoid them, while underweighting yields faster planning but leaves room for improvement in congestion mitigation. Likewise, enlarging the window size can improve the quality of the solution up to a certain extent, but at the expense of computation time.

Plan Iteration and Initialization. The guidance construction process is essentially non-stationary, as the guidance path for each agent changes in consideration of the others. We tackle this issue by caching the previous result and using it for bootstrapping. Figure 6 demonstrates its effect. Our results so far are obtained with one refinement iteration, while removing the cache, which corresponds to zero refinement, significantly drops the performance. Further refining the guidance can improve its quality, but the benefits gradually diminish, and the computation time becomes costly.

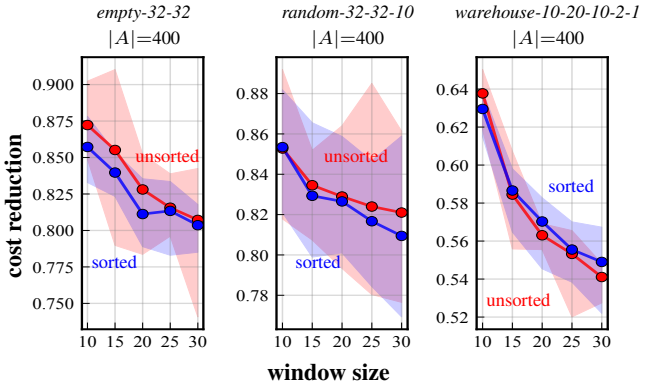


Figure 7: Effect of the agent order used in the guidance construction. The runtime difference is negligible and omitted.

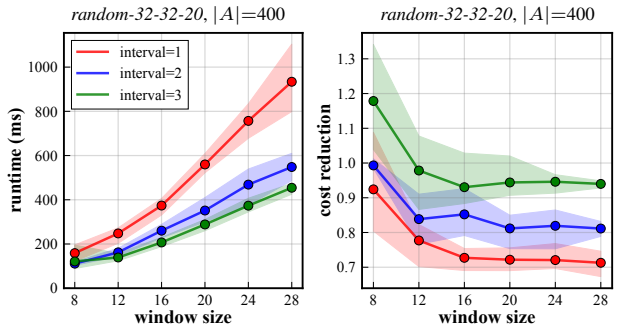


Figure 8: Effect of the frequency of the local guidance update. "1" updates every configuration generation, "2" updates every two generations, and so on.

Agent Order. When constructing the guidance, the agent order (Alg. 1 in Alg. 1) has a design choice. We sort the agents according to the number of collisions as described. Figure 7 illustrates the typical behavior, showing performance improvements across varying window sizes, compared to the unsorted case. Unlike PP, whose performance greatly depends on the ordering, this modest effect occurs as the guidance *indirectly* facilitates collision-free path computation. Meanwhile, the *warehouse-10-20-10-2-1* map is an exception; it contains numerous narrow passages, which may cause the agents to exhibit oscillatory behavior with the sorting operation. We note that the differences in computation time are negligible.

Less frequent guidance update is ineffective. One might think that local guidance construction is not required for each PIBT call. Instead, it would be more computationally attractive if we could update the guidance less frequently. Unfortunately, Fig. 8 shows that this is not the case. This compares local guidance update frequencies: updating every configuration generation (default), every two generations, and every three. Although there is a speed gain, the results show that less frequent updates could worsen performance compared to the original LaCAM. Instead, updating the guidance every time with smaller window sizes is faster yet results in a higher-quality solution. This suggests that "live" guidance, based on up-to-date observations, is essential for mitigating local congestion.

Combined with Anytime Refinement

In real-time planning situations, suboptimal MAPF methods can refine solutions within a given time budget after discovering initial solutions. Here, there are two options: *(i)* allocate more time to find better initial solutions and less time to refine them; or *(ii)* allocate more time to refine them with quicker, but highly suboptimal initial solutions. Since constructing guidance—either globally or locally—can improve the quality of the initial solution, albeit at a computational cost, we investigate which real-time planning strategy results in superior solutions at the end. In particular, following (Okumura 2024), we adopt a popular anytime MAPF scheme called LNS (Li et al. 2021a; Okumura, Tamura, and Défago 2021), which repeatedly selects a subset of agents and refines their paths by PP (Silver 2005). For each subset selection, 1–30 agents are chosen at random, while four LNS processes are run concurrently (Chan et al. 2024b).

