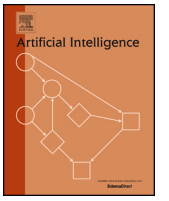
Artificial Intelligence 348 (2025) 104397

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Social behavior as a key to learning-based multi-agent pathfinding dilemmas

Chengyang He, Tanishq Duhan, Parth Tulsyan, Patrick Kim, Guillaume Sartoretti *

Department of Mechanical Engineering, College of Design and Engineering, National University of Singapore, 21 Lower Kent Ridge Rd, 117575, Singapore

ARTICLE INFO

Keywords:
Multi-agent pathfinding
Parameter sharing
Symmetry dilemmas
Social value orientation

ABSTRACT

The Multi-agent Path Finding (MAPF) problem involves finding collision-free paths for a team of agents in a known, static environment, with important applications in warehouse automation, logistics, or last-mile delivery. To meet the needs of these large-scale applications, current learning-based methods often deploy the same fully trained, decentralized network to all agents to improve scalability. However, such parameter sharing typically results in homogeneous behaviors among agents, which may prevent agents from breaking ties around symmetric conflict (e.g., bottlenecks) and might lead to live-/deadlocks. In this paper, we propose SYLPH, a novel learning-based MAPF framework aimed to mitigate the adverse effects of homogeneity by allowing agents to learn and dynamically select different social behaviors (akin to individual, dynamic roles), without affecting the scalability offered by parameter sharing. Specifically, SYLPH offers a novel hierarchical mechanism by introducing Social Value Orientation (SVO) as a temporally extended latent variable that plays a central role in both policy generation and reward assignment. To support this hierarchical decision-making process, we introduce Social-aware Multi-Policy PPO (SMP3O), a reinforcement learning method that ensures stable and effective training through a mechanism for the cross-utilization of advantages. Moreover, we design an SVO-based learning tie-breaking algorithm, allowing agents to proactively avoid collisions, rather than relying solely on post-processing techniques. As a result of this hierarchical decision-making and exchange of social preferences, SYLPH endows agents with the ability to reason about the MAPF task through more latent spaces and nuanced contexts, leading to varied responses that can help break ties around symmetric conflicts. Our comparative experiments show that SYLPH achieves state-of-the-art performance, surpassing other learning-based MAPF planners in random, room-like, and maze-like maps, while our ablation studies demonstrate the advantages of each component in SYLPH. We finally experimentally validate our trained policies on hardware in three types of maps, showing how SYLPH allows agents to find high-quality paths under real-life conditions. Our code and videos are available at: marmotlab.github.io/mapf_sylph.

* Corresponding author.
E-mail address: guillaume.sartoretti@nus.edu.sg (G. Sartoretti).

https://doi.org/10.1016/j.artint.2025.104397
Received 5 August 2024; Received in revised form 23 July 2025; Accepted 24 July 2025

Available online 30 July 2025

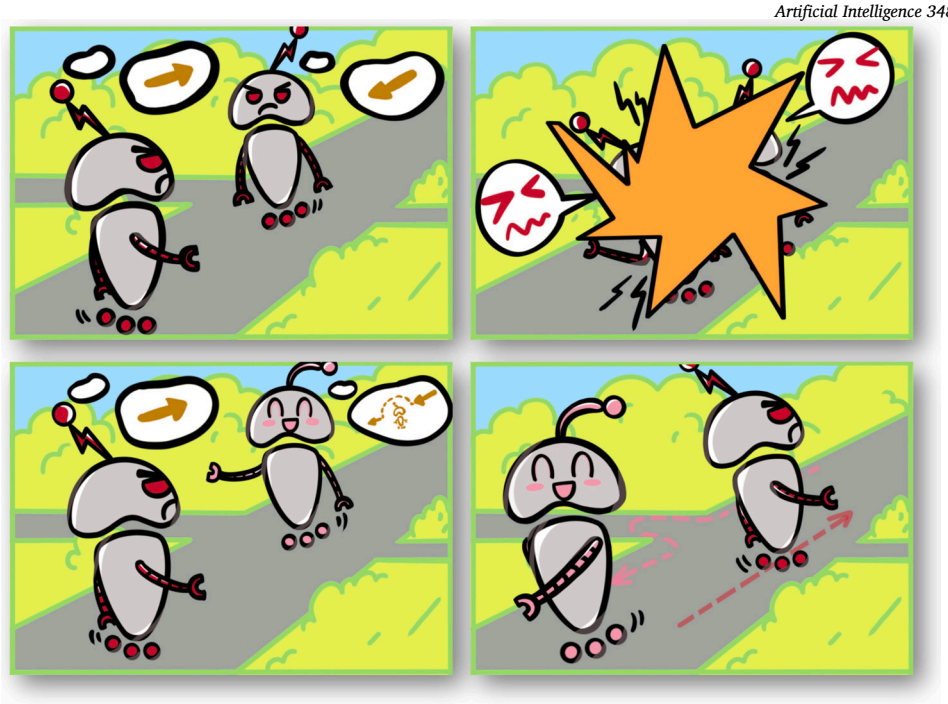


Fig. 1. A simple example illustrates the difference between a completely selfish team and a team with diverse social roles. The two figures above show that when facing a symmetric challenge, a team of selfish agents falls into a social dilemma. In contrast, agents with different SVOs can more easily achieve cooperation by breaking the homogeneity of their behavior patterns. They benefit from a combination of individualism and pro-socialism within the team, as shown in the two figures below.

1. Introduction

Multi-Agent Path Finding (MAPF) involves devising collision-free paths for multiple agents within a known and static space, guiding them from their current positions to their respective goals [1]. MAPF is commonly used in warehouse automation [2,3], air traffic control [4], autonomous driving [5], and video game AI [6]. While the objectives for coordinating the behaviors of all agents may differ among these scenarios, such as minimizing makespan, ensuring safety margins, or optimizing resource usage, a common feature is the need to deploy and coordinate always-increasing numbers of agents.

To meet the needs of such large-scale applications, the community has increasingly turned to learning-based methods [7–9] that leverage deep reinforcement learning (DRL) and advanced neural network architectures. These decentralized, reactive MAPF planners offer improved scalability, addressing the limitations of traditional algorithms that struggle with the curse of dimensionality as team sizes increase [10,11], though they often result in suboptimal solutions. To enhance scalability, learning-based MAPF algorithms usually adopt the Independent Policy Learning (IPL) paradigm [12], where agents consider/observe each other as dynamic features of the environment, greatly reducing the complexity of learning and increasing the robustness of policies. The learning-based MAPF methods benefit from good scalability also partly due to decentralized decision-making driven by *parameter sharing* [13,3,14], in which experiences collected by multiple agents are combined to train a single neural network during the training process, and the fully trained network is then deployed to all agents for execution. However, independent learning with parameter sharing tends to homogenize the behavior of the agents, resulting in all agents exhibiting similar social preferences, often individualistic due to their need to maximize individual rewards and complete individual tasks. In highly structured scenarios, such as bottlenecks and narrow corridors, homogeneous behaviors may cause agents to fall into symmetric social dilemmas [15], which can lead to live- or deadlocks, as illustrated in Fig. 1. These *social dilemmas* arise from symmetries in the environment / agents' states and are exacerbated by conflicts between individual interests, where none of the agents involved can obtain higher rewards unless compromises are made by one of them. Another limitation of this training approach is that the independent nature of learning results in agents that cannot easily encourage/exhibit coordinated maneuvers, which limits their performance in dense scenarios. A viable solution for agents to address these two issues is by learning social behaviors, specifically through reasoning about and balancing short-term self-interests with long-term team benefits. Learning social behavior equips agents with varying levels of prosociality, breaking homogeneity and by more tightly coupling agents directly in reward space. This approach helps with coordinated maneuvers, essential for resolving social dilemmas in large-scale MAPF instances.

