Online Guidance Graph Optimization for Lifelong Multi-Agent Path Finding

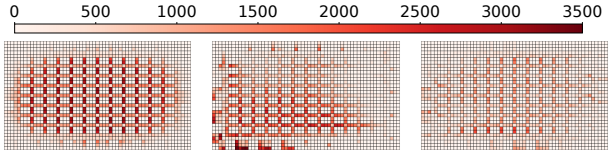
Hongzhi Zang[1*], Yulun Zhang[2*], He Jiang[2], Zhe Chen[3], Daniel Harabor[3], Peter J. Stuckey[3], Jiaoyang Li[2]

[1] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, 100084, China
[2] Robotics Institute, Carnegie Mellon University, Pittsburgh, PA15213, USA
[3] Department of Data Science and AI, Monash University, Melbourne, 3800 VIC, Australia
zanghz21@mails.tsinghua.edu.cn
{yulunzhang, hejiangrivers, jiaoyangli}@cmu.edu
{zhe.chen, daniel.harabor, peter.stuckey}@monash.edu

Abstract

We study the problem of optimizing a guidance policy capable of dynamically guiding the agents for lifelong Multi-Agent Path Finding based on real-time traffic patterns. Multi-Agent Path Finding (MAPF) focuses on moving multiple agents from their starts to goals without collisions. Its lifelong variant, LMAPF, continuously assigns new goals to agents. In this work, we focus on improving the solution quality of PIBT, a state-of-the-art rule-based LMAPF algorithm, by optimizing a policy to generate adaptive guidance. We design two pipelines to incorporate guidance in PIBT in two different ways. We demonstrate the superiority of the optimized policy over both static guidance and human-designed policies. Additionally, we explore scenarios where task distribution changes over time, a challenging yet common situation in real-world applications that is rarely explored in the literature.

Code — https://github.com/zanghz21/OnlineGGO
arXiv — https://arxiv.org/abs/2411.16506



(a) No guidance. (b) Offline guidance. (c) Online guidance.

Figure 1: Comparison of no guidance, offline guidance (Zhang et al. 2024), and our online guidance with a simulation of 5,000 timesteps with 600 agents in a warehouse map of size 33 $\times$ 57 with 1,091 non-obstacle cells. The average and standard deviation of throughput over 10 simulations for no guidance, offline guidance, and online guidance are 3.18 ± 0.04, 6.42 ± 0.09, and 8.66 ± 0.04, respectively. The heatmaps show the number of times the agents take wait action in each cell, approximating the level of congestion. Our online guidance results in the most balanced traffic and thus less congestion and higher throughput.

1 Introduction

We study the problem of optimizing a *guidance policy* capable of dynamically updating guidance graphs with optimized edge weights to guide the agents' movement and improve throughput in lifelong Multi-Agent Path Finding. Multi-Agent Path Finding (MAPF) (Stern et al. 2019) aims to compute collision-free paths for agents from their starts to goals. The lifelong variant of MAPF (LMAPF) constantly assigns new goals to agents upon arriving at their current goals and aims to maximize *throughput*, namely the number of goals reached per timestep. LMAPF has wide applications in automated warehouses (Li et al. 2021b; Zhang et al. 2023b,a) and game character control (Ma et al. 2017b; Jansen and Sturtevant 2008).

Many algorithms have been proposed to solve LMAPF. The *replan-based* algorithms (Ma et al. 2017a; Li et al. 2020; Kou et al. 2020; Damani et al. 2021) decompose LMAPF into a series of MAPF problems and solve them sequentially. Although replan-based algorithms possess high solution quality for small problems, they scale poorly to real-world scenarios with large maps, large numbers of agents, and limited planning time. Meanwhile, the *rule-based* algorithms (Wang and Botea 2008; Okumura et al. 2019; Yu and Wolf 2023) compute a shortest path or a shortest-distance heuristic for each agent without considering the collisions and use pre-defined rules to resolve the collisions. Rule-based algorithms run very fast and scale to extremely large maps and large numbers of agents.

However, rule-based algorithms have no guarantee on the solution quality. Zhang et al. (2024) have shown that with 150 agents, the throughput of RHCR, a state-of-the-art *replan-based* algorithm, is 24.2% better than PIBT, a state-of-the-art *rule-based* algorithm, in a 33 $\times$ 36 warehouse map. To improve the solution quality of rule-based algorithms, prior works (Zhang et al. 2024; Chen et al. 2024; Yu and Wolf 2023; Li and Sun 2023) have explored providing *guidance* to the agents such that they automatically avoid congested areas, thereby improving throughput. Most previous works adopt a static *offline* guidance, with the *guidance graph* (Zhang et al. 2024) providing a versatile and state-of-the-art representation of guidance. Upon optimizing an of-

*These authors contributed equally.

fline guidance for PIBT, the throughput gap between PIBT and RHCR is reduced to less than 4.2% (Zhang et al. 2024). Nevertheless, the offline nature of the guidance graph assumes that the area of congestion does not change, an assumption that does not always hold in the real world. For example, in automated warehouses, task distribution can shift because of changes in the distribution of orders.

Therefore, instead of optimizing an offline guidance graph that provides static guidance, we optimize an *online guidance policy* capable of adapting a dynamic guidance graph over time based on real-time traffic information to improve the throughput of PIBT-based LMAPF algorithms. Figure 1 shows the traffic congestion resulting from different guidance.

To incorporate the online guidance policy into rule-based LMAPF algorithms, we design two pipelines. One directly uses the guidance policy to dynamically generate guidance graphs based on real-time traffic information and replaces uniform edge weights with the generated guidance graphs. The other uses the dynamically generated guidance graphs to adaptively plan better guide paths and move the agents along the guide paths while resolving collisions. To optimize the guidance policy, we follow Zhang et al. (2024) in using Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) (Hansen 2016), a single-objective black-box optimization algorithm, to optimize the policy.

