HI-MAPF - Towards Resource-Efficient Multi-Agent Deployment

Bharath Muppasani, Risha Patel, Biplav Srivastava, Vignesh Narayanan
University of South Carolina, USA

***Abstract*— Multi-Agent Path Finding (MAPF) is central to coordinating multiple agents in shared or contested environments. Existing algorithmic solutions to MAPF impose significant requirements on communication, sensing, and computation, limiting their applicability in resource-constrained robotic deployments. For instance, centralized search methods require access to global state information and high-bandwidth channels, while distributed learning-based methods depend on sophisticated onboard perception (e.g., LiDAR, RGB-D cameras) and frequent inter-agent communication. Consequently, these methods incur high hardware, computational, and communication costs that form a critical bottleneck for long-term robotic deployment. To address these limitations, we formulate information-centric MAPF (I-MAPF) and introduce HI-MAPF, a hybrid framework that augments decentralized planning with a lightweight centralized coordinator. We further introduce a method-agnostic metric, Information Units (IU), to quantify coordination overhead as a proxy for deployment cost. We evaluate HI–MAPF against recent communication-focused algorithms in both simulation, across unstructured and structured environments with 8 to 128 agents, and on ground robots in a physical deployment, demonstrating $2\times$ to $510\times$ reduction in information sharing while maintaining high success rates. Our results show that in controlled environments, HI-MAPF eliminates the need for sophisticated onboard perception, expensive compute, and frequent inter-agent communication, enabling coordination through minimal event-triggered alerts.**

I. Introduction

Multi-Agent Path Finding (MAPF) addresses the problem of computing collision-free trajectories for multiple agents navigating a shared or contested environment. It is central to applications like warehouse automation, hospital logistics, search and reconnaissance, and autonomous fleets. In practice, each agent operates under resource constraints, including limited onboard compute, finite energy, and scarce communication bandwidth, which limit the scale at which multi-agent systems can be practically deployed. Despite the problem's practical significance, MAPF is computationally demanding, classified as NP-hard in its general form, which can render traditional centralized methods intractable as the number of agents or the environment complexity increases [1], [2]. Furthermore, practical deployment additionally requires that the solution respect the sensing, communication, and computation constraints of each agent.

Existing MAPF approaches span a range of coordination paradigms, each with distinct deployment implications. Centralized methods such as Conflict-Based Search (CBS) [3] and ICBS [4] assume global knowledge of all agents' start and goal states, the environment, and compute joint plans using a centralized search algorithm. While effective, they incur significant computational overhead as the number of agents grows and introduce a single point of failure in practical deployments. Furthermore, the addition of new agents typically necessitates global replanning, as classical centralized MAPF solvers assume a fixed agent set and compute joint plans accordingly. This limits their suitability for highly dynamic teams where agents frequently enter or leave the system. Decoupled methods, like M^* [5], coordinate agents only during conflict, improving scalability but often sacrificing optimality or completeness (i.e., solution quality). Multi-Agent Reinforcement Learning (MARL) methods such as PRIMAL [2], [6], [7] train agents to plan using field-of-view (FOV)-based local observations, while SCRIMP [8] augments FOV sensing with transformer-based inter-agent communication for improved coordination [9]–[15]. Among these, EPH [16] represents a recent advancement in communication-focused MAPF algorithms, achieving strong success rates on structured maps through ensembled hybrid policies with expert-guided heuristics, outperforming prior neural baselines in communication-efficient coordination. Nevertheless, EPH still requires each robot to carry FOV sensing hardware, perform neural inference when other agents are within FOV (reverting to A* planning otherwise), and maintain inter-agent communication, leaving the fundamental deployment overhead unaddressed. Other representative MAPF techniques proposed in the literature, such as DHC [9], DCC [10], SACHA [11], FOLLOWER [7], ALPHA [13], SIGMA [14], MAPF-GPT [15], use either sophisticated sensing/communication and/or include compute heavy neural computation, and have similar limitations of frequent inter-agent communication, or computationally intensive neural inference, which make long-term deployment challenging. Furthermore, commonly used metrics such as success rate, makespan, and sum-of-costs do not capture the sensing, communication, and computation resources consumed during coordination, making it difficult to assess deployment readiness across paradigms.

To address these limitations, we formulate the MAPF problem from an information-centric perspective (I-MAPF) and propose *Hybrid Information-Centric MAPF (HI–MAPF)*, a novel hybrid framework that combines decentralized planning with a lightweight centralized coordinator and *minimizes* inter-agent information sharing while maintaining solution feasibility, directly supporting long-term robotic deployment by reducing the communication frequency. In HI–MAPF, agents plan independently and receive only minimal, event-triggered alerts from the coordinator when conflicts are anticipated. Further, we introduce Information Units

(IU), a method-agnostic metric that quantifies the total cell-level information each agent must sense or exchange during coordination, serving as a unified proxy for deployment cost across centralized, distributed, and hybrid MAPF paradigms.

We evaluate HI–MAPF through extensive simulation benchmarks and hardware experiments on TurtleBot4 robots. To ensure fair comparison, all baseline methods are deployed in a distributed setup on TurtleBot4 robots. Importantly, our baseline methods (DCC, SCRIMP, EPH, MAPF-GPT) have themselves been evaluated against a broad set of solvers, including CBS, wPBS, ODrM*, PRIMAL, DHC, and SACHA, on the same map configurations we adopt, making HI–MAPF's results transitively comparable to this wider field. Our findings show that this design reduces the total information load by $\mathbf{2}\times$ to $\mathbf{510}\times$ compared to state-of-the-art communication-efficient algorithms (see Table I, Table II).

The remainder of this paper is organized as follows. Section II reviews MAPF background, coordination paradigms, and related work. Section III presents the I-MAPF formulation, the IU metric, and the HI-MAPF framework. Section IV-A describes the experimental setup, including training, hardware deployment, and simulation benchmarks. Section IV-B presents results and discussion, and Section V concludes with limitations and future directions.