To these ends, this paper develops a novel learning-based hierarchical MAPF framework, SYLPH (SociallY-aware muLti-agent PatHfinding), which helps agents mitigate the adverse effects of homogeneity on scalability by allowing them to learn dynamic roles while following the parameter-sharing paradigm. At the core of SYLPH is the introduction of Social Value Orientation (SVO), a

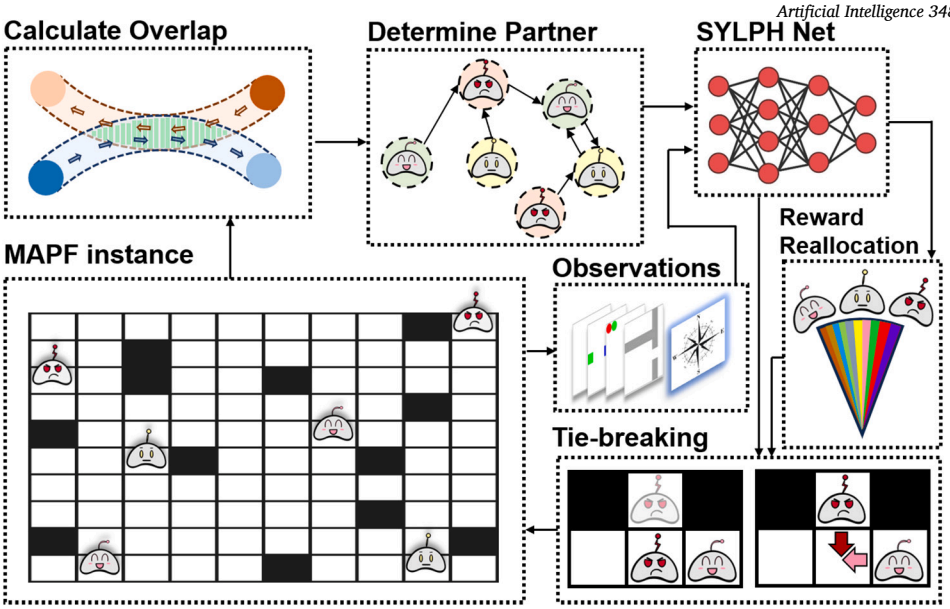


Fig. 2. The key components and overall architecture of SYLPH. By introducing social preference into the MAPF framework as a temporary extension variable, the agent is equipped with social behavior to better cope with social dilemmas such as symmetry problems and blocking problems.

concept borrowed from social psychology, which captures agents' latent social preferences such as altruism or selfishness. We embed SVO into the agents' high-level decision space to reflect their cooperative intent in a temporally extended variable. Unlike prior works where SVO is used to predict uncontrollable agents [16], we use it to guide the behavior of controllable agents through both policy and reward adaptation. This tight coupling between the agent's high-level social preference (SVO) and both its policy and reward function enables behavior differentiation not just through input variance, but through diverging training trajectories. This fundamentally distinguishes our approach from previous communication-based methods. Therefore, this design enables agents to dynamically adjust their social behavior based on local observations and the social context communicated via SVO.

To implement this SVO-based reasoning process, we build a hierarchical policy architecture (Fig. 2) where the upper level encodes the latent social preference (SVO) and the lower level executes primitive movements (cardinal directions and staying idle). Each agent first identifies its most influential global/local peer (referred to as its *partner*) by predicting potential conflicts and interactions, and then determines its own SVO based on local observations and the partner's SVO. This SVO then conditions a decentralized MAPF policy that not only takes SVO as input but also adapts the reward structure based on it, enabling both behavior differentiation and adaptive social learning. We introduce a novel reinforcement learning algorithm, SMP3O (Social-aware Multi-Policy PPO), which leverages a cross-utilization advantage mechanism to jointly train the action and SVO policies. Furthermore, we design an SVO-based tie-breaking mechanism that is tightly integrated into the reinforcement learning process, rather than being merely a post-processing [17,18]. By incorporating tie-breaking into learning, SYLPH enhances agents' inference abilities, effectively reducing collisions from model level rather than just resolving them after they occur. Together, these mechanisms enable SYLPH to reason about MAPF in a more latent and context-aware manner, producing diverse and adaptable responses to break symmetric conflicts, surpassing previous learning-based methods in both coordination and decision-making flexibility.

In addition, as a social psychology tool, SVO has semantic meaning and can clearly express the degree of altruism or selfishness of the agent, providing direct explainability for its decision-making process. Specifically, we introduce the interpretable variables (SVO) in the communication, decision-making, and execution process by: 1) Replace high-dimensional embeddings as messages for agent communication with human-interpretable SVO, which introduce the explainable symbols to drive decisions; 2) Utilizing a hierarchical decision-making structure as an explainable reasoning pipeline that clearly defines reasoning steps; 3) Implementing SVO-based tie-breaking to determine/learn priority in high-traffic areas, making conflict resolution more structured and explainable. Finally, to validate the advantages of SYLPH, we conduct a comprehensive evaluation of the proposed framework. We test MAPF configurations with various team sizes on three different types of maps: random maps, room-like maps, and maze maps. Additionally, we also present an ablation study to verify the effectiveness of each component of our framework. Finally, through real-robot experiments on three different types of maps, we demonstrate that the paths provided by SYLPH are executable under real-life conditions.

The rest of the paper is arranged as follows: Section 2 provides an overview of prior work, including traditional MAPF methods, learning-based MAPF methods, and research on social preferences. Section 3 introduces the MAPF problem formulation and the environment used in our study. Section 4 details the technical aspects of SYLPH, covering partner selection, SVO generation, and the tie-breaking policy. Section 5 provides an in-depth explanation of the neural network architecture. Section 6 presents the experimental results, while Section 7 concludes the paper.