We make the following contributions: (1) We generalize the offline, static guidance graph to an online, dynamic guidance policy capable of updating the guidance graph based on real-time traffic information. (2) We propose two methods to incorporate guidance policy into PIBT. (3) We address the issue of dynamic task distribution, a critical concern in industrial settings. Our online guidance policy was tested in LMAPF simulations with both static and dynamic task distributions. The results demonstrate its advantages over offline guidance with an improvement in throughput of up to 30.75%, and over human-designed online guidance with an improvement of up to 31.59% across various maps and algorithms.

2 Preliminary and Related Work

2.1 LMAPF and LMAPF Algorithms

Definition 1 (Lifelong MAPF (LMAPF)) *Given a graph $G(V, E)$, a set of agents and their corresponding start and goal locations, LMAPF aims to move all agents from their start to their goal locations while avoiding collisions. Furthermore, agents are constantly assigned new goals upon reaching their current goals. At each timestep, agents may either move to an adjacent vertex or remain stationary. Collisions happen when two agents are at the same vertex or swap locations in the same timestep. The objective is to maximize throughput, defined as the average number of goals reached per timestep.*

LMAPF algorithms mainly include *replan-based* algorithms (Li et al. 2021b; Liu et al. 2019; Li et al. 2021b) and *rule-based* algorithms (Wang and Botea 2008; Okumura et al. 2019; Yu and Wolf 2023). Replan-based algorithms decompose the LMAPF problem into a sequence of MAPF problems and rely on MAPF algorithms to solve them. RHCR (Li et al. 2021b), Rolling-Horizon Collision Resolution, is the state-of-the-art of this category. For every h timesteps, RHCR replans collision-free paths of all agents within a time window of w timesteps ($w \geq h$), ignoring collisions beyond it. While RHCR has state-of-the-art throughput with a small number of agents in small graphs (or "maps" as we will refer to them later) (Li et al. 2021b), it scales poorly to instances with large numbers of agents and limited planning time. Prior works have shown that the throughput of RHCR drops to almost zero with more than 200 agents in a 33 $\times$ 36 small warehouse (Zhang et al. 2023a,b) with a per-5-timestep planning time limit of 60 seconds. Even with optimized guidance graph, Zhang et al. (2024) shows that RHCR does not scale to more than 250 agents in the same map.

Rule-based algorithms, on the other hand, run much faster than replan-based ones at the expense of solution quality. Rule-based algorithms compute a shortest path or a shortest distance heuristic for each agent without considering the collisions and leverage pre-defined rules to move the agents while resolving collisions. PIBT (Okumura et al. 2019), Priority Inheritance with Backtracking, is the state-of-the-art of this category. Chen et al. (2024) shows that the planning time of PIBT per timestep is over 150 times faster than RHCR. Jiang et al. (2024) further shows that PIBT is currently the only existing algorithm that can handle a limited planning time of 1 second for up to $10,000$ agents. Therefore, we focus on improving the throughput of PIBT using guidance.

PIBT. PIBT maintains and adjusts a priority for each agent and leverages a single-timestep rule to move agents. At each timestep, PIBT ranks each agent's actions based on the shortest distance from its next location to its goal, with a preference for shorter distances. By default, an agent always tries to follow its shortest path if no higher-priority agents act as obstacles. Otherwise, a lower-priority agent needs to avoid collisions with higher-priority agents by taking the next preferred action. If a lower-priority agent fails to avoid a collision, PIBT applies a backtracking mechanism that forces higher-priority agents to take their next preferred actions and retries all the movement until all agents can take a collision-free action. Note that PIBT possesses favorable properties; for instance, it is complete for LMAPF on biconnected graphs.

2.2 Guidance in MAPF and LMAPF

While not necessarily using the term "guidance", the general idea of guiding agents' movements is widely explored. Prior works fall into two categories, namely *offline* guidance that provides static guidance to the agents, and *online* guidance in which the guidance changes over time based on real-time traffic information.

Offline Guidance. A static offline guidance usually leverages edge directions or edge costs to encourage the agents to move along certain edges. The pre-defined directions and costs are generated in advance and are not updated during the execution of the algorithm. Wang and Botea (2008) leverage unidirectional edges in a graph to force the agents to move in one direction, eliminating head-to-head colli-

sions. The idea of highway (Cohen, Uras, and Koenig 2015; Cohen et al. 2016; Cohen 2020; Li and Sun 2023) predefines a subset of edges in the graph as highway-edges and encourages the agents to move along those edges.

The state-of-the-art offline guidance is the guidance graph (Zhang et al. 2024). It unifies the representation of guidance by using a directed weighted graph. The edge weights of the guidance graph define the cost of moving and waiting for agents at different locations. The edge weights of the guidance graphs are optimized automatically either directly using a single-objective derivative-free optimizer CMA-ES (Hansen 2016) or indirectly with Parameterized Iterative Update (PIU), an iterative update procedure. Starting with a uniform guidance graph with all edge weights being 1, PIU runs the LMAPF simulator to collect the average traffic information, which is used by a parameterized update model to update the guidance graph. The updated model is then optimized by CMA-ES for PIU to generate high-throughput guidance graphs.

In this work, we adopt the definition of guidance graphs from previous work (Zhang et al. 2024), with slight rephrasing.

Definition 2 (Guidance Graph) *Given a graph $G(V, E)$, a guidance graph is a directed weighted graph $G_g(V_g, E_g, \boldsymbol{\omega})$, where $V_g = V$, and $E_g = \{(u, v) | u, v \in V\}$ encodes the action costs in every single vertex, including moving and waiting, for all agents. The action costs are represented collectively as the edge weights $\boldsymbol{\omega} \in \mathbb{R}_{>0}^{|E_g|}$.*

Online Guidance. Online guidance is able to adapt the guidance based on real-time traffic information during the execution of the LMAPF algorithm. Jansen and Sturtevant (2008) assign a direction vector to each location in the graph and encourage agents to move along the direction vector by setting the movement cost of that location to be inversely related to the dot product of the direction vector and the edge cost. The direction vectors are computed from the past traffic information by using a human-designed function. Han and Yu (2022) compute a handcrafted temporal heuristic function to estimate the movement cost, guiding the agents' movement. Similarly, Chen et al. (2024) and Skrynnik et al. (2024) collect the planned paths of all agents and use human-designed functions to compute the movement costs. The above methods rely solely on handcrafted functions, which demand significant human effort and greatly limit the effectiveness of online guidance. Yu and Wolf (2023) rely on a trained data-driven model to predict the movement delays of the agents and uses the delays as the movement costs. However, predicting delays is not directly related to throughput, which is the objective of LMAPF.