II. Background and Literature Review

A. Multi-Agent Path Finding

Let $G = (V, E)$ be an undirected graph, where V is the set of vertices (grid cells) and $E \subseteq V \times V$ is the set of edges connecting adjacent cells. A team of n agents $A = \{a_1, \ldots, a_n\}$ must move from start vertices $s_i \in V$ to goal vertices $g_i \in V$, where $(s_i, g_i) \neq (s_j, g_j), \forall i \neq j : i, j \in \{1, ..., n\}$. Time is discretized into steps $t = 0, 1, 2, \ldots$, and at each step, an agent may either move along an edge or wait. A path for agent a_i is a sequence $\pi_i = (v_0^i, v_1^i, \ldots, v_{T_i}^i)$ with $v_0^i = s_i$ and $v_{T_i}^i = g_i$. A solution to MAPF is $\Pi = \{\pi_1, \ldots, \pi_n\}$, and it is collision-free if for all $i \neq j$ and all t, $v_t^i \neq v_t^j$ (vertex collision free) and $(v_t^i, v_{t+1}^i) \neq (v_{t+1}^j, v_t^j)$ (edge collision free). The primary goal in standard MAPF is typically to find a path set Π that is collision-free [3]. Common efficiency objectives include minimizing the makespan, $\max_i T_i$, minimizing the sum of individual completion times (sum-of-costs), $\sum_i T_i$.

B. Coordination Paradigms

As formalized by Sharon et al. [3], solution approaches to MAPF problems can be categorized based on their coordination strategy. We extend this by focusing on three key elements: state observability (global vs. local), communication (allowed vs. none), and control (centralized vs. decentralized). As shown in Figure 1, in *centralized* paradigms, a global planner with full observability of the entire state controls all agents, and communication is implicit through this central controller. By contrast, other approaches grant agents local control over their actions. Within this category, we draw a key distinction based on communication: In

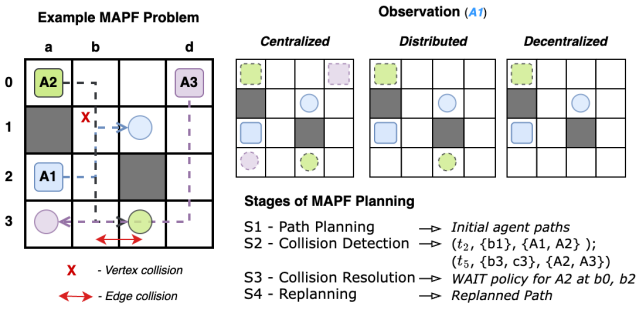


Fig. 1. Example MAPF problem and four-stage pipeline. **Left:** Three agents (A1, A2, A3) navigate a grid with static obstacles (gray), showing a vertex collision (red X) and an edge collision (red arrows). **Top Right:** Information available to agent A1 under centralized, distributed, and decentralized paradigms. **Bottom Right:** The planning stages: S1 (Path Planning), S2 (Collision Detection), S3 (Resolution), and S4 (Replanning).

distributed approaches, agents that plan their own paths are allowed to explicitly exchange information, such as local observations, intended paths, or goals, with other agents to achieve cooperative behavior. In *decentralized* MAPF, as defined in this work, inter-agent communication is eliminated. Agents plan and act entirely independently, relying only on their own path information. Finally, *hybrid* frameworks, like the one we propose in Section III, combine these elements. They typically employ a central coordination mechanism that has full observability to detect conflicts and selectively intervene, while agents otherwise plan in a decentralized manner.

C. Related Works

Existing approaches for MAPF can be broadly categorized into search- and learning-based, and hybrid methods. **Search-based Methods:** Search-based MAPF solvers compute collision-free joint solutions by systematically exploring a constraint space over individual agent plans. Classical MAPF solvers such as CBS [3] plan independently per agent and resolve conflicts by branching on constraints, guaranteeing optimality. Variants including ICBS [4] accelerate search via cardinality heuristics, while suboptimal approaches like LNS [17] iteratively repair conflicting subsets for scalability. PIBT [18] and LaCAM [19] further improve throughput via adaptive prioritization and constraint-guided search. Despite these gains, search-based methods rely on global information and require high-bandwidth channels for state collection, creating scalability limits and a single point of failure impractical for real-world deployment.

Learning-based Methods: Learning-based approaches train decentralized policies under the Centralized Training with Decentralized Execution (CTDE) paradigm. PRIMAL/PRIMAL2 [2], [6], combine imitation and reinforcement learning for scalability, while SCRIMP [8] and MAPF-GPT [15] use transformer-based communication along with FOV-based coordination. DCC [10] augments FOV sensing with a learned selective communication block. DHC [9] exchanges information with K nearest neighbors. However,

real deployment requires each agent to carry onboard perception hardware (e.g., RGB-D, LiDAR) for FOV construction, maintain active communication channels, and perform neural inference at each timestep, increasing hardware cost and energy consumption.

Hybrid Methods: Hybrid approaches combine planning and learning. FOLLOWER [7] pairs congestion-aware A^* subgoal generation with decentralized RL. EPH [16] uses ensembled hybrid policies with expert-guided heuristics, outperforming prior neural baselines on structured benchmarks (`den312d`, `warehouse`). LNS2+RL [20] integrates MARL into LNS for localized conflict resolution. While effective, these designs still require FOV-based sensing and inter-agent message passing, leaving substantial hardware and communication overhead.

Taken together, existing paradigms reveal a gap between algorithmic capability and deployment readiness across three practical dimensions: solution quality (makespan), communication frequency, and total information exchanged (IU). Centralized methods achieve strong solution quality but require global state access, resulting in high information and communication costs. Learning-based methods reduce centralization yet rely on continuous sensing and per-timestep communication, increasing communication frequency and total information exchange. Current hybrid methods improve solution quality and scalability but still carry significant communication overhead. The hybrid framework HI–MAPF, introduced in Section III, is designed to address this gap by minimizing inter-agent information exchange through sparse, event-driven coordination without requiring FOV sensing, thereby improving deployment feasibility under communication and hardware constraints.

III. Methodology

In this section, we propose the I-MAPF formulation, then define the IU metric, and finally present the HI–MAPF framework.