2. Prior work

2.1. Traditional MAPF methods

The research community's exploration of MAPF originated with traditional methods. Traditional MAPF techniques are categorized by optimality into three types: optimal, bounded suboptimal, and unbounded suboptimal [19]. Optimal methods, by definition, enable multiple agents to achieve their goals with minimal overall cost. Theoretically, optimal methods are typically complete; they should offer a solution as long as a solution exists for the MAPF instance. Among the key approaches in MAPF, M* [20] and CBS [21] are recognized as influential optimal planners, with many subsequent methods are extension and improvements of them [10,22,11]. When M* does not detect conflicts between agents, the state space only expands by one cell at each timestep, following the optimal choices of all agents. For conflicting agents, M* evaluates combinations of their possible actions while attempting to balance these with the optimal actions of other agents. It does so by searching through the joint space of agents around collisions, and at worst can fall back onto exhaustive search for the whole team. CBS adopts a two-layer approach, where the upper layer uses conflict-based binary tree search and the lower layer uses the optimal single-agent planner A* to provide each agent with an optimal path based on conflict constraints.

Bounded suboptimal MAPF planners are designed to handle the increased computational load encountered by optimal solvers on expansive maps and with large-scale teams. These planners strike a balance between solution quality and computational efficiency by introducing a suboptimality factor, which can be tightened to make the planner approximate the optimal solution. For instance, inflated M* [23,24] achieves a relaxation of M* by altering the heuristic function of the A* algorithm. ECBS implements both levels of CBS as a focal search [25], reducing the number of collisions and accelerating the CBS search process. Building on ECBS, an advanced planner called EECBS [26] has been developed, which combines explicit estimation search at the high level with focal search at the low level. Thanks to its integration of multiple improvement techniques, EECBS maintains good performance even with large-scale teams.

To further pursue solver effectiveness rather than optimality, the community gradually began to study unbounded suboptimal solvers. This type of method's focus shifts from achieving the optimal solution to enhancing success rates and solution speeds without strict adherence to optimality, which is particularly useful in highly complex scenarios. The current state-of-the-art MAPF solvers also belong to this method.[1] MAPF-LNS [27] designs a new framework by combining small-scale high-quality solvers and large-scale low-quality solvers. First, any efficient MAPF algorithm is used to find the initial solution for the instance. From there, a Large Neighborhood Search (LNS) [28] is used to re-plan the subgroup of agents to improve the quality of the solution. In MAPF-LNS2⋆ [29], an opposite idea is adopted. Many collision-allowed paths are generated at first, and subgroups of conflicting paths are then continuously selected and updated within a limited time until they are collision-free. In addition to the search-based methods mentioned above, prioritized planning [30] has also played an important role in promoting MAPF research, inspiring advanced algorithms like PIBT and LaCAM. PIBT is a traditional one-step update method based on priority [31]. It generates a single step of the path for each agent at every timestep until the problem is solved or a preset maximum number of steps is reached. LaCAM⋆ [32], building upon PIBT, is a more advanced two-layer MAPF planner. At the higher level, it searches a sequence of configurations, where a configuration is a tuple of locations for all agents. The generation of these configurations is a low-level task. PIBT, as a low-level planner, can generate configurations that satisfy constraints extremely quickly.

The progression from optimal to unbounded suboptimal solvers in the MAPF community reflects a shift towards more flexible and scalable solutions that are capable of managing the increasing complexity of practical applications. This trend underscores a growing emphasis on algorithms that can deliver reasonable solutions within acceptable time frames, especially under the constraints of large-scale environments and agent populations.

2.2. Learning-based MAPF methods

Deep learning has been a promising tool to solve one-shot MAPF problems ever since PRIMAL was introduced [7]. Recent works have shown that agents can achieve better cooperation through communication [33], as it allows access to richer information. These communication learning-based methods employ their respective information aggregation mechanisms to enable effective knowledge exchange among agents, thereby enriching decision-making and fostering team cooperation. Previous MAPF communication learning frameworks typically rely on Graph Neural Networks (GNNs) [34–36] or Transformers [18] to facilitate message transmission and aggregation between agents. In these approaches, messages are represented as high-dimensional embeddings learned by neural networks [37]. These representations lack explicit meaning, making it challenging to interpret how agents interact and what factors drive their decision-making [38,39]. To address this issue, some studies have attempted to improve interpretability by selectively filtering message recipients [40,36], ensuring that only relevant agents receive communication. However, these methods do not resolve the fundamental problem: the messages themselves remain opaque and unexplainable, limiting our understanding of the system's internal reasoning. In contrast, SYLPH redefines communication in MAPF by using human-understandable variables (SVO) as messages. This shift ensures that communication reflects explicitly meaningful agent preferences, rather than abstract embeddings. Additionally, SYLPH's communication structure is inherently interpretable, as each agent selectively communicates with the predicted agents that

[1] We mark the state-of-the-art algorithms with ⋆.

has the great influence on its subsequent decisions. By combining these two mechanisms, SYLPH not only improves performance but also provides greater transparency and interpretability in multi-agent interactions.

Previous learning-based MAPF methods primarily rely on observations and messages for decision-making. While part of the research in the community has focused on optimizing communication mechanisms, another key effort has been to improve the encoding of observations [41]. Specifically, researchers have sought to design more effective encoding methods that allow agents to interpret a wider range of observations [42,3,43], leveraging the fact that most MAPF scenarios take place in known environments. Unlike previous methods, SYLPH integrates SVO into the decision-making process and introduces a hierarchical framework, ensuring that an agent's decisions are not only based on observations and messages, but also conditioned on its own learned social preferences from its previous experiences. This shift enables agents to adapt their behavior dynamically, rather than relying solely on static input representations. Unlike introducing heterogeneous input through communication, SYLPH treats SVO as a temporally extended variable. This approach not only makes the reasoning pipeline more transparent and interpretable but also expands the latent space and provides nuanced contextual awareness, allowing agents to break symmetry in structured environments. These improvements play a crucial role in resolving social dilemmas in highly symmetric scenarios.

Learning-based planners can sometimes encounter dead-/livelocks due to unforeseen circumstances. In such cases, tie-breaking mechanisms are essential for resolving conflicts and enhancing overall planning performance. Previous learning-based methods typically call pre-designed post-processing algorithms to resolve conflicts after detecting conflicts in the learned policies. However, many of these post-processing methods require re-planning [44] or additional model inference [18], resulting in significant computational overhead when frequently applied. Furthermore, these methods often act as external shieldings [17], while they can resolve conflicts, they do not inherently improve the decision-making capability of the model itself. In SYLPH, we address this limitation by incorporating SVO into the tie-breaking process. Instead of relying solely on post-processing, agents learn to determine priority in high-traffic areas based on SVO. Through reinforcement learning, agents develop the ability to adjust their actions dynamically when encountering agents with different SVO values, improving coordination and enhancing model performance intrinsically, rather than treating tie-breaking as a separate corrective step. As a result, during execution, SYLPH minimizes the need for post-processing interventions, significantly improving computational efficiency while maintaining robust conflict resolution.

While reinforcement learning (RL) remains the dominant approach in the learning-based MAPF community, imitation learning (IL) has recently begun to attract increasing attention. Given that traditional MAPF algorithms can already achieve near-optimal performance, even matching this level through imitation would represent a significant milestone for learning-based methods. Previous methods often combined IL with RL [7,18,41], but the contribution from the IL component was typically limited. Simple behavior cloning (BC), for instance, has shown only modest results [45], as models often struggle to capture the underlying logic of state-of-the-art handcrafted algorithms. As a result, such approaches generally scale poorly, often supporting fewer than 100 agents [34,35]. Some recent works have adopted new architectures, such as imitating collision shields which has more plain mechanisms [46], or building large-scale datasets to achieve foundation-model-level learning [47].