2.3 Challenge of Online Guidance

Since online guidance is able to adapt the guidance on-the-fly based on real-time traffic information, it should be able to provide better guidance for agents when the traffic pattern shifts abruptly. For example, in automated warehouses in which robots are used to transport packages from one location to another, the distribution of packages might change over time, resulting in different traffic patterns. In this case, it is necessary to provide online guidance to agents.

However, it is non-trivial to incorporate online guidance into LMAPF algorithms because of the computational overhead of computing the heuristic values. And it is unavoidable because PIBT requires the cost-to-go heuristic, which is the shortest distance from the current locations to the goal locations, to plan agents' actions. With an offline guidance graph, the heuristic values only need to be computed once at the beginning. With an online guidance graph, however, the heuristic values of the entire map need to be updated once the guidance graph is updated. To tackle this issue, a recent work (Chen et al. 2024) proposes a variant of PIBT, referred to as Guided-PIBT (GPIBT).

GPIBT. GPIBT first plans a guide path for each agent that minimizes a handcrafted congestion cost equation, which takes other agents' guide-path edge usage into account. It then moves agents to their goals following the guide paths and resolves collisions using PIBT. Since the heuristics that guide agents to follow guide paths only need to be updated while agents are assigned new goals, GPIBT reduces the computational overhead of heuristic computation. However, since the guide paths are not updated until the agents reach their current goals, GPIBT is less flexible than PIBT in terms of the paths that the agents eventually take to reach the goals.

Given the state-of-the-art performance of PIBT and GPIBT, we focus on incorporating automatically optimized online guidance in them, thereby improving their performance in our work.

3 Approach

In this section, we first introduce the policy for generating dynamic guidance, defined in Definition 3. We then discuss how guidance policies are integrated with PIBT-based LMAPF algorithms in Section 3.1. Finally, we provide a detailed explanation of the optimization of the guidance policy in Section 3.2.

Definition 3 (Guidance Policy) *Given a guidance graph $G_g(V_g, E_g, \boldsymbol{\omega})$, a guidance policy is a function $\pi_{\boldsymbol{\theta}} : \boldsymbol{O} \to \mathbb{R}_{>0}^{|E_g|}$ that computes the updated edge weights $\boldsymbol{\omega'} \in \mathbb{R}_{>0}^{|E_g|}$ given the observation $\boldsymbol{o} \in \boldsymbol{O}$ collected in a LMAPF simulation. The policy $\pi_{\boldsymbol{\theta}}$ is parameterized by $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ is the space of all parameters.*

Depending on the LMAPF algorithm, the observation of guidance policy consists of one or more past traffic patterns, the current distribution of tasks, and future planned paths. We discuss the choice of observation in Section 3.1.

3.1 Incorporating Guidance Policy

Here, we discuss two ways of using guidance graphs and guidance policies in LMAPF algorithms.

Direct Planning. Most LMAPF algorithms directly plan on the guidance graph by minimizing the sum of action costs encoded in the guidance graph. PIBT (Okumura et al. 2019) falls into this category. Figure 2 shows the overview of this method. Starting from a uniform guidance graph with all weights being 1, we run the LMAPF algorithm for m

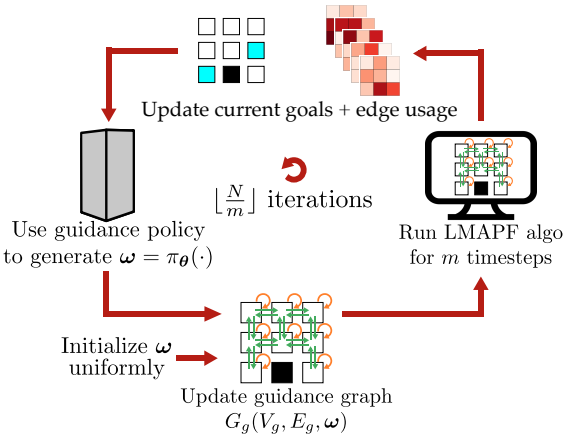


Figure 2: Overview of incorporating guidance policy with Direct Planning algorithms like PIBT.

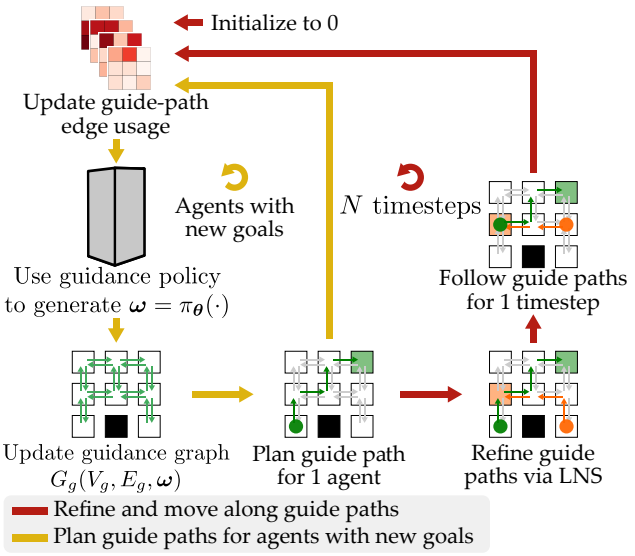


Figure 3: Overview of incorporating guidance policy with Guide-Path Planning algorithms like GPIBT.

timesteps. After m timesteps, we obtain the *edge usage* (i.e., the number of timesteps that each edge in E_g is used by some agent) of the past m timesteps and the currently assigned goals of the agents. The edge usage reflects the recent traffic information, and the current goals project the future possible congestion locations. Upon obtaining the edge usage and current goals, we use the guidance policy to update the guidance graph, starting a new iteration.