A. Information-Centric MAPF

We characterize MAPF coordination by the information available to each agent i at time t,

$$\mathcal{I}_i(t) = \big(\mathcal{M}_i,\, \mathcal{S}_i,\, \mathcal{O}_j,\, \mathcal{E}_{ij}(t)\big),$$

where $\mathcal{M}_i = (V, E, O)$ encodes (possibly partial) environment knowledge; $\mathcal{S}_i = (s_i^t, g_i^t, \pi_{i|t_s:t_r})$ is the agent's self information; $\mathcal{O}_j = \{(s_j^t, g_j^t, \pi_{j|t_s:t_r})\}_{j\in\mathcal{N}_i(t)}$ is the other-agent information available through sensing or communication; and $\mathcal{E}_{ij}(t)$ denotes event-triggered coordination signals (e.g., predicted collisions, deadlocks, or constraint/alert messages) generated by a coordinator or inferred locally.

Centralized methods use global $\mathcal{M}$ and all agents' trajectories over $\{t_0, \ldots, t_{\max}\}$. Distributed methods restrict $\mathcal{M}_i$ to a local FOV $\mathcal{M}_i^k$ and $\mathcal{O}_j$ to neighbors. Decentralized methods omit $\mathcal{O}_j$ (and thus do not rely on explicit inter-agent information). Hybrid methods selectively reveal additional $\mathcal{O}_j$ and/or constraints in response to $\mathcal{E}_{ij}(t)$.

B. Information Units

Each component of $\mathcal{I}_i(t)$, map knowledge $\mathcal{M}_i$, self-state $\mathcal{S}_i$, neighbor observations $\mathcal{O}_j$, and event signals $\mathcal{E}_{ij}(t)$, contributes to the total information budget each agent consumes during coordination. To quantify this budget in a unified, method-agnostic way across different MAPF paradigms, we introduce the *Information Unit* metric.

Definition 1 (Information Unit (IU)): Let S be a finite set of possible cell states such that $S \supseteq \{\texttt{free}, \texttt{obstacle}\} \cup \{\texttt{occupied}(a) \mid a \in A\} \cup \{\texttt{reserved}(a) \mid a \in A\}$. Let T denote the discrete timestep domain and let $E = \mathbb{R}$ denote the domain of message scalars.

An *IU* is one atomic datum of the form $(v, s, t, e) \in V \times S \times T \times E$, where $v \in V$ is a grid cell, $s \in S$ is its state, $t \in T$ is a timestep (or a null value when time is not applicable), and $e \in E$ is a scalar message entry (or a null value when no message scalar is associated). Any sensing output or transmitted message is modeled as a finite multiset of IUs.

Remark. One IU corresponds to exactly one fact about a single grid cell v representing state s (optionally at time t). This state may encode static map information (e.g., `obstacle`), agent-specific occupancy (e.g., $\texttt{occupied}(a)$), or time-indexed reservations/constraints (e.g., $\texttt{reserved}(a)$). A d-dimensional inter-agent message vector is represented as d distinct IUs, each containing one scalar entry $e \in \mathbb{R}$ Accordingly, composite artifacts such as maps, trajectories, constraints, or messages are represented as collections of such atomic cell-state facts, and their information cost is measured by the number of IUs they contain.

In search-based methods, IU accounts for access to other agents' states or full plans mediated by a central solver. In learning-based methods, IU accumulates per timestep from position broadcasts, action transmissions, FOV-derived observations of other agents, and any explicit messages (e.g., 512-dimensional vectors in SCRIMP). In hybrid methods such as HI–MAPF, IU includes the initial path submission, and conflict-triggered time-indexed constraints or short subpath fragments represented as collections of (v, s, t) facts. In practical robotic deployments, learning-based and hybrid approaches typically incur a one-time environment broadcast consisting of the grid map ($H \times W$ cells) and the $2N$ start and goal locations. Lower IU therefore indicates lower bandwidth, energy, and onboard hardware burden, improving feasibility for long-term robotic deployment.

C. Proposed Approach for HI–MAPF

The design of HI–MAPF is driven by a central principle: minimize the sensing, communication, and computation resources that each agent uses during coordination. HI–MAPF is a concrete realization of the I-MAPF framework within the hybrid MAPF category, in which $\mathcal{I}_i(t) = (\mathcal{M}_i, \mathcal{S}_i, \mathcal{O}_j, \mathcal{E}_{ij}(t))$ is kept minimal by design: $\mathcal{M}_i$ is a one-time broadcast, $\mathcal{O}_j$ is restricted to conflict-triggered neighbor disclosures, and $\mathcal{E}_{ij}(t)$ are the alert tuples introduced in our algorithm. At a high level, each iteration of our hybrid four-stage pipeline proceeds as follows: (S1) each

agent independently plans a trajectory from start to goal using $\mathcal{M}_i$; (S2) a coordinator scans the set of submitted trajectories for vertex/edge conflicts and reports any conflict (time, conflict location/edge, involved agents); (S3) a heuristic controller selects a replanning agent, chooses a bounded replanning window around t_j, and decides what constraints/information to provide; and (S4) the selected agent replans within this bounded window following the constraints and resubmits an updated trajectory. This detect–control–replan loop repeats until no conflicts remain. Every design choice in our framework, from decentralized-by-default planning to event-triggered alerts to tiered repair strategies, is tailored to reduce inter/intra-agent information need, thereby lowering the communication and hardware requirements per agent and enabling extended operational lifespan of the deployed robots. The proposed methodology of the four-stage MAPF pipeline is detailed below.

a) S1: Decentralized Path Planning: Each agent a_k generates an independent trajectory

$$\rho_k = \pi_\theta(s_k, g_k) = (v_0^k, \ldots, v_{\tau_k}^k), \tag{1}$$

where $v_0^k = s_k$, $v_{\tau_k}^k = g_k$, and π_θ is a parameterized RL policy trained to minimize path length and collision risk over a planning horizon H. Let τ_i denote the set of makespan of each agent and $\tau = \{\tau_1, \ldots, \tau_n\}$ be the set of all such makespans. No information about other agents is exchanged during this stage, eliminating the need for FOV sensing hardware or inter-agent communication. In our implementation, A* serves as the planner π_θ at this stage. S1 accesses only $\mathcal{M}_i$, $\mathcal{S}_i = (s_k, g_k)$ from $\mathcal{I}_i(t)$, with $\mathcal{O}_j = \emptyset$ and $\mathcal{E}_{ij} = \emptyset$, contributing 0 inter-agent IU.