2.3. Social preference usage

In social psychology, Social Value Orientation (SVO) is a common metric for encapsulating an individual's social preferences and propensities [48], which serves as an indicator of a person's inclination to cooperate with fellow team members. Specifically, SVO quantifies the extent to which an individual prioritizes personal versus team benefits. It can be represented by the angle ϕ, as described in previous studies [49,16]. An agent is considered more egoistic when the ϕ value is closer to 0 (i.e., when its reward depends primarily on its own extrinsic rewards), and more altruistic as the ϕ value approaches 90 (i.e., when its reward depends mainly on the rewards of others). In robotics, Schwarting et al. pioneered the application of this concept in the field of autonomous driving [16]. They employed this metric as an intermediary variable, enhancing the accuracy of predictive models for human-driven vehicle trajectories. Subsequent research in autonomous driving has expanded upon this concept. Examples include investigations into social communication among multiple vehicles in the presence of adversaries [50], enabling agents to autonomously adapt their SVO preferences [51]. Furthermore, the field of video games has also seen significant research on SVO [52,53]. These studies show that in scenarios involving social dilemmas, the diversity of SVOs within the population is beneficial.

Previous SVO-based approaches typically set pre-allocated, immutable SVOs to agents. This arrangement is reasonable for tasks with a clear division of labor (e.g., Cleanup and Harvest [54]) or shared objectives (e.g., StarCraft II and Google Research Football [55]), where the necessity for diverse roles to collaboratively contribute to the team's objective is evident. In such scenarios, roles are often predetermined based on prior knowledge and maintained throughout the task. The absence of any role type can cause the entire system to malfunction, making the problem unsolvable. For example, in the CleanUp environment, successful task execution requires some agents to focus on cleaning while others handle resource collection [56]. If no agents assume the cleaning role, waste accumulates, making collection inefficient and ultimately leading to a substantial reduction in the final reward. In MAPF tasks, however, an agent may need to adopt various behavioral patterns depending on the situation in order to achieve more effective solutions. In other words, the allocation of roles is not individual-oriented but situation-oriented, and thus should remain dynamic throughout the task. Furthermore, while there is a common overarching goal, each agent also pursues individual objectives in MAPF. In order to balance self-interest and social benefit, there is a need for agents to learn a flexible SVO that can adapt to environmental changes. Drawing inspiration from skill learning [57–59], we propose viewing an agent's SVO as a temporally extended variable that depends on the previous timestep's SVO and the SVOs of other agents. The agent can then dynamically select an SVO based on observations in varying contexts, thereby influencing its lower-level action space decision-making.

In this paper, SYLPH adopts a flexible approach where agents can adaptively learn real-time SVO policies in response to their current environment, thereby moving away from the rigid and predefined allocation of social roles.

3. Problem statement

The Multi-Agent Path Finding (MAPF) problem exhibits numerous variants, including classic one-shot MAPF [29], MAPF with kinematic constraints [60], lifelong MAPF [61,13], prioritized MAPF [62], multi-agent pickup and delivery [31], and etc. This paper focuses on the classic one-shot MAPF problem. Typically, a classic MAPF instance is defined on an undirected simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \cdots\}$ represents the set of vertices that agents can occupy, and $\mathcal{E} = \{e_1, e_2, \cdots\}$ denotes the edges connecting these vertices. The presence of an edge between two vertices indicates that an agent can traverse between them within one time step. The graph $\mathcal{G}$ includes a set of agents $\mathcal{A} = \{a_1, a_2 \cdots a_n\}$, a pre-settled set of starting positions $\mathcal{S} = \{s_1, s_2, \cdots, s_n\} \in \mathcal{V}$, and a designated set of destinations/goals $\mathcal{D} = \{d_1, d_2 \cdots d_n\} \in \mathcal{V}$, where n represents the number of agents. In this context, time $t \in \mathbb{N}$ is treated as discrete, allowing an agent i to either move to an adjacent vertex $v_j \in \mathcal{N}_{\mathcal{G}}(v_i)$ or remain stationary at its current vertex v_i within a single timestep. The aim of the MAPF task is to generate collision-free paths for all agents from their respective initial occupied vertices s_i to their goal vertices d_i within the minimum possible number of timesteps. We consider a set of paths $\{\tau_i\}$ for agents $i = 1, 2, \cdots, n$, where each path τ_i is a sequence of vertices that agent a_i traverses over time $t = 0, 1, \cdots, T$, with T being the maximum timestep considered. All the paths must satisfy both **Condition 1:** $\forall i, j \in \{1, 2, \cdots, n\}, i \neq j, \forall t \in \{0, 1, \cdots, T\}, \tau_i(t) \neq \tau_j(t)$ (no two agents occupy the same vertex at any time), **Condition 2:** $\forall i, j \in \{1, 2, \cdots, n\}, i \neq j, \forall t \in \{0, 1, \cdots, T-1\}, (\tau_i(t) \neq \tau_j(t+1)) \vee (\tau_i(t+1) \neq \tau_j(t))$ (no two agents swap vertices between consecutive timesteps), and **Condition 3:** $\forall i \in \{1, 2, \cdots, n\}, ((\tau_i(0) = s_i) \wedge ((\tau_i(T) = d_i))$ (all agents start from the pre-settled position and reach their goal at the end of the task), to make sure the paths are collision-free and effective.

4. Social behavior learning

In this section, we delve into the integration of the Social Value Orientation (SVO) concept within the Multi-Agent Path Finding (MAPF) problem, introducing a novel MAPF framework named SYLPH that incorporates social preferences. We explore this integration from two primary perspectives (as shown in Fig. 2): the generation of hierarchical policies and the influence of upper level SVO policies on the formulation of action-oriented policies. This integration aims to highlight how the incorporation of SVO not only enriches the policy depth available to agents but also enhances their problem-solving efficacy within the MAPF context, enabling a more nuanced context.

4.1. Partner selection

In this work, introducing SVO into the MAPF process is aimed at mitigating potential social dilemmas. A crucial initial question is identifying the origins of these dilemmas from the perspective of an agent, specifically determining with whom to collaborate to effectively solve them. Since SVO is used to balance an agent's self-interest with collective interests, representing the collective interests becomes our first task. An intuitive approach is to average the rewards of other members in the team or other members within a certain observation range as the collective interests. However, there are two significant disadvantages to doing so. First, this mixed collective interest representation of multiple agents is biased from the perspective of the current agent. Because it may mix information from agents that are irrelevant to the social dilemma encountered by the current agent, which will confuse the agent. Second, it difficult for the neural network to establish relationships between the agent and other fellow agents under such a collective interest representation. Compressing too much information into an average reward causes the neural network to lose significant information, making it challenging to reason about the original relationships between agents. Therefore, in this paper we let the agent choose a *partner* for SVO determination. The term *partner* refers to this other specific agent involved in the dyadic team with the primary (ego) agent. Additionally, it is also worth mentioning that the agent-partner pair is not bi-directional, which means an agent's partner may choose the third agent as its partner. This approach draws parallel insights from some recent autonomous driving research [51,63], which suggests that an autonomous vehicle's behavior is predominantly influenced by the vehicle in its immediate vicinity rather than others within the broader field of view (FoV). These studies have led us to realize that focusing on a single partner can sometimes be more advantageous than considering multiple agents, because it allows neural networks to more easily reason about the relationships between different team members.