Since we end the LMAPF simulation after N timesteps, we run the above procedure for $\lfloor \frac{N}{m} \rfloor$ iterations. By choosing m, we control the trade-off between the adaptability of the online updated guidance graph and additional computational costs associated with the online update mechanism. This is because we need to maintain a heuristic table that contains the shortest path length between every pair of vertices on the guidance graph, which is used for PIBT to rank the actions for every agent at every timestep. [1] Admittedly, the heuristics table needs to be updated every time the guidance graph is updated, thereby making the LMAPF algorithm more computationally expensive. During implementation, we use techniques to reduce recomputation costs. Since the guidance graph updates frequently, calculating the shortest paths for all vertex pairs often results in unnecessary computations. Within one update interval, agents have a limited number of goal locations, and it is likely that some locations on the map are never visited by any agents. Instead of computing the shortest distances for all vertex pairs, we refer Silver (2005) and use a lazy mechanism. Each agent maintains a search tree rooted at its goal location. When the shortest distance at a location is queried, the tree expands until it finds this location, saving a lot of unnecessary computation.

[1] Following Zhang et al. (2024), we modify PIBT's neighbor ranking. Instead of ranking neighbors by their shortest distance to the goal, we rank them by the sum of the action cost from the current location to the neighbors (including itself) and the distance from the neighbors to the goal. This accounts for varying costs in the guidance graph and enables agents to stay at their current location due to self-edges.

Guide-Path Planning. Given the heavy computational overhead of repeatedly computing heuristic tables or trees, another approach is to use guide paths. Figure 3 shows the overview. We generally follow the GPIBT framework (Chen et al. 2024) but modify the guide-path generation process. At each timestep, GPIBT plans guide paths for each agent assigned a new goal in a sequential manner (yellow loop). It may then use Large Neighborhood Search (LNS) (Li et al. 2021a) to refine the guide paths for some randomly selected agents. Finally, each agent moves in a PIBT style, ranking its neighbors according to the distance on the original graph (instead of the guidance graph) to their guide paths (red loop). This procedure is repeated for N timesteps.

We incorporate our guidance policy during guide-path generation. The guide paths for agents are the shortest paths from their previous goal locations to their current goal locations on the guidance graph. After planning guide paths for each agent, the guidance graph updates. GPIBT originally uses a human-designed function to compute the guidance graph weights based on all agents' guide-path edge usage. In our method, we replace this equation with our guidance policy. The guide-path edge usage is initialized to 0 and updated whenever any agent's guide path changes. This approach combines past and future traffic information, as each agent follows its guide path from its previous goal to its current location and will continue following its guide path until the agent reaches its current goal (with possible temporary deviations to avoid collisions), at which point a new guide path is generated to guide it to its next goal. Note that for GPIBT, the guidance graph is only used to compute the guide paths, and there is never a "wait" action on the guide paths. Therefore, there is no need to incorporate self-edges on the guidance graph.

Compared to direct planning, guide-path planning minimizes the effort on heuristic updates because it does not

maintain the pairwise distances between vertices. It only computes the agents' guide paths once when an agent is assigned a goal. If agents deviate from their guide paths, they do not plan new paths but try to return to the guide paths, which is much less computationally intensive.

3.2 Guidance Policy Optimization

We aim to optimize the parameter $\boldsymbol{\theta}$ of the guidance policy π_θ to maximize the throughput given by the LMAPF simulators. Following Zhang et al. (2024), we use CMA-ES (Hansen 2016), a single-objective derivative-free optimization algorithm, to optimize $\boldsymbol{\theta}$. CMA-ES maintains a multi-variate Gaussian distribution to represent the distribution of solutions. Starting with a standard normal Gaussian, CMA-ES samples a batch of b parameter vectors $\boldsymbol{\theta}$, forming b guidance policies. It then evaluates these guidance policies by running the given LMAPF simulator N_e times to compute the average throughput. Finally, based on the evaluated guidance policies, CMA-ES updates the parameters of the Gaussian towards the high-throughput region of the search space. CMA-ES repeats the above procedure until the number of guidance policy evaluations reaches N_{eval}. We include information on the selection of CMA-ES-related hyperparameters in Appendix A.1.

Note that if the dimension of $\boldsymbol{\theta}$ is too large, it becomes difficult for CMA-ES to optimize. Therefore, we use a Convolutional Neural Network (CNN) for PIBT with 3,119 parameters and a specialized windowed quadratic network with 560 parameters. Detailed information can be found in Appendices A.2 and A.3. Admittedly, the design of our current guidance policy architecture can only handle 4-neighbor grids instead of general graphs. However, 4-neighbor grids are currently the main focus of the community. Moreover, the concept of applying the guidance policy is not limited to the current network structures.

4 Experiments and Analysis

4.1 Experiment Setup

We conduct experiments comparing online and offline guidance, optimized guidance policies versus human-designed guidance policies, and the advantages of online guidance with dynamic task distribution. We also compare the runtime of all algorithms. Then, we present results for guidance policies for GPIBT with LNS. Furthermore, we demonstrate that our approach mitigates deadlock issues in PIBT-based algorithms.

Maps. We conduct experiments on 4 maps: (1) *sortation-33-57*, (2) *warehouse-33-57*, (3) *empty-32-32*, and (4) *random-32-32*, shown on top of Figure 4. The first two have regular patterns and are used to test MAPF algorithms in automated warehouse settings. Specifically, the *sortation* map is the same as in (Chen et al. 2024) and the *warehouse* map is generated by us. The latter two are selected from the MAPF benchmark (Stern et al. 2019).

Task Generation. As shown at the top of Figure 4, the *sortation* and *warehouse* maps have workstations (pink) and endpoints (blue). Each agent's goals constantly alternate between the workstations and the endpoints, simulating the warehousing scenario in which robots pick up items from the workstations and drop them off in chutes or shelves (black) reachable from the endpoints in the middle. For *empty* and *random* maps, goals are sampled from empty spaces (white).