b) S2–S3: Centralized Collision Detection and Control: A central module takes as input the independent trajectories of all the agents $\rho = \{\rho_1, \ldots, \rho_n\}$ along with the makespan set τ, and identifies all vertex and edge conflicts, using

$$\begin{aligned} C(\rho, \tau) = \{(t_j, \Delta_c, A_c) \mid (v_{t_j}^k = v_{t_j}^l := \Delta_c) \vee \\ ((v_{t_j}^k, v_{t_{j+1}}^k) = (v_{t_j+1}^l, v_{t_j}^l) := \Delta_c)\} \end{aligned} \tag{2}$$

$$\begin{aligned} A_c = \{\{a_k, a_l\} : \rho_k|\{t_j\} = \rho_l|\{t_j\} = \Delta_c \vee \rho_k| \\ \{(t_j, t_{j+1})\} = \rho_l|\{(t_j, t_{j+1})\} = \Delta_c, \forall k \neq l\}, \end{aligned} \tag{3}$$

where $\rho_k|\{t_j\}$ and $\rho_k|\{(t_j, t_{j+1})\}$ denotes the position of the k^{th} agent at step t_j and steps (t_j, t_{j+1}), respectively. For each conflict $c = (t_j, \Delta_c, A_c) \in C(\rho, \tau)$, which occurs at timestep t_j, the controller issues an alert $\mathcal{A}$ defined as

$$\mathcal{A}(c) = \mu_{c_k} = (a_{c_k}, t_{j-r}, \Delta_c), r \geq j, \tag{4}$$

where policy μ_{c_k} selects an agent $a_{c_k} \in A_c$ via one of three heuristics – (i) *Random*: $a_{c_k} \sim \text{UniformRandom}(A_c)$; (ii) *Farthest*: $a_{c_k} = \arg\max_{a \in A_c} d_{\text{Manh}}(v_{t_{j-r}}^a, g_a)$; or (iii) *Fewest Future Collisions (FFC)*: $a_{c_k} = \arg\min_{a \in A_c} |\{\tilde{c} \mid a \in A_{\tilde{c}}, \tilde{c} \in C(\rho, \tau)\}|$. We use FFC in all experiments. Critically, the coordinator transmits only the conflict location and time to the affected agent, a payload of a few bytes, rather than full trajectories or FOV observations. In I-MAPF terms, $\mathcal{A}(c)$ constitutes the event signal $\mathcal{E}_{ij}(t)$; the alert tuple $(a_{c_k}, t_{j-r}, \Delta_c)$ encodes $|\Delta_c|$ IUs (each a (v, s, t) fact), keeping $\mathcal{O}_j = \emptyset$ at this stage. Once an alert is issued to an agent, the selected agent is prompted to perform a constrained replanning, following collision set Δ_c, of its trajectory over a rollout window $\{t_{j-r}^{c_k}, \ldots, t_j^{c_k}, \ldots, \tau_{c_k}\}$, where r is the rewind window.

c) S4: Replanning: Upon receiving a collision alert for conflict time t_j and rewind parameter r, the selected agent a_{c_k} conceptually decomposes its original trajectory $\rho_{c_k} = (v_0^{c_k}, \ldots, v_{\tau_{c_k}}^{c_k})$ into a fixed prefix, a replanning window, and an untouched suffix. We denote the fixed prefix up to just before the rewind step as

$$\rho_{c_k|t_{j-r-1}} = (v_0^{c_k}, \ldots, v_{t_{j-r-1}}^{c_k}). \tag{5}$$

All S4 operators then construct a new path segment $\rho'_{c_k}|\{t_{j-r}, ..., t_{j+r}\}$ that replaces the original states on the interval $\{t_{j-r}, ..., t_{j+r}\}$.

(S4.0) Yield-based local coordination. As a first step, the controller attempts a lightweight yield maneuver. If the conflict set A_c contains an agent that is already parked at its start or is waiting at its goal at time t_j, we mark this agent as *anchored* and select it as the replanning agent a_{c_k}; otherwise, S3 selects the replanning agent $a_{c_k} \in A_c$ using a heuristic policy (e.g., farthest from goal). In both cases, the remaining agents in A_c are treated as protected and keep their current plans. Starting from its rewind state $v_{t_{j-r}}^{c_k}$, a_{c_k} then executes a short-horizon yield: it moves to a nearby parking cell, waits while the protected agent(s) traverse the conflict region, and then returns to a state from which it can resume progress toward g_{c_k}. The parking cell is chosen by a local search that ensures the resulting detour is free of vertex and edge conflicts with all other agents over the considered window. Formally, S4.0 expands $\mathcal{O}_j$ by one local cell occupancy fact (1 IU for every occupancy check). If no such parking cell exists, or if the yield trajectory does not eliminate the conflict when re-simulated by the controller, the plan is abandoned, and the system falls back to try the next strategy.

For static obstacle avoidance *(S4.1)*, the constraint set, Δ_c in Eq. 6, is fixed by the alert and encodes the static cells that must be avoided. The "bounded" aspect comes from limiting how much of the trajectory is recomputed in time: we select a sub-initial point $v_{t_{j-r}}^{c_k}$ and a sub-goal (either the true goal g_{c_k} or $v_{t_{j+r}}^{c_k}$), and only replan this suffix under the fixed constraint. The planner π_θ here is A*, constrained to avoid Δ_c. Within the I-MAPF formulation, $\mathcal{E}_{ij}$ is extended by $|\Delta_c|$ static (v, s, t) IUs while $\mathcal{O}_j$ remains empty. The segment of the new path is generated as:

$$\rho'_{c_k} = \pi_\theta(v_{t_{j-r}}^{c_k}, v_{t_{j+r}}^{c_k} \mid v_{t_i}^{c_k} \notin \Delta_c, \forall t \in \{t_{j-r}, ..., t_{j+r}\}). \tag{6}$$