4.1.1. Global partner selection

In order to find the most impactful partner among all agents, we formalize the method of selecting partners in Algorithm 1. Specifically, the procedure for selecting a temporary partner in the context of MAPF with an emphasis on SVO can be detailed in four steps:

- Calculating single agent optimal paths [Line 1-2]: Initially, for each agent within the system, an optimal path is computed using the $A*$ algorithm based on the current environmental configuration $(\mathcal{G}, \mathcal{A}, \mathcal{D})$. Each agent's path is delineated as a sequence of vertex coordinates:

$$\begin{aligned} &\mathcal{T}_{A*} = \{\tau^1_{a*}, \cdots, \tau^n_{a*}\} \\ &\tau^i_{a*} = \{v^i_0, v^i_1, \cdots, v^i_T\} \qquad i = 1, 2, \cdots, n. \end{aligned} \tag{1}$$

Algorithm 1: Selection of Temporary Partner.

Input: The grid obstacle map: $\mathcal{G}$; the set of all agents' current positions: $\mathcal{A}$; the set of all agents' goals: $\mathcal{D}$.
Output: All agent's partner: $\mathcal{P} \in \mathbb{N}_{+}^{n\times 1}$; The potential overlap of optimal paths among all agents: $\mathcal{O}_L \in \mathbb{R}^{n\times n}$.

1 Initialize $\mathcal{P}$ as $\emptyset$, and $\mathcal{O}_L$ as $\mathbf{0}^{n\times n}$; ▷ single optimal paths
2 Compute A* paths $\mathcal{T}_{A*} \leftarrow \{\mathcal{G}, \mathcal{A}, \mathcal{D}\}$ for all agents;
3 **for** $\forall v_t^i \in \mathcal{T}_{A*}$ **do**
4 Determine the direction of the agent: $\delta_v^i \leftarrow \{v_t^i, v_{t+1}^i\}$; ▷ optimal paths direction fields
5 **end**
6 **for** $\forall v \in \mathcal{T}_{A*}$ **do**
7 **if** $\exists v_{t_i}^i, v_{t_j}^j \equiv v$ **and** $i \neq j$ **and** $\delta_v^i \neq \delta_v^j$ **then**
8 $\mathcal{O}_L[i,j] = \mathcal{O}_L[i,j] + \gamma_{ol}^{t_i} + \gamma_{ol}^{t_j}$; ▷ overlap calculation
9 $\mathcal{O}_L[j,i] = \mathcal{O}_L[j,i] + \gamma_{ol}^{t_i} + \gamma_{ol}^{t_j}$;
10 **end**
11 **end**
12 **foreach** $row \in \mathcal{O}_L$ **do**
13 **if** $row = 0$ **then**
14 $\mathcal{P}[\texttt{Index}(row)] = \texttt{Index}(row)$; ▷ temporary partner
15 **else**
16 $\mathcal{P}[\texttt{Index}(row)] = \arg\max_i row[i]$;
17 **end**
18 **end**

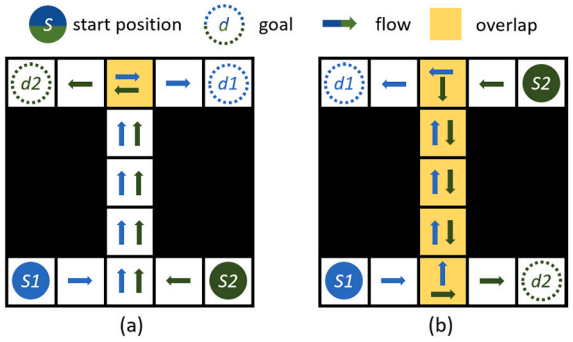


Fig. 3. Overlap in optimal path direction fields between agents, caused by their varied starting and goal position configurations within the same map. In scenario (a), two agents traverse a narrow corridor moving in the same direction, which results in minimal conflict. Conversely, scenario (b) involves agents needing to navigate in opposite directions within the same space, significantly heightening the potential for conflict due to the direct opposition in their intended paths. According to Algorithm 1, the calculated overlap in scenario (a) is markedly less than that in scenario (b). This distinction aligns well with intuitive expectations and the specific objectives of managing social dilemmas within multi-agent path finding. The larger overlap in scenario (b) suggests a higher degree of conflict and necessitates more critical intervention or strategy adjustment to avoid collision or deadlock, highlighting a situation of greater social distress.

Recent studies [64,65] have proposed methods to replan shortest paths to minimize potential conflicts among agents when the shortest path is not unique. However, in MAPF tasks, the primary challenges arise in highly structured areas of the map, such as doors and narrow corridors, where space resource is limited. In these cases, the optimal paths for agents are often unique in these areas. Therefore, A* algorithm efficiently identifies agents that have the greatest potential impact on the ego agent's subsequent planning.

- Determining the direction fields of these individual paths [Line 3-5]: Based on the optimal path computed, the optimal path direction field can be determined from the sequence of vertex coordinates. Specifically, the direction at any vertex δ_v^i along an agent's optimal path is determined by its coordinates v_t^i and the vertex coordinates at the next time step v_{t+1}^i. The ordered set of these directions forms the optimal path direction field of the agent.
- Computing the overlap of optimal path direction fields between agents [Line 6-11]: With the direction field of individual paths established, the next step involves assessing the overlap between these direction fields. A key idea is that if the direction fields of agents are in the same direction at a cell, i.e. $\delta_v^i = \delta_v^j$, then we consider no potential conflicts at this cell between agents and therefore consider this cell's overlap as 0. However, if the agents are not moving in the same direction at a cell (as shown in Fig. 3), indicating potential crossing points or interactions, the overlaps should be assessed according to Line [8-9]. The impact of such overlaps on an agent's decision-making weakens as the distance between the overlapping cell's position and the agent's current location increases. The decay of overlap impact based on distance introduces the concept of a decay factor $\gamma_{ol} \in (0, 1]$, which is a predefined hyper-parameter. This factor adjusts the significance of distant overlaps on an agent's current choices, with a higher value indicating a more far-sighted agent that considers distant overlaps more significantly, while a lower value indicating an agent more focused on immediate or nearby overlaps. Mathematically, the overlap between two agents' paths is quantified as the weighted sum of all overlapping cells within their path direction fields:

Algorithm 2: Fixed Partner Update Criteria.