Task Distribution. We consider both static and dynamic task distributions. Static task distribution samples goals uniformly, the most common setting in previous works (Chen et al. 2024; Zhang et al. 2023b,a). For the dynamic task distribution, goals are sampled from the Gaussian or multi-modal Gaussian distribution, where the Gaussian centers change for every 200 timesteps. Concretely, in *sortation* and *warehouse* maps, goals on endpoints are sampled from a Gaussian distribution, and goals on workstations are sampled uniformly. For *empty* and *random* maps, goals are sampled from a multi-modal Gaussian distribution with K Gaussian centers. The hyperparameters for the distribution are provided in Table 3 in Appendix A.1.

Algorithm Comparison. We compare 6 algorithms, each with a different LMAPF solver or guidance approach.

on+PIBT our proposed online guidance policy applied to PIBT. We optimize an online guidance policy, periodically call it to update the guidance graph, and directly plan on the guidance graph.

off+PIBT PIBT with an offline guidance graph, optimized using PIU (Zhang et al. 2024).

on+GPIBT our proposed online guidance policy applied to GPIBT. We optimize an online guidance policy and call it when each agent plans its guide path.

[p-on]+GPIBT GPIBT with periodically updated guidance graph. We use the same pipeline as on+PIBT. The inputs of online policy are past traffic and current goals instead of guide-path information. We present the results of this algorithm to show the advantages of on+GPIBT.

off+GPIBT GPIBT with an offline guidance graph, optimized using PIU (Zhang et al. 2024). The guide paths are generated according to the optimized offline guidance graph.

hm+GPIBT the original GPIBT that uses a human-designed equation as the guidance policy (Chen et al. 2024). Specifically, we compute $\boldsymbol{\omega}$ using the SUM_OVC function (Chen et al. 2024), where each weight is the sum of the guide-path vertex usage and the product of the head-on-head guide-path edge usage.

Note that for the PIBT-based algorithms, we used the "swap" technique mentioned in (Okumura 2023). This is necessary because vanilla PIBT is only deadlock-free on bi-directed graphs, and the *random* map does not meet this criterion. With the "swap" technique, the deadlock issue can be mitigated a lot. Additionally, unless specifically mentioned, we do not include LNS in the main results for all GPIBT-based algorithms, which limits their performance. In one of the following subsections, we present the results with LNS included.

We run all LMAPF algorithms for $N = 1,000$ timesteps. In on+PIBT and [p-on]+GPIBT, we update the guidance graph for every $m = 20$ timesteps. The CPU runtime for all algorithms is measured on a local machine with a 64-core

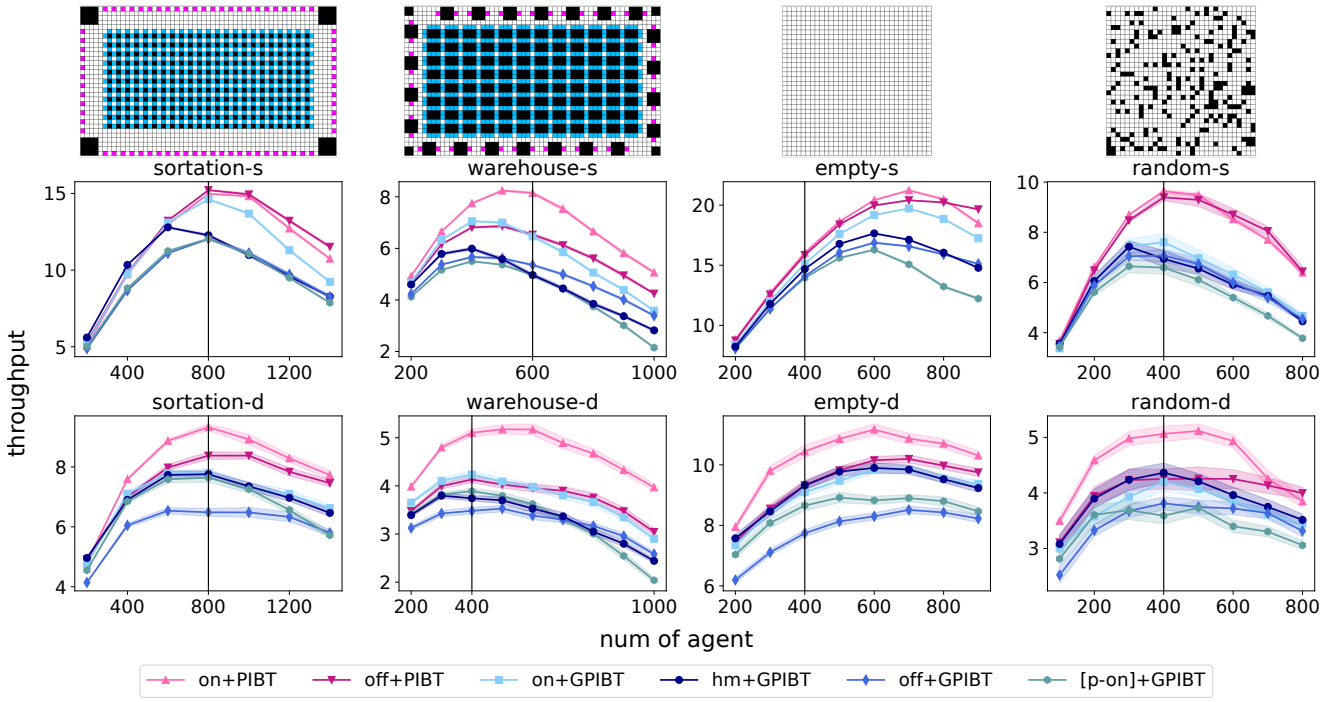


Figure 4: Throughput with different numbers of agents. The black vertical lines show the number of agents that are used to optimize the guidance policies. The solid line shows the average throughput over 50 LMAPF simulations, and the shaded areas denote the 95% confidence interval. "s" and "d" stand for static and dynamic task distribution, respectively.

AMD Ryzen Threadripper 3990X CPU, 192 GB of RAM, and an Nvidia RTX 3090Ti GPU. More compute resource information can be found in Appendix A.4. For relevant software libraries, see Appendix A.5.