If bounded static replanning still fails to resolve the conflict, the controller may share richer information and trigger dynamic obstacle avoidance *(S4.2)*, where the replanning

must account for the predicted movements of other agents involved in the collision. Here, the RL-based policy π_θ is used. Each agent observes a tensor $\mathbf{s} \in \mathbb{R}^{H\times W\times 4}$ with channels: ObstacleMap (static obstacles), AgentMap (own position), GoalMap (goal), and AlertMask (conflict cells from $\mathcal{E}_{ij}$, updated online), augmented with a unit vector and distance to goal. The discrete action space is $\mathcal{A} = \{0, 1, 2, 3, 4\}$ (four cardinal moves + WAIT). The reward encourages efficient conflict-free navigation: $+20$ on reaching the goal, -3 for obstacle collision, -2 for timeout, -0.02 per timestep, -0.1 for waiting, and -0.05 near a dynamic obstacle [6]. This encodes r time-indexed (v, s, t) facts (r IUs) from $\mathcal{O}_j$ via $\mathcal{E}_{ij}$. In this case, the RL policy is conditioned on the sub-paths of the conflicting agents as illustrated by the constraints in Eq. 7

$$\rho'_{c_k} = \pi_\theta(v^{c_k}_{t_{j-r}}, v^{c_k}_{t_{j+r}} \mid v^{c_k}_t \neq v^{c_l}_t, v^{c_l}_t \in \rho_{c_l}, \\ \forall l \neq k, a_{c_l} \in A_c, \forall t \in \{t_{j-r}, ..., t_{j+r}\}). \quad (7)$$

The information guiding the replanning, whether it constitutes minimal details such as static obstacle constraints (Eq. 6) or more detailed sub-path information about colliding agents' paths treated as dynamic obstacles (Eq. 7), is integrated into the planners' decision-making process to guide it towards a conflict-free solution.

In all cases, the agent's new complete trajectory $\rho^{new}_{c_k}$ is formed by concatenating the initial segment with the newly planned one, given by

$$\rho^{new}_{c_k} = \rho_{c_k|t_{j-r-1}} \parallel \rho'_{c_k} \parallel \rho_{c_k|t_{j+r+1}}. \quad (8)$$

Each updated trajectory is submitted to the central controller. The iterative cycle of detection (S2), control (S3), and decentralized replanning (S4) continues until no further conflicts can be resolved by single-agent strategies or a global termination condition is reached.

In a small fraction of cases, even dynamic single-agent replanning leaves a persistent local conflict among a tightly coupled subset of agents. To address these situations, the controller invokes a local joint planner *(S4.3)* over a subset $A^J_c \subseteq A_c$ and a bounded time window around the conflict. Let $\{t_{j-r_J}, ..., t_{j+r_J}\}$ denote a short horizon centered at t_j. For each agent $a_k \in A^J_c$, we treat $v^k_{t_{j-r_J}}$ as its joint-planning start state and $v^k_{t_{j+r_J}}$ as a temporary sub-goal, and we solve a small-horizon problem in the joint configuration space of A^J_c over this interval while treating all other agents as fixed reservations. The resulting joint segment $\{\rho^{\text{joint}}_k\}_{a_k\in A^J_c}$ is spliced into the original trajectories between t_{j-r_J} and t_{j+r_J}, analogous to Eq. 8. This joint search over $\mathcal{O}_j = A^J_c$ reveals $|A^J_c|\cdot r$ IUs, the maximum IU expenditure across all tiers.

If none of the above strategies can successfully remove a conflict, the corresponding agent is temporarily *deferred*: it is parked (typically at its start) and excluded from further replanning during the current phase. Once the non-deferred agents have reached a conflict-free configuration up to the current makespan, a second phase is initiated in which the deferred agents are reintroduced and planned using the same tiered strategies starting from the current makespan. This two-phase design specifically targets deadlock-like situations by decoupling stubborn local clusters and allowing them to be resolved after the rest of the system has stabilized.

By varying what constraints are set for replanning, we obtain a family of tiered replanning strategies (S4.1–S4.4) that trade off information usage against solution completeness. (S4.1) Yield-based local coordination performs a short local detour using only immediate (typically 1 to 3 hop distance) cell occupancy information, (v, s, t) facts (1 IU). (S4.2) Static replanning recomputes a bounded path segment while treating a fixed set of conflict cells, Δ_c represented as (v, s, t) facts ($|\Delta_c|$ IUs), as forbidden. (S4.3) Dynamic replanning incorporates short prefixes of other agents' trajectories over a horizon r, r (v, s, t) facts (r IUs). (S4.4) Local joint planning replans a tightly coupled subset of agents $A^J_c \subseteq A_c$ over window r, requiring (v, s, t) facts totaling $|A^J_c|\cdot r$ IUs. Together, these tiers provide a controlled way to gradually increase the amount of information used for coordination only when simpler, lower inter-agent information-based repairs fail. By attempting cheaper repairs first, HI–MAPF minimizes the average per-conflict communication and computation cost.

IV. Experimental Validation

A. Experimental Setup

In this section, we describe the training procedure for the per-agent navigation policy (S1), our hardware deployment on physical robots, and the simulation benchmarks used to evaluate scalability.

a) Training Procedure: We train a Double DQN [21] policy $Q_\theta(s, a)$ with prioritized experience replay (PER, buffer size 10^6, batch size 128) and ε-greedy exploration (ε : $1.0 \rightarrow 0.01$, decay 0.999) to navigate toward a goal while avoiding static and dynamic obstacles. Dynamic obstacles follow precomputed trajectories with hidden goals, simulating online collision alerts from S3. Training runs for $30{,}000$ episodes on an 11×11 maze ($T_{\max} = 50$) with a curriculum that progressively increases static obstacle density ($\rho_s : 0.10 \rightarrow 0.30$) and dynamic obstacle count (0 to 4). A validity mask $m_t \in \{0, 1\}^{|\mathcal{A}|}$ constrains action selection to feasible moves [6]. The network is optimized with Adam ($\alpha = 3 \times 10^{-4}$, $\gamma = 0.97$) and a target network updated every 300 steps, minimizing the PER-weighted Bellman loss $L(\theta) = \mathbb{E}_{i\sim p(i)}[w_i\, \delta_i^2]$, where $p(i) \propto |\delta_i|^\alpha$ and w_i are importance-sampling weights.

b) Hardware Experiments: TurtleBot4: We deploy HI–MAPF and baseline algorithms on five TurtleBot4 robots in a 6×6 indoor grid to validate deployment feasibility under realistic constraints. For HI–MAPF, the centralized coordinator module (S3) runs on a host PC, while each robot executes its own local planning (S1) and replanning (S4) under isolated ROS 2 namespaces. For learning-based baselines, an orchestrator broadcasts positions and collects actions via ROS 2 topics, with each robot running its respective neural policy locally. A ROS 2 Discovery Server ensures stable communication. All robots communicate through a shared synchronization topic (`/sync_barrier`); other

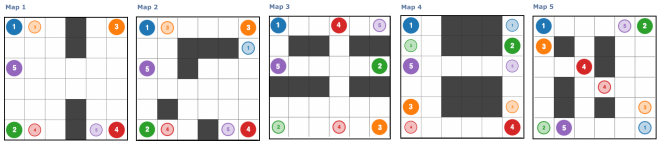


Fig. 2. Five hardware test instances on a 6×6 grid with five agents each. Top row: start positions; bottom row: goal positions. Gray cells are static obstacles. Solid colored circles represent agents' positions.