Input: All agent's temporary partner: $\mathcal{P} \in \mathbb{N}_+^{n\times 1}$; The potential overlap of optimal paths among all agents: $\mathcal{O}_L \in \mathbb{R}^{n\times n}$.
Output: All agents' fixed partners: $\mathcal{P}_{fix} \in \mathbb{N}_+^{n\times 1}$.

1 Initialize $\mathcal{P}_{fix}$ according to the temporary partner $\mathcal{P}$ provided by Algorithm 1 at the beginning of the episode;
2 **for** ∀agent i **do**
3 **if** $\mathcal{O}_L[i, \mathcal{P}_{fix}[i]] = 0$ **then**
4 **Update** $\mathcal{P}_{fix}[i]$ **as** $\mathcal{P}[i]$; ▷ update fixed partner
5 **else**
6 **Keep** the partner of i unchanged. ▷ keep fixed partner
7 **end**
8 **end**

$$\mathcal{O}_L[i,j] = \mathcal{O}_L[j,i] = \underbrace{\sum_{v\in(\tau^i_{a*}\wedge\tau^j_{a*})}}_{\text{all overlapping cells}} \underbrace{\gamma_{ol}^{\texttt{Index}_{\tau^i_{a*}}(v)}}_{\substack{\text{the importance of}\\ \text{cell } v \text{ to agent } i}} + \underbrace{\gamma_{ol}^{\texttt{Index}_{\tau^j_{a*}}(v)}}_{\substack{\text{the importance of}\\ \text{cell } v \text{ to agent } j}} \tag{2}$$

where $\texttt{Index}_{\tau^{i/j}_{a*}}(v)$ represents the index number of vertex v in the $A*$ path $\tau^{i/j}_{a*}$ of agent i/j. The weight of each overlapping cell is adjusted by the decay factor γ_{ol}, relative to its distance from the agent's current position. This method provides a nuanced approach to evaluating potential path conflicts, allowing agents to prioritize their immediate navigation decisions while still accounting for future interactions.

- Finding a temporary partner [Line 12-17]: Based on the overlap analysis, agents are then paired or assigned a temporary partner. If an agent's optimal path direction field does not exhibit any overlap with the direction fields of other agents, the agent defaults to selecting itself as its partner. This scenario indicates that the agent can proceed without the need to adjust its path in response to potential conflicts with others, allowing for path finding towards its goal without external coordination. Conversely, if there is an overlap between an agent's path direction field and that of one or more other agents, the agent will choose as its temporary partner the agent with which it has the largest weighted overlap.

After establishing the method above for selecting a temporary partner, the next step involves delineating the criteria for selecting and switching partners. We propose that updates to partner selection may not have to occur per timestep to ensure system stability and consistency in agent interactions. The formal process for updating partners is outlined in Algorithm 2. Firstly, an agent selects a temporary partner at the start and treats this agent as its initial fixed partner. This fixed partnership remains unchanged until the overlap in the optimal path direction field between the agent and its fixed partner is eliminated ($\mathcal{O}_L[i, \mathcal{P}_{fix}[i]] = 0$), indicating that any potential social dilemma or conflict with this particular partner has been resolved. By adopting this method, updates regarding the agents' partner occur asynchronously. Such a mechanism not only reflects a realistic and practical approach to managing interactions within a dynamic multi-agent environment but also significantly contributes to the overall stability of the system. This strategy allows for the focused resolution of conflicts with specific partners before considering a shift to new partner dynamics, ensuring that changes in partnerships are meaningful and based on resolved interactions rather than fluctuating frequently without resolving underlying conflicts.

The algorithm introduced in this subsection quantifies the degree of conflict between agents by measuring the extent of their optimal path overlaps, thereby offering a formal method to express social dilemmas within a multi-agent environment. By precisely delineating the magnitude of these social dilemmas, the approach facilitates the identification of the agent that presents the greatest potential for conflict. This approach to clarifying social dilemmas enables the SVO mechanism to be applied more effectively. By focusing on resolving the most significant potential conflicts, agents can better navigate their environment, avoid collisions, and reach their goals in a more efficient manner.

4.1.2. Local partner selection

Building on the Partner Selection method introduced in Section 4.1.1, this section proposes a variant designed for scenarios where the communication range is limited. In this case, the ego agent can only select temporary partners from agents within its communication range, as we assume that only those agents can exchange information with the ego agent to provide the necessary input for Algorithm 1. It is important to note that introducing a communication range necessitates modifications to our partner update method. Specifically, unlike the original condition $\mathcal{O}_L[i, \mathcal{P}_{fix}[i]] = 0$ in Line 3 of Algorithm 2, we update it to:

$$\mathcal{O}_L[i, \mathcal{P}_{fix}[i]] \leq \xi, \tag{3}$$

where ξ is a pre-set threshold ($\xi = 1$ in practice). This adjustment ensures that when an agent with greater influence on the ego agent enters its communication range, the ego agent can shift its attention to the new partner while ensuring that the previous conflict is almost resolved. For example, consider two agents moving through a narrow corridor in the same direction, where their optimal paths exhibit a small degree of overlap. Under these conditions, they initially regard each other as partners. However, when an agent approaching from the opposite direction enters the communication range, they should be able to switch partners to enable better coordination. Compared to the vanilla partner selection method, the communication-range-restricted partner selection introduced in this section is better suited for decentralized settings, where strict locality constraints are required.

4.2. SVO generation

4.2.1. Socially-aware reward

Inspired by *skill learning* [59,66], our approach moves away from the way of assigning fixed, immutable SVO to agents. Instead, we conceptualize SVO as a temporally extended variable - a dynamic attribute that does not reside with any single agent but rather emerges from the patterns of coordinated behavior among agents. This reconceptualization recognizes that SVO is not a one-size-fits-all attribute. This dynamic, learnable SVO enables agents to adjust their social preferences based on the current context and interactions with other agents. Agents with dynamic SVO can change their behavior to meet the needs of the team, helping in avoiding live-/deadlocks and reduces the chances of prolonged conflicts, especially in highly structured maps.

In this light, the selection of an SVO is treated as an additional policy $\pi_\phi(z|z')$, akin to skill selection in skill learning frameworks. The training of this SVO policy is conducted in tandem with the training of the agents' action policies, creating a synergistic relationship where both policies are interconnected through the mechanism of SVO. This linkage depends on partner selection as outlined in Section 4.1, where the choice of partner directly influences the dynamics of the SVO policy. Specifically, in our context, any agent and its partner must reach their respective goals due to task requirements. Since learning-based planners rely on parameter sharing to improve scalability, a self-interested reward structure is necessary to motivate the agent to pursue its own goal. In our work, the reward structures of the SVO policy R_i^s and action policy R_i^a are as follows:

$$\begin{aligned} R_i^s &:= (R_i^{ex} + R_p^{ex}) \;/\; \rho, \quad i, p \in \mathcal{A} \\ R_i^a &:= \cos Z \cdot R_i^{ex} + \sin Z \cdot R_p^{ex}, \quad Z \sim \pi_\phi(z|z'), \end{aligned} \tag{4}$$