4.2 Experiment Results

Figure 4 compares the throughput of different algorithms. To show the generalizability of our approach, we optimize the guidance policies and guidance graphs with the numbers of agents indicated by the black vertical lines. We then evaluate them with various numbers of agents. We highlight several key findings below.

Online vs. Offline Guidance. We first compare our online guidance policy with the offline guidance graph (Zhang et al. 2024) under static task distributions. That is, we focus on comparisons between on+PIBT with off+PIBT and on+GPIBT with off+GPIBT in the first row of Figure 4. Under most settings, on+PIBT and on+GPIBT generally match or outperform off+PIBT and off+PIBT throughout different numbers of agents, respectively. However, in the *sortation* map with static task distribution, on+PIBT is (slightly) worse than off+PIBT because congested locations do not change a lot over time. Therefore, a well-optimized offline guidance graph can alleviate congestion well enough. Besides, there exist outliers on some number of agents because our model is not directly optimized for these cases.

Static vs. Dynamic Task Distribution. We expect that dynamic task distribution gives more advantages to online guidance over offline guidance because the guidance policy can dynamically change the guidance graph based on real-time traffic information, which depends on the task distribution. Clearly in Figure 4, by comparing on+PIBT with off+PIBT, our online guidance policy is more advantageous than the offline guidance graph when moving from static to dynamic task distributions. However, the improvement ratio of on+GPIBT over off+GPIBT does not consistently increase when moving from static to dynamic task distributions. This is because the difference between on+GPIBT and off+GPIBT goes beyond simply online vs. offline: on+GPIBT generates the guidance graph based on the current guide paths, meaning that a new guide path is more likely to avoid areas frequently used by the guide paths of other agents, while off-GPIBT uses a static guidance graph, providing no incentive to generate guide paths that avoid congestion with other guide paths. To isolate the impact of the online mechanism alone, we compare [p-on]+GPIBT against off+GPIBT, where neither algorithm uses guide-path information to generate guidance graphs. In this case, the improvement ratio consistently increases while moving from static to dynamic task distributions. In addition, the comparison between on+GPIBT with [p-on]+GPIBT shows the effectiveness of using guide-path information.

Optimized vs. Handcrafted Online Guidance. We also compare our optimized online guidance policy (on+GPIBT) with the human-designed guidance policy (hm+GPIBT). As shown in Figure 4, on+GPIBT significantly outperforms hm+GPIBT under a static task distribution. For a dynamic task distribution, both on+GPIBT and hm+GPIBT

capture the traffic patterns effectively, resulting in similar performance. In such dynamic settings, we believe that hm+GPIBT is near optimal, considering the observations, task distribution, and the map. To further illustrate this, we optimize the on+GPIBT policy with fewer parameters. The detailed explanation and results are provided in Appendix B.2. However, for the *warehouse* map, on+GPIBT maintains an advantage over hm+GPIBT.

Runtime. Admittedly, online mechanisms incur more computational overhead. on+PIBT is up to 4 times slower than off+PIBT due to the need to recompute the heuristic tables. on+GPIBT is up to 7 times slower than hm+GPIBT and off+GPIBT due to the need to update the guidance graph every time an agent replans its guide path. However, this runtime overhead is often acceptable in practice, as the average runtime per timestep never exceeds 0.028 seconds across all experiments for all algorithms, which is fast enough for realistic settings. Besides, on average, on+PIBT algorithms are two to three times slower than on+GPIBT algorithms because on+GPIBT uses the guide-path mechanism to reduce the computational overhead of heuristics. Numerical results for runtime can be found in Table 4 in Appendix B.1.

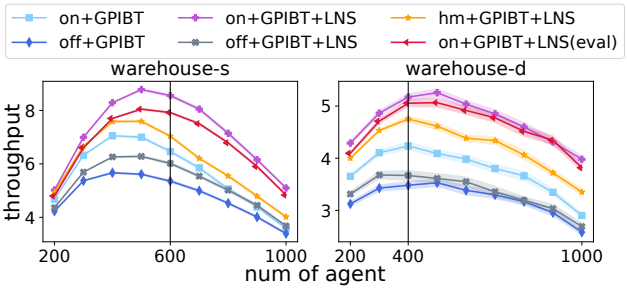


Figure 5: GPIBT with LNS results. The notation of this figure is similar to that in Figure 4.

GPIBT with LNS. Section 3.1 discusses how GPIBT-based methods can incorporate LNS to enhance the quality of guide paths and improve throughput. LNS refines agents' guide paths with an iterative procedure. In each LNS iteration, n_g agents are selected to replan their guide paths from their current locations. The new guide paths are accepted if they result in a lower total guide-path length for all agents on the guidance graph. At each timestep, LNS is terminated either if it exceeds N_{iter} iterations or if it runs for more than t_{LNS} seconds. We focus on the *warehouse* map with both static and dynamic task distribution. LNS parameters are set as $N_{iter} = 10$, $n_g = 10$, and $t_{LNS} = 8$. CMA-ES hyperparameters are kept the same as in the main results.

Figure 5 shows the *throughput-agents* curve with LNS experiments. The results include on+GPIBT+LNS, off+GPIBT+LNS, hm+GPIBT+LNS, as well as on+GPIBT+LNS(eval). We optimize the first three guidance policies or guidance graphs with LNS included in the optimization loop. For on+GPIBT+LNS(eval), we use the on+GPIBT guidance policy and incorporate LNS only during the evaluation phase.

Comparing on+GPIBT with on+GPIBT+LNS and on+GPIBT+LNS(eval) shows that LNS substantially improves the throughput. The throughputs of on+GPIBT+LNS, on+GPIBT+LNS(eval), and the hm+GPIBT+LNS all dominate that of off+GPIBT+LNS, indicating that under the LNS setting, all online methods are better than the offline method. The on+GPIBT+LNS(eval) policy ranks as the second best, only outperformed by the on+GPIBT+LNS approach. This result demonstrates the generalizability of our guidance policy, which is capable of sidestepping the costly optimization of guidance policies with LNS. Interestingly, the improvement of on+GPIBT over off+GPIBT is larger than off+GPIBT+LNS over off+GPIBT, indicating that the online mechanism is more helpful than LNS. That is because off+GPIBT ignores other agents' guide-paths. LNS only helps when an agent deviates from its guide path, replanning the agent's guide path from the deviated location instead of continuing to follow the previously planned guide path. However, ignoring other agents' guide paths cannot lead to higher throughput.