ROS interfaces remain fully namespace-isolated. No onboard perception (LiDAR, RGB-D, or OAK-D camera) is used; robots follow their assigned plans using open-loop odometry control. We evaluate five MAPF problem instances (Figure 2), each repeated five times.

c) Baseline Deployment on TurtleBot4: To establish a fair comparison, we deploy SCRIMP, DCC, EPH, and MAPF-GPT on the same five TurtleBot4 robots in a distributed configuration, implementing each algorithm's coordination architecture. For **SCRIMP**, each robot runs its trained policy with a 3x3 FOV constructed from broadcast positions. Since TurtleBot4 lacks native grid-based FOV sensing, each robot's position is broadcasted via ROS 2 topics and agents construct local FOV observations from these messages. Critically, SCRIMP requires each robot to transmit a 512-dimensional hidden-state message to all other robots at every timestep through its transformer-based communication module, representing inter-agent message passing over the network. For **DCC** and **EPH**, which use attention-based communication internally, each robot receives broadcast positions and runs neural inference for *all* agents locally, computing the same attention-based coordination that would occur if hidden states were exchanged. Since all robots receive identical inputs, the attention produces identical results on each robot without requiring actual inter-robot message transmission. For MAPF-GPT, each robot similarly receives broadcast positions and constructs a local 11×11 FOV, but runs a pretrained GPT model (2M) for inference independently, requiring no inter-agent communication or hidden-state exchange. All inter-agent communication is logged, and IU is calculated by counting actual information exchanged during planning phase, ensuring that IU measurements reflect the true information overhead each method requires in deployment.

d) Plan Execution: MAPF algorithms produce discrete action sequences on a grid, but physical robots require motion commands. We develop a plan executor that bridges this gap for all algorithms deployed on TurtleBot4. The executor implements four key components: (i) *Action mapping*: Discrete MAPF actions are converted to target headings and $0.45\,\text{m}$ forward motions on the physical grid. (ii) *Following conflict detection*: A pre-computation step identifies timesteps where agent B moves into a cell being simultaneously vacated by agent A. Such conflicts are safe in discrete-time MAPF but create collision risk in real-world execution; the executor identifies and inserts configurable delays at flagged steps. (iii) *Closed-loop control*: Each step consists of a rotation phase (closed-loop angular control with creep-speed corrections for fine alignment) followed by a forward motion phase (odometry-based distance tracking with undershoot compensation). (iv) *Lockstep synchronization*: Each step has separate turn-phase and move-phase synchronization barriers via a shared `/sync_barrier` topic; all robots must acknowledge completion of each phase before the system advances, enforcing collision-free execution across the team.

e) Simulation: Scalability Evaluation: To validate HI–MAPF's scalability beyond the 5-robot hardware testbed, we evaluate on standard benchmark maps from the *Moving AI Lab* dataset [22], including random grids (32×32, 64×64 with 20% obstacles) and structured maps (`den312d`, an office-like layout with bottleneck corridors; and `warehouse`, with long parallel aisles and cross-passages). Agent counts range from 10–128 on random maps and 8–128 on structured maps. For each configuration, we generate 20 random instances. We compare HI–MAPF against representative search- and learning-based baselines: CBS, DCC, SCRIMP, MAPF-GPT (2M parameter model), and EPH. HI–MAPF is given a 600 seconds limit per instance (Python implementation), CBS is capped at 300 seconds, and learning-based solvers are run with step limits (128 for random maps, 256 for `den312d`/`warehouse`) and a 5-minute wall-clock bound. Section IV-B presents comparative results.

f) Performance Criteria: We evaluate using three metrics. *Success Rate (SR)* is the proportion of instances solved within defined time limits. *Makespan (MS)* measures the time until the last agent reaches its goal. *IU*, as defined in Section III-B, quantify the total cell-level data exchanged during planning and execution. We omit IU for CBS as it uses full global information. For hardware experiments, we additionally report *Communication Frequency (Comm. Freq.), the average total messages exchanged during planning;* Execution Messages (Exec. Msgs)*, the total lockstep synchronization messages during physical execution; and* Execution Time*, the wall-clock duration on physical robots.*

B. Results and Discussion

We present empirical results demonstrating that HI–MAPF enables practical multi-robot deployment with minimal information exchange, validated first on physical hardware and then at scale in simulation.

1) Hardware Deployment Results: We evaluate HI–MAPF, DCC, EPH, MAPF-GPT, and SCRIMP on five problem instances with five TurtleBot4 robots in a distributed deployment configuration. Table I reports planning and execution results, and Figure 3 provides a multi-axis comparison across key deployment metrics.

All five algorithms achieve 100% success rate across the five problem instances (Table I). HI–MAPF achieves the lowest average makespan (12.2) and the lowest total IU (199), followed by DCC (596, $3.0\times$) and EPH (623, $3.1\times$). MAPF-GPT achieves a competitive makespan (13.6) and IU (588, $3.0\times$), with FOV sensing as its dominant IU source

TABLE I

HARDWARE RESULTS AVERAGED ACROSS FIVE PROBLEM INSTANCES WITH 5 TURTLEBOT4 ROBOTS. HI–MAPF ACHIEVES THE LOWEST DEPLOYMENT OVERHEAD FOR BOTH PLANNING AND EXECUTION.