Figure 9 compares LaCAM and those with guidance, followed by LNS refinement. With fewer agents (i.e., less congestion), LNS quickly refines solutions, yielding similar final outcomes. Meanwhile, densely populated scenarios show that superior initial solutions lead to substantial performance differences by the planning deadline, as smoothing highly interacting paths is challenging. The results imply that, anytime MAPF methods with the current leading refinement scheme should devote more time to finding better initial solutions in such a dense setup.

We further compare LG+LNS with `lacam3` (Okumura 2024), one of the most advanced MAPF solvers available today. At its core, it employs GG and LNS, but it also uses other advanced techniques that are useful during the initial solution discovery and the refinement stages. Note that the runtime required by `lacam3` for initial solutions is nearly identical to that of GG. The result, shown in Fig. 10, reflects the trends observed thus far; LG outperforms `lacam3` in finding better initial solutions, except on *warehouse*-type maps. This leads to the final solutions that are comparable to and sometimes superior to those of `lacam3`.

Conclusion

This paper demonstrated that local guidance significantly improves the performance of the leading configuration-based MAPF method, LaCAM, incurring a non-negligible yet justifiable computational overhead. The guidance construction is based on windowed decoupled planning, a concept that frequently appears in MAPF algorithms, and is therefore easy to implement. Considering its ability to resolve congestion in bottleneck areas, where many agents inevitably use these, we believe it is worthwhile to invest more research effort in this concept, not limited to LaCAM. One direct application is lifelong MAPF, where PIBT has proven to be a highly effective solution for building strong planners (Jiang et al. 2025; Yukhnevich and Andreychuk 2025).

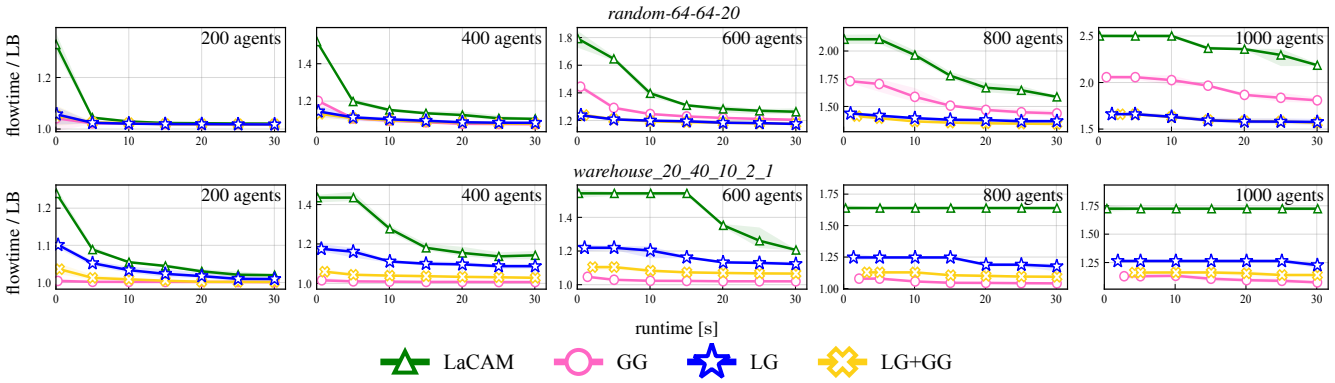


Figure 9: Anytime MAPF performance by LaCAM (+guidance) followed by LNS.

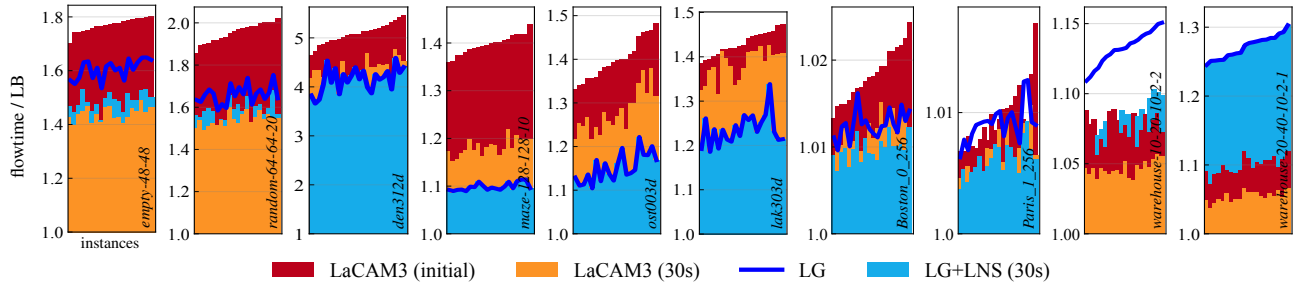


Figure 10: Instance-wise comparison with `lacam3`, using 25 instances with 1000 agents for each map.