As shown in Tables 4 and 5 in Appendix B.1, all LNS-based algorithms are significantly slower than those without LNS. However, the runtime remains acceptable, with the most expensive algorithm taking only 0.043 seconds per timestep.

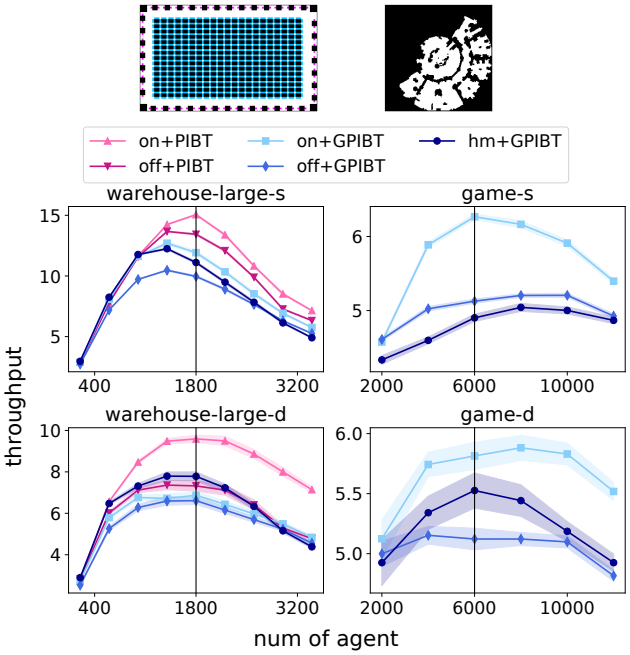
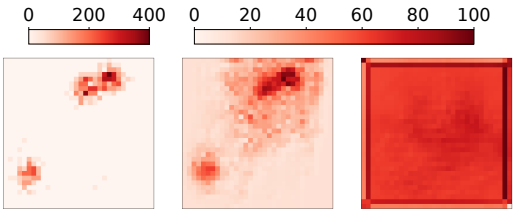


Figure 6: The visualization and the *throughput-agent* curve for the *warehouse-large* map and the *game map*. The notation of this figure is similar to that in Figure 4.

Results on Large Maps. We present results for larger maps, including the *warehouse-large-60-100* and *game-194-194* maps. The *warehouse-large* map is designed based on the layout of *warehouse-33-57*, while the *game* map is taken from the MAPF benchmark (Stern et al. 2019). Visualizations of both maps are shown in Figure 6. For the dynamic task distribution, we apply a Gaussian and multi-



(a) task (b) on+PIBT (c) off+PIBT

Figure 7: Online and offline guidance given task distribution on the *empty* map. Darker colors indicate more goals and higher wait costs.

modal Gaussian distribution on the *warehouse-large* map and the *game* map, respectively.

The left section of Figure 6 shows results for the *warehouse-large* map. Online guidance (on+PIBT and on+GPIBT) consistently outperforms offline guidance (off+PIBT and off+GPIBT). For the static task distribution, our optimized policy (on+GPIBT) surpasses the human-designed function (hm+GPIBT). However, under dynamic task distribution, hm+GPIBT performs better than on+GPIBT on this map. As map size increases, simulation time grows. For the *game* map, we did not optimize PIBT guidance due to computational constraints. However, GPIBT-based algorithms utilize guided paths to reduce heuristic update efforts, demonstrating adaptability to larger maps. The right section of Figure 6 shows on+GPIBT outperforming off+GPIBT and hm+GPIBT.

Algorithm runtimes are presented in Table 4 (Appendix B.1), and CMA-ES hyperparameters are detailed in Table 2 (Appendix A.1).

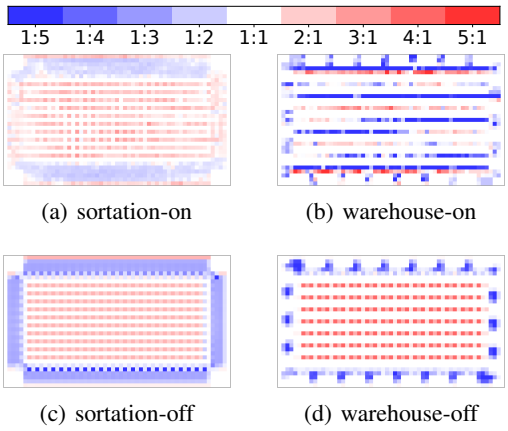
Additional Results. We present the ablation results for the guidance update interval m in the on+PIBT algorithm in Appendix B.2. We discuss the deadlock issue in Appendix B.4.

4.3 Guidance Policy Visualization

In this section, we expand on earlier hypotheses and explain some results mentioned before by visualizing the guidance graph given by on+PIBT.

Online guidance can capture congestion locations. The online guidance policies outperform the offline guidance graphs because they can get more real-time information from the downstream algorithms. For instance, the online policy can capture different congestion locations and alleviate them. Figures 7(b) and 7(c) show the online and offline wait costs given the task distribution shown in Figure 7(a) on the *empty* map with the dynamic task distribution. The tasks are sampled from a 3-mode Gaussian distribution. We observe that the wait costs given by our guidance policy align with the task distribution. Specifically, the areas with more tasks are expected to be more congested. Therefore, our guidance policy generates higher wait costs to encourage the agents to move away from such areas, alleviating congestion. The wait costs given by the offline guidance graph, however, have no correlation with the task distribution at all.