	SCRIMP	DCC	MAPF-GPT	EPH	HI–MAPF
Planning Results					
Success Rate (%)	**100**	**100**	**100**	**100**	**100**
Avg. Makespan	16.4	19.4	13.6	21.8	**12.2**
Avg. Total IU	42,413	596	588	623	**199**
Avg. IU vs HI–MAPF	213×	3.0×	3.0×	3.1×	**1×**
Avg. Comm. Freq.	180	116	82	131	**10**
Execution Results					
Avg. Exec. Time (s)	211.24	234.98	**176.20**	234.94	181.46
Avg. Exec. Msgs	169	199	141	223	**127**

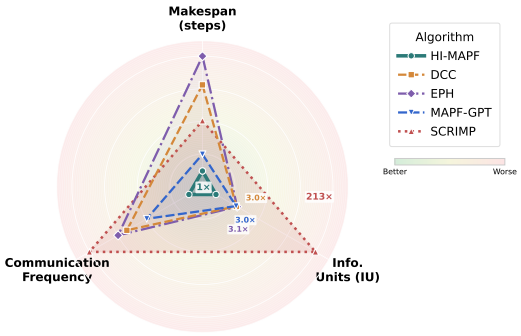


Fig. 3. Multi-axis comparison of hardware deployment results across makespan (steps), information units (IU), and communication frequency. HI–MAPF achieves the most balanced profile (smallest area). Baselines trade off one dimension for another, to acheive coordination capability.

(∼41%), followed by position broadcasts (∼23%) and the map (∼24%); it requires no inter-agent communication. For DCC and EPH, IU is dominated by per-timestep position broadcasts and action transmissions (∼49 to 53%), with FOV sensing contributing ∼20 to 22%. SCRIMP incurs 213× more IU than HI–MAPF (42,413), with 99% of its IU attributed to the 512-dimensional inter-agent messages transmitted through its transformer-based communication module at every timestep. In contrast, HI–MAPF's IU is dominated by the initial map broadcast (71%), with the remaining IU arising from conflict-triggered constraints during replanning. These results demonstrate that HI–MAPF achieves competitive coordination while requiring substantially less inter-agent information exchange than the learning-based baselines. For physical execution, all algorithms use the same lock-step synchronization protocol, where each plan step requires a turn and move phase with synchronization acknowledgements from all robots after each phase. Execution time and total synchronization messages therefore scale linearly with plan length. MAPF-GPT and HI-MAPF achieves the lowest execution times, while HI–MAPF's shorter MS yield the fewest synchronization messages (127) among all algorithms.

2) Simulation: Scalability Validation: Having validated HI–MAPF and baselines on physical robots, we assess scalability across larger maps and agent counts. Table II summarizes the comparative performance of HI–MAPF against the considered baselines.

a) Information Efficiency: HI–MAPF achieves high success rates with *minimal, event-triggered information sharing*. Across all four benchmarks (Table II), learning-based baselines require 2×–510× more IU than HI–MAPF, with the gap widening on larger maps where SCRIMP and DCC often fail to scale. MAPF-GPT-2M achieves strong success rates and competitive makespans but incurs the highest IU overhead among all baselines, as it requires 11x11 FOV observations compared 3x3 in SCRIMP. On structured maps, EPH maintains strong SR through its ensembled hybrid policies but still consumes 3.6×–11× more information than HI–MAPF. These results identify *targeted, conflict-triggered sharing of short path fragments* as the minimal information sufficient for effective MAPF.

b) Scalability: CBS relies on full global information and fails to scale under imposed time limits. Learning-based methods show strong performance in some dense scenarios but incur substantially higher information loads. HI–MAPF scales favorably across both random and structured maps, maintaining high SR with competitive makespans while keeping information exchange sparse and bounded. Across all benchmarks, S4.1 alone resolves 92 to 99% of conflicts, S4.2 handles 1 to 8%, and S4.3/S4.4 are invoked in fewer than 0.3% of cases. These simulation results are consistent with our hardware findings (Table I), supporting HI–MAPF's suitability for resource-constrained robotic deployment.

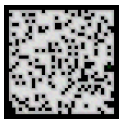
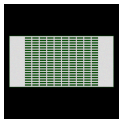
V. DISCUSSION AND CONCLUSIONS

We introduce the I-MAPF formulation and propose HI–MAPF, a hybrid algorithm that reduces inter-agent information exchange while maintaining high coordination performance under tight sensing and computation requirements. Using *IU* as a method-agnostic deployment-cost metric, our results on physical TurtleBot4 robots and in simulation at scale show that effective MAPF coordination can rely on targeted, conflict-triggered sharing of short path fragments rather than continuous observation. As shown in Figure 3, HI–MAPF achieves the most balanced profile across makespan, IU, and communication frequency without onboard perception hardware, while baselines such as EPH [16] trade higher IU and makespan for coordination capability.

Limitations. HI–MAPF may face challenges in extremely dense settings where conflicts are frequent, and its tiered repair strategy is currently heuristic based. Our evaluation focuses on communication- and information-focused baselines (DCC, EPH, SCRIMP, MAPF-GPT) and does not include comparisons with recent algorithms that do not target information efficiency as a design objective. As future work, we aim to explore learning-based mechanisms that automatically adapt the tier-selection policy, investigate reinforcement learning under explicit information budgets, and extend hardware validation to larger robot teams and more complex environments.

TABLE II

PERFORMANCE OF SEARCH-BASED (CBS), LEARNING-BASED (SCRIMP, DCC, MAPF-GPT-2M), AND HYBRID (EPH, HI–MAPF) MAPF SOLVERS. m, SR, MS, IU REPRESENT THE NUMBER OF AGENTS, SUCCESS RATE, MAKESPAN, AND INFORMATION UNITS RESPECTIVELY. VALUES ARE AVERAGED WITHIN AGENT COUNT GROUPS. IU FOR BASELINE SOLVERS IS SHOWN AS AN N× MULTIPLIER RELATIVE TO HI–MAPF (1×). LOWER MS↓ AND IU↓, HIGHER SR↑ ARE BETTER. **BOLD** IU VALUES HIGHLIGHT HI–MAPF AS THE INFORMATION-EFFICIENCY BASELINE.