Acknowledgments

This research was supported by a gift from Murata Machinery, Ltd.

References

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

Chan, S.-H.; Chen, Z.; Guo, T.; Zhang, H.; Zhang, Y.; Harabor, D.; Koenig, S.; Wu, C.; and Yu, J. 2024a. The League of Robot Runners Competition: Goals, Designs, and Implementation. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

Chan, S.-H.; Chen, Z.; Lin, D.-L.; Zhang, Y.; Harabor, D.; Huang, T.-W.; Koenig, S.; and Phan, T. 2024b. Anytime Multi-Agent Path Finding using Operation Parallelism in Large Neighborhood Search. In *Proceedings of International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*.

Chen, Z.; Harabor, D.; Li, J.; and Stuckey, P. J. 2024. Traffic flow optimisation for lifelong multi-agent path finding. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Erdmann, M.; and Lozano-Perez, T. 1987. On multiple moving objects. *Algorithmica*.

Gandotra, N.; Veerapaneni, R.; Saleem, M. S.; Harabor, D.; Li, J.; and Likhachev, M. 2025. Anytime Single-Step MAPF Planning with Anytime PIBT. *arXiv preprint arXiv:2504.07841*.

Han, S. D.; and Yu, J. 2022. Optimizing space utilization for more effective multi-robot path planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Jain, R.; Okumura, K.; Amir, M.; and Prorok, A. 2025. Graph Attention-Guided Search for Dense Multi-Agent Pathfinding. *arXiv preprint arxiv:2510.17382*.

Jiang, H.; Wang, Y.; Veerapaneni, R.; Duhan, T.; Sartoretti, G.; and Li, J. 2025. Deploying Ten Thousand Robots: Scalable Imitation Learning for Lifelong Multi-Agent Path Finding. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

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 Annual Symposium on Combinatorial Search (SoCS)*.

Kato, T.; Okumura, K.; Sasaki, Y.; and Yokomachi, N. 2025. Congestion Mitigation Path Planning for Large-Scale Multi-Agent Navigation in Dense Environments. *IEEE Robotics and Automation Letters (RA-L)*.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P.; and Koenig, S. 2021a. Anytime multi-agent path finding via large neighborhood search. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

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. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

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 AAAI Conference on Artificial Intelligence (AAAI)*.

Okumura, K. 2023a. Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Okumura, K. 2023b. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Okumura, K. 2024. Engineering LaCAM*: Towards Real-Time, Large-Scale, and Near-Optimal Multi-Agent Pathfinding. In *Proceedings of International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority Inheritance with Backtracking for Iterative Multi-agent Path Finding. *Artificial Intelligence (AIJ)*.

Okumura, K.; and Nagai, H. 2025. Lightweight and Effective Preference Construction in PIBT for Large-Scale Multi-Agent Pathfinding. In *Proceedings of Annual Symposium on Combinatorial Search (SoCS)*.

Okumura, K.; Tamura, Y.; and Défago, X. 2021. Iterative Refinement for Real-Time Multi-Robot Path Planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Phillips, M.; and Likhachev, M. 2011. Sipp: Safe interval path planning for dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Shen, B.; Chen, Z.; Cheema, M. A.; Harabor, D. D.; and Stuckey, P. J. 2023. Tracking progress in multi-agent path finding. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

Silver, D. 2005. Cooperative Pathfinding. In *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

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 Annual Symposium on Combinatorial Search (SoCS)*.

Tan, J.; Luo, Y.; Li, J.; and Ma, H. 2025. Reevaluation of Large Neighborhood Search for MAPF: Findings and Opportunities. In *Proceedings of Annual Symposium on Combinatorial Search (SoCS)*.