where, R_i^{ex} denotes the external reward for agent i, adhering to the reward configuration established in our previous research [7,18,41]. Z denotes agent i's SVO. The hyper-parameter ρ plays a role in calibrating the influence of the SVO policy reward on the agent's learning process, allowing for fine-tuning to achieve desired behaviors. However, when an agent insists on following its optimal path, its corresponding partner may have to incur an additional external penalty to manage the arising conflict, embodying the zero-sum nature of their interaction. Therefore, we constraint that the agent's SVO Z, should be between egoistic ($Z \approx 0°$) and prosocial ($Z \to 45°$). When Z satisfies this restriction and all external rewards are non-positive (consistent with our previous research), R_i^a is monotonically non-increasing within the domain of $[0°, 45°]$.[2] It implies that in cases devoid of conflict, agents are predisposed towards egoistic behavior, thereby maximizing their own long-term cumulative rewards R_i^a. Conversely, in situations where conflicts exist, there is a tendency for agents to adopt a more prosocial demeanor to maximize R_i^s. Such a shift facilitates the achievement of superior long-term rewards for the group formed by the agent and its partner, highlighting the utility of SVO in mediating self-sacrifice for collective gain.

In the SVO-based reward structure as shown in Eq (4), R_i^s is conceptualized as the aggregate of the external rewards received by both the agent and its selected partner. This reward structure is rooted in the intention to encourage agents to adopt SVO choices that enhance the collective well-being of the small group (consisting of the agent and its partner). Through this tightly coupled mechanism, the framework incentivizes the agent to learn and select SVOs that are not only beneficial to itself, but also advantageous to its cooperative interactions, thereby facilitating coordinated maneuvers. R_i^a, on the other hand, is based on the definition of SVO. This ensures that the agent's actions are coherent with its selected SVO Z. The reward received through R_i^a motivates the agent to execute actions that are in harmony with its SVO, fostering a congruent and integrated approach to decision-making and behavior. By simultaneously maximizing cumulative R_i^s and R_i^a, the model can derive both a upper-level SVO policy and a lower-level action policy:

$$\begin{aligned} \pi_\phi^*(z_i|z_i') &= \arg\max_\phi \mathbb{E}_{\tau_i \sim \pi_\theta}[\sum_{t=0}^{T} \gamma^t R_i^s]; \\ \pi_\theta^*(a_i|o_i, z_i') &= \arg\max_\theta \mathbb{E}_{\tau_i \sim \pi_\theta, Z_i \sim \pi_\phi}[\sum_{t=0}^{T} \gamma^t R_i^a]; \end{aligned} \tag{5}$$

where $\gamma \in (0, 1]$ is the discount factor, and T is the time horizon. z_i and $z_i' \in [0°, 45°]$ denote the SVO value of agent i at the current and previous time steps, respectively. Additionally, o_i represents the agent's observation, while a_i is used to denote its action.

4.2.2. Policy optimization

Proximal Policy Optimization (PPO) stands out as a highly favored framework in the domain of reinforcement learning (RL), celebrated for its stability, straightforward hyper-parameter tuning, and impressive performance. In our work, we use a variation of PPO [67] to support the training of SYLPH agents while distinguishing this approach from vanilla PPO by integrating an additional layer: a higher-level, socially-aware policy. This novel layer complements the action policy, equipping agents with social behavior in addition to moving towards their goals. We call this novel variation as Social-aware Multi Policy PPO (SMP3O).

In the context of SMP3O, where the framework needs to optimize two policies (the SVO policy $\pi_\phi(z_i|z_i')$ and the action policy $\pi_\theta(a_i|o_i, z_i')$) simultaneously, it becomes important to consider the losses associated with each policy independently. We formalize the losses for these policies as $\mathcal{L}_{\pi_\phi}$ and $\mathcal{L}_{\pi_\theta}$:

[2] Proof can be found in Appendix B.

$$\mathcal{L}_{\pi_\phi} = \mathbb{E}_t[\min(r^t_{svo}(\phi)\hat{A}^t_{action}, clip(r^t_{svo}(\phi), 1-\epsilon, 1+\epsilon)\hat{A}^t_{action})]$$
$$\mathcal{L}_{\pi_\theta} = \mathbb{E}_t[\min(r^t_{action}(\theta)\hat{A}^t_{svo}, clip(r^t_{action}(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^t_{svo})] \tag{6}$$

where $r^t_{svo}(\phi) = \frac{\pi_\phi(z^t_i|z^{t-1}_i)}{\pi_{\phi_{old}}(z^t_i|z^{t-1}_i)}$ and $r^t_{action}(\theta) = \frac{\pi_\theta(a^t_i|o^t_i,z^t_i)}{\pi_{\theta_{old}}(a^t_i|o^t_i,z^t_i)}$ are the probability ratios of the action under the new policy $\pi_{\phi/\theta}$ over the old policy $\pi_{\phi_{old}/\theta_{old}}$. ϵ is a hyper-parameter that defines the clipping range to avoid excessively large policy updates. $\hat{A}^t_{svo}$ and $\hat{A}^t_{action}$ are the advantage functions, which estimate how much better a particular SVO/action is if it was taken over the average. Unlike the policy loss in PPO, which is calculated by directly associating the advantage of an action with the likelihood ratio of that action under the current policy versus the old policy, SMP3O introduces a novel approach. The core insight behind integrating the SVO policy with the action policy in the SMP3O framework lies in leveraging the hierarchical nature of these policies. Specifically, we provide a cross-utilization advantages mechanism, where each policy derives indirect benefits from the learning signals of the others. As the upper level, the SVO policy plays a pivotal role in redistributing action rewards, thereby ensuring that actions are aligned with the social preferences of the agent (as illustrated in Eq. (4)).

Therefore, when calculating the action policy loss $\mathcal{L}_{\pi_\theta}$, $\hat{A}^t_{svo}$ should be combined with the action policy $\pi_\theta(a_i|o_i, z'_i)$. This mechanism is particularly beneficial in scenarios characterized by social dilemmas. In such situations, even though the SVO policy may dictate a course of action that aligns with long-term group benefits or conflict resolution, it might disadvantageous to the agent in the short term. SMP3O encourages the action policy to overcome this potential short-sightedness by following the guidance of the SVO policy, which can be expressed as:

$$\triangle\theta \propto \nabla_\theta \mathbb{E}[\hat{A}_{svo} \cdot \log \pi_\theta(a_i|o_i, z'_i)] \tag{7}$$

This approach fosters behaviors that potentially sacrifice immediate individual gains and instead contribute to the collective well-being of the agent pair, emphasizing group rewards over individual rewards. When calculating the SVO policy loss $\mathcal{L}_{\pi_\phi}$, incorporating the action advantage $\hat{A}_{action}$ facilitates the formation of a feedback loop. Because of reward reallocation and action loss, agent action will be consistent with the SVO decision. Under this premise, the aforementioned feedback enables the SVO policy to assess the quality of executed actions, guiding the agent towards decisions that concurrently optimize both SVO alignment and action efficacy. Formally:

$$\triangle\phi \propto \nabla_\theta \mathbb{E}[\hat{A}_{action} \cdot \log \pi_\phi(z_i|z'_i)] \tag{8}$$

This bidirectional reinforcement between the SVO and action policies create a synergistic learning environment. It not only aligns individual actions with broader social goals but also ensures that the SVO policy is refined based on the outcomes of these actions, establishing a coherent and mutually beneficial relationship between social behaviors and action executions.