Map		m	Search-based		Learning-based									Hybrid					
			CBS		SCRIMP			DCC			MAPF-GPT-2M			EPH			HI–MAPF		
			SR↑	MS↓	SR↑	MS↓	IU↓	SR↑	MS↓	IU↓	SR↑	MS↓	IU↓	SR↑	MS↓	IU↓	SR↑	MS↓	IU↓
	random 32×32–20	10-30	100	47.6	100	50.3	19x	100	55.5	11x	100	52.3	96x	98	57.4	9.0x	100	54.0	**29.6 (1×)**
		40-60	65	85.3	95	62.1	12x	85	82.1	23x	100	59.7	109x	97	77.4	22x	100	98.9	**161.5 (1×)**
		70-90	0	–	77	83.8	2.6x	50	111.3	10x	100	70.4	28x	53	114.5	12x	98	73.4	**1838.1 (1×)**
		100	0	–	60	97.2	1.9x	40	117.5	10x	100	77.0	23x	50	121.0	11x	85	76.5	**3693.9 (1×)**
		128	0	–	70	96.6	–	0	127.0	–	90	94.1	–	5	127.5	–	0	–	–
	random 64×64–20	10-30	85	106.3	23	121.9	137x	97	105.9	7.9x	100	107.2	170x	98	104.1	6.5x	100	113.9	**10.2 (1×)**
		40-60	42	130.0	0	128.0	–	83	116.8	9.5x	95	115.2	115x	98	114.0	7.3x	100	125.6	**88.5 (1×)**
		70-90	0	–	0	128.0	–	43	125.0	14x	88	120.3	127x	85	121.1	12x	98	129.1	**212.9 (1×)**
		100	0	–	0	128.0	–	25	127.2	13x	75	123.8	89x	45	125.3	11x	90	130.1	**472.9 (1×)**
		128	0	–	0	128.0	–	0	128.0	–	55	126.2	132x	15	127.7	23x	5	132.0	**530.0 (1×)**
	den312d 65×81	8	100	97.2	20	226.2	317x	100	114.2	6.2x	100	114.0	82x	100	111.9	3.6x	100	115.3	**3.6 (1×)**
		16	85	128.7	0	256.0	–	100	130.7	8.8x	100	120.9	67x	100	122.2	5.9x	100	124.0	**21.8 (1×)**
		32	20	148.0	0	256.0	–	85	163.5	11x	100	126.4	61x	100	127.0	6.0x	100	136.7	**105.2 (1×)**
		64	0	–	0	256.0	–	90	190.7	11x	100	137.8	36x	100	146.1	8.8x	100	153.1	**759.9 (1×)**
		128	0	–	0	256.0	–	5	254.8	–	35	234.0	–	90	207.7	–	0	–	–
	warehouse 161×63	8	100	175.7	0	256.0	–	100	191.8	5.8x	100	193.6	143x	100	191.6	2.3x	100	192.8	**0.8 (1×)**
		16	100	196.2	0	256.0	–	100	213.3	16x	100	214.0	510x	100	212.9	11x	100	215.1	**1.0 (1×)**
		32	15	221.3	0	256.0	–	100	219.2	8.3x	100	221.8	215x	100	218.1	5.4x	100	220.3	**10.7 (1×)**
		64	0	–	0	256.0	–	100	227.6	15x	100	229.3	400x	100	227.8	11x	100	227.6	**23.6 (1×)**
		128	0	–	0	256.0	–	95	236.8	14x	95	237.2	277x	90	239.1	11x	100	234.7	**139.1 (1×)**

REFERENCES

[1] J. Ren, E. Eric, T. K. S. Kumar, S. Koenig, and N. Ayanian, "Empirical hardness in multi-agent pathfinding: Research challenges and opportunities," in *Blue Sky paper at 24th International Conference on Autonomous Agents and Multiagent Systems*, 2025.

[2] G. Sartoretti *et al.*, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2559–2566, 2019.

[3] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[4] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, "Icbs: The improved conflict-based search algorithm for multi-agent pathfinding," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 6, no. 1, 2015, pp. 223–225.

[5] G. Wagner and H. Choset, "M*: A complete multirobot path planning algorithm with performance bounds," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 3260–3267.

[6] M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti, "Primal2: Pathfinding via reinforcement and imitation multi-agent learning—lifelong," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2666–2673, 2021.

[7] A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, and A. Panov, "Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 17 541–17 549.

[8] Y. Wang, B. Xiang, S. Huang, and G. Sartoretti, "Scrimp: Scalable communication for reinforcement-and imitation-learning-based multi-agent pathfinding," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 9301–9308.

[9] Z. Ma, Y. Luo, and H. Ma, "Distributed heuristic multi-agent path finding with communication," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8699–8705.

[10] Z. Ma, Y. Luo, and J. Pan, "Learning selective communication for multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1455–1462, 2021.

[11] Q. Lin and H. Ma, "Sacha: Soft actor-critic with heuristic-based attention for partially observable multi-agent path finding," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5100–5107, 2023.

[12] C. He, T. Duhan, P. Tulsyan, P. Kim, and G. Sartoretti, "Social behavior as a key to learning-based multi-agent pathfinding dilemmas," *Artificial Intelligence*, p. 104397, 2025.

[13] C. He, T. Yang, T. Duhan, Y. Wang, and G. Sartoretti, "Alpha: Attention-based long-horizon pathfinding in highly-structured areas," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 576–14 582.

[14] S. Liao, W. Xia, Y. Cao, W. Dai, C. He, W. Wu, and G. Sartoretti, "Sigma: Sheaf-informed geometric multi-agent pathfinding," *arXiv preprint arXiv:2502.06440*, 2025.

[15] A. Andreychuk, K. Yakovlev, A. Panov, and A. Skrynnik, "Mapf-gpt: Imitation learning for multi-agent pathfinding at scale," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 22, 2025, pp. 23 126–23 134.

[16] H. Tang, F. Berto, and J. Park, "Ensembling prioritized hybrid policies for multi-agent pathfinding," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 8047–8054.

[17] J. Li, Z. Chen, D. Harabor, P. J. Stuckey, and S. Koenig, "Mapf-lns2: Fast repairing for multi-agent path finding via large neighborhood search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 10 256–10 265.

[18] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[19] K. Okumura, "Lacam: Search-based algorithm for quick multi-agent pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 11 655–11 662.

[20] Y. Wang, T. Duhan, J. Li, and G. A. Sartoretti, "Lns2+rl: Combining multi-agent reinforcement learning with large neighborhood search in multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[21] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[22] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski,

and R. Bartak, “Multi-agent pathfinding: Definitions, variants, and benchmarks,” *Symposium on Combinatorial Search (SoCS)*, pp. 151–158, 2019.