Map structure impacts the improvement ratio of the online guidance policy. Although the *sortation* and *warehouse* maps appear similar, their structural differences lead to varying throughput performances. In the *warehouse* map, the narrower white spaces between workstations (pink) and endpoints (blue) make detours less effective, while the longer corridors among endpoints limit path choices when agents are in the map's center. Consequently, the online guidance policy must direct agents to select appropriate corridors and directions upon entering the central area to minimize congestion. Under the static task distribution, Figure 8 clearly illustrates the alternating preferred directions in the online guidance for the *warehouse* map, a pattern not evident in the *sortation* map. In contrast, offline guidance graphs in both maps direct agents uniformly, favoring one direction in central areas and another in surrounding areas. These differences explain why online guidance significantly improves throughput on the *warehouse* map but has minimal impact on the *sortation* map.



(a) sortation-on (b) warehouse-on

(c) sortation-off (d) warehouse-off

Figure 8: Visualization of the left-to-right edge weights ratio in the guidance graph for the *sortation* and *warehouse* maps, with "on" and "off" representing on+PIBT and off+PIBT, respectively. Red cells indicate a higher left-to-right ratio, while blue cells indicate a higher right-to-left ratio.

5 Conclusion

Lifelong Multi-Agent Path Finding (LMAPF) is a challenging problem with significant practical importance. As the number of agents increases, addressing congestion becomes critical for any solution. We present an approach that dynamically generates guidance graphs, leveraging traffic patterns to alleviate congestion and improve throughput. Our proposed online guidance policy enhances throughput in two state-of-the-art LMAPF algorithms across four maps. This work opens several future directions. First, incorporating the guidance policy into other LMAPF algorithms, such as RHCR (Li et al. 2021b) and Learn to Follow (Skrynnik et al. 2024), could be explored. Second, as our approach uses CMA-ES, which requires many evaluations in LMAPF simulators, future work could explore surrogate-assisted optimization (Zhang et al. 2022; Bhatt et al. 2022; Kent and Branke 2023) to improve sample efficiency.

Acknowledgments

Hongzhi Zang performed her research during her visit to Carnegie Mellon University. This work used Bridge-2 at Pittsburgh Supercomputing Center (PSC) through allocation CIS220115 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

Bhatt, V.; Tjanaka, B.; Fontaine, M.; and Nikolaidis, S. 2022. Deep Surrogate Assisted Generation of Environments. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 37762–37777.

Chen, Z.; Harabor, D.; Li, J.; and Stuckey, P. 2024. Traffic Flow Optimisation for Lifelong Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 20674–20682.

Cohen, L. 2020. *Efficient Bounded-Suboptimal Multi-Agent Path Finding and Motion Planning via Improvements to Focal Search*. Ph.D. thesis, University of Southern California.

Cohen, L.; Uras, T.; and Koenig, S. 2015. Feasibility Study: Using Highways for Bounded-Suboptimal Multi-Agent Path Finding. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, 2–8.

Cohen, L.; Uras, T.; Kumar, T. K. S.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Improved Solvers for Bounded-Suboptimal Multi-Agent Path Finding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 3067–3074.

Damani, M.; Luo, Z.; Wenzel, E.; and Sartoretti, G. 2021. PRIMAL_2: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning - Lifelong. *IEEE Robotics and Automation Letters*, 6(2): 2666–2673.

Han, S. D.; and Yu, J. 2022. Optimizing Space Utilization for More Effective Multi-Robot Path Planning. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 10709–10715.

Hansen, N. 2016. The CMA Evolution Strategy: A Tutorial. *ArXiv*, abs/1604.00772.

Jansen, M. R.; and Sturtevant, N. R. 2008. Direction Maps for Cooperative Pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 185–190.

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 (SoCS)*, 234–242.

Kent, P.; and Branke, J. 2023. Bayesian Quality Diversity Search with Interactive Illumination. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 1019–1026.

Kou, N. M.; Peng, C.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2020. Idle Time Optimization for Target Assignment and Path Finding in Sortation Centers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 9925–9932.

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2021a. Anytime Multi-Agent Path Finding via Large Neighborhood Search. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 4127–4135.

Li, J.; Sun, K.; Ma, H.; Felner, A.; Kumar, T. K. S.; and Koenig, S. 2020. Moving Agents in Formation in Congested Environments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 726–734.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. K. S.; and Koenig, S. 2021b. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 11272–11281.

Li, M.-F.; and Sun, M. 2023. The Study of Highway for Lifelong Multi-Agent Path Finding. *ArXiv*, 2304.04217.

Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019. Task and Path Planning for Multi-Agent Pickup and Delivery. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1152–1160.

Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017a. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 837–845.

Ma, H.; Yang, J.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2017b. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 270–272.

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

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2019. Priority Inheritance with Backtracking for Iterative Multi-agent Path Finding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 535–542.

Silver, D. 2005. Cooperative Pathfinding. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 117–122.

Skrynnik, A.; Andreychuk, A.; Nesterova, M.; Yakovlev, K.; and Panov, A. 2024. Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 16, 17541–17549.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, 151–159.

Tjanaka, B.; Fontaine, M. C.; Lee, D. H.; Zhang, Y.; Balam, N. R.; Dennler, N.; Garlanka, S. S.; Klapsis, N. D.; and

Nikolaidis, S. 2023. pyribs: A Bare-Bones Python Library for Quality Diversity Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 220–229.

Wang, K. C.; and Botea, A. 2008. Fast and Memory-Efficient Multi-Agent Pathfinding. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 380–387.

Yu, G.; and Wolf, M. 2023. Congestion prediction for large fleets of mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 7642–7649.

Zhang, Y.; Fontaine, M. C.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2023a. Arbitrarily Scalable Environment Generators via Neural Cellular Automata. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, 57212–57225.

Zhang, Y.; Fontaine, M. C.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2023b. Multi-Robot Coordination and Layout Design for Automated Warehousing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 5503–5511.

Zhang, Y.; Fontaine, M. C.; Hoover, A. K.; and Nikolaidis, S. 2022. Deep Surrogate Assisted MAP-Elites for Automated Hearthstone Deckbuilding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 158–167.

Zhang, Y.; Jiang, H.; Bhatt, V.; Nikolaidis, S.; and Li, J. 2024. Guidance Graph Optimization for Lifelong Multi-Agent Path Finding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 311–320.