Wang, K.-H. C.; Botea, A.; et al. 2008. Fast and Memory-Efficient Multi-Agent Pathfinding. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

Wu, W.; Bhattacharya, S.; and Prorok, A. 2020. Multi-robot path deconfliction through prioritization by path prospects. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

Yukhnevich, E.; and Andreychuk, A. 2025. Enhancing PIBT via multi-action operations. In *League of Robot Runners Expo*.

Zhang, Y.; Jiang, H.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2024. Guidance Graph Optimization for Lifelong Multi-Agent Path Finding. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

Appendix

Figure 11 presents further heatmaps corresponding to Fig. 2. In *room-64-64-8*, LG vividly mitigates the local congestion especially in bottleneck regions where many agents need to use, resulting in significant cost reduction compared to GG. Meanwhile, *warehouse-20-40-10-2-1* represents a situation in which GG is more effective, as discussed in the paper.

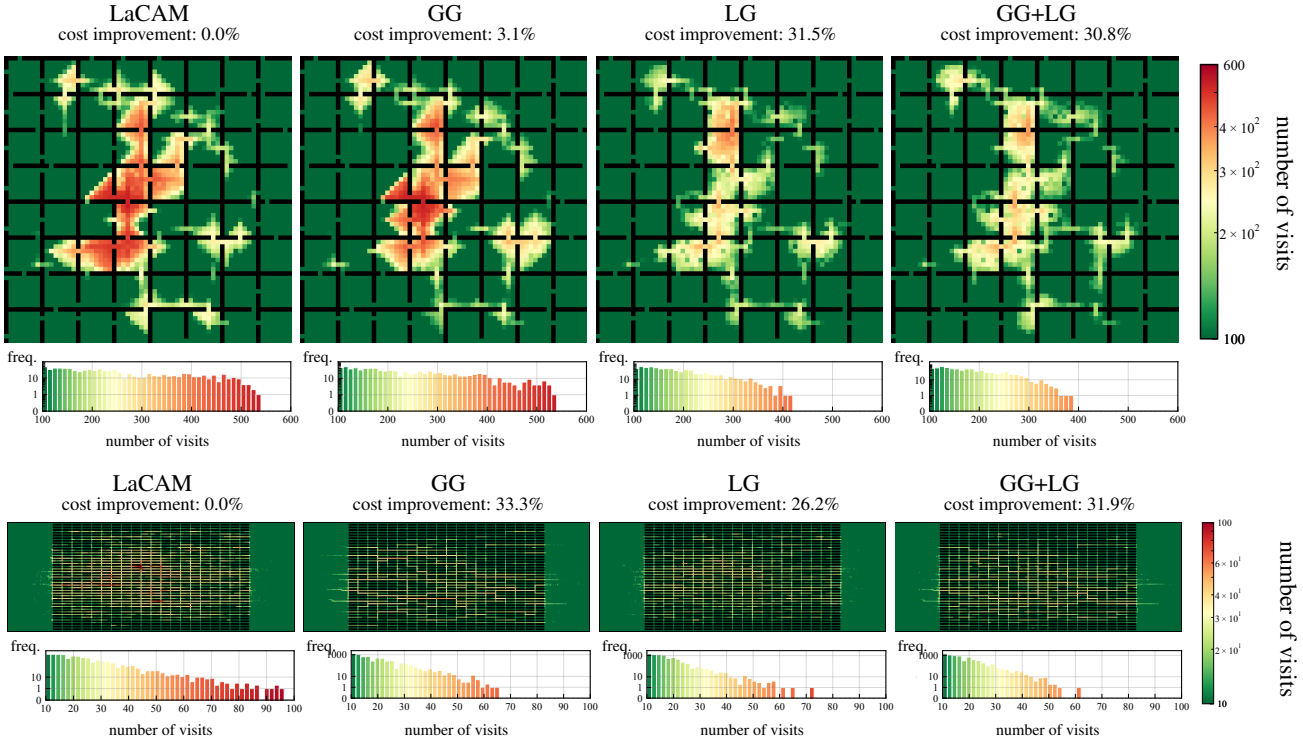


Figure 11: *Top:* Heatmaps showing how many times each vertex is used in each solution on *room-64-64-8* with $1,000$ agents. *Bottom:* For *warehouse-20-40-10-2-1* with $1,000$ agents.