To enhance the stability of an agent's SVO, we rely on supervised learning. Drawing from our prior experiences with valid and blocking losses, the formulation for the SVO stability-enhancing loss, $\mathcal{L}_{stab}$, can be expressed as follows:

$$\mathcal{L}_{stab} = \mathbb{E}[z_{i,exp} \log(\pi_\phi(z_i|z'_i)) + (1 - z_{i,exp}) \log(1 - \pi_\phi(z_i|z'_i))] \tag{9}$$

Here, $\mathcal{L}_{stab}$ is fundamentally the cross entropy between the SVO policy output by the neural network and the expected SVO probability distribution. For shaping our desired SVO distribution, we introduce an under-relaxation factor, α, within the range [0,1], to modulate the adjustment of the SVO towards the expected value. The expected SVO, z_{exp}, is determined by blending the current SVO, z, with previous SVO, z', using α:

$$z_{i,exp} = \alpha z'_i + (1 - \alpha) z_i \tag{10}$$

The calculation of α is designed to reflect the degree of overlap between the agent and its partner:

$$\alpha = clip(\mathcal{O}_L[i, p], 0, \kappa)/\kappa \tag{11}$$

where $\mathcal{O}_L[i, p]$ represents the optimal paths overlap between agent i and its partner p. κ sets a boundary condition for the overlap magnitude; when the actual overlap between agents falls below κ, the agent is permitted to modify the SVO with greater latitude. The rationale behind this formulation is to encourage SVO policy stability especially under conditions of significant overlap, thereby mitigating potential volatility in the agent's social behavior. As the overlap diminishes, indicating reduced immediate conflict or competition for resources, the model permits more flexible adjustments to the SVO, aiming to foster higher degrees of cooperation and coordination.

4.3. Enhancing policy with SVO

Tie-breaking remains essential in learning-based MAPF, as reinforcement learning alone does not guarantee consistent conflict resolution, especially in dense or structured environments, where agents may select equally optimal actions, leading to dead-/live-locks. Previous methods, such as SCRIMP [18], rely on state-value-based post-processing, requiring multiple model inferences, which is computationally expensive and lacks termination guarantees. Other works [17,44] use rule-based and search-based traditional MAPF algorithms as post-processing, ensuring collision-free paths but failing to proactively reduce conflicts or improve decision-making ability of model. [68] proposed two post-processing methods, IStay and IAVoid, which enable rapid conflict resolution. Our

Algorithm 3: SVO-based Tie-breaking Method.

Input: All agent's action set $\mathcal{A} \in \mathbb{N}^n$ and SVO set $\mathcal{Z} \in \mathbb{N}^n$ output by the neural network.
Output: Adjusted conflict-free action set $\mathcal{A}' \in \mathbb{N}^n$; Socially-aware adjusted rewards $R \in \mathbb{R}^n$.

1 **Sort** agents in descending order of SVOs $\mathcal{Z}$ to obtain the consideration chain C; ▷ initialize consideration chain
2 **while** $len(C) > 0$ **do**
3 $i \leftarrow C$.pop() ▷ check action status
4 **if** $\mathcal{A}[i] \in$ *Invalid Action* **then**
5 $\mathcal{A}'[i] = 0$ (stay idle); $R[i] = -2$ (Collision Penalty); ▷ invalid
6 **if** $0 \in$ *Restricted Action* **then**
7 **for** $\forall j$ **causing** *Restricted Action* **do**
8 C.append(j) **If** $j \notin C$
9 **end**
10 **else if** $\mathcal{A}[i] \in$ *Restricted Action* **then**
11 **for** $\forall j$ **causing** *Restricted Action* **do**
12 $\mathcal{A}'[i] = 0$ (stay idle); ▷ restricted
13 $R[i] = -2$ **If** $\mathcal{Z}[i] > \mathcal{Z}[j]$; $R[j] = -2$ **If** $\mathcal{Z}[j] > \mathcal{Z}[i]$;
14 **end**
15 **Repeat** Line 6 to 9
16 **else**
17 $\mathcal{A}'[i] = \mathcal{A}[i]$; $R[i] =$ External Reward; ▷ normal
18 **end**
19 **end**

tie-breaking mechanism builds upon the core idea of IStay, but further introduces Social Value Orientation (SVO) to explicitly differentiate between agents and guide conflict resolution. Additionally, our method is integrated into the learning process rather than applied as a separate post-processing step. In other words, our tie-breaking mechanism helps improve the model's decision-making quality itself. This eliminates the need for costly post-hoc model inference while mitigates the need for tie-breaking, making SYLPH both computationally effective and adaptively scalable.

The tie-breaking mechanism in our paper outlines an approach to conflict resolution among agents based on their social preferences. We first need to clarify the definitions of *Invalid Actions* and *Restricted Actions*. Invalid Actions are actions that would cause an agent to collide with a static object or boundary within the environment, rendering the action unfeasible. Restricted Actions are actions would result in dynamic collisions between agents, depending on their intended movements during the same timestep. To process action validation and conflict resolution, agents are sorted based on their SVOs at the begin, from the most prosocial to the most self-interested [Line 1]. This ranking dictates the priority with which each agent's actions are validated and adjusted in the face of potential conflicts. Then, agents are checked in descending order of prosociality for invalid actions [Line 2-9]. If an invalid action is detected, the agent's action is changed to 'stay idle' to avoid static collision [Line 5]. If 'staying idle' still results in a restricted action (potential collision with another agent), this action is flagged, and the situation needs further resolution [Line 6-9]. The next operation of Algorithm 3 is resolving restricted actions. For agents causing dynamic collisions, their actions are set to 'stay idle' [Line 12]. If waiting does not resolve the collision, actions of conflicting agents are reassessed and adjusted at the end of the consideration chain. The chain continues until all agents have valid actions [Line 15]. Algorithm 3 runs recursively until the termination condition is met: the consideration chain becomes empty [Line 2], indicating that no agent keeps any Invalid Actions or Restricted Actions. Only agents with higher SVOs (more prosocial) receive penalties for collisions [Line 13], reflecting their role in facilitating smoother group dynamics by yielding. Self-interested agents are less likely to be penalized, preserving their direct routes or actions unless absolutely necessary. This tie-breaking mechanism effectively uses SVO as a prioritization tool in conflict resolution, where more cooperative agents are more inclined to compromise for the greater good. The algorithm potentially makes these systems more acceptable and understandable in human-centric environments.

Fig. 4 illustrates an example scenario. The numbers in circles indicate the initial order in which agents are checked based on their SVOs, arranged in descending order. The top array represents the current consideration chain for resolving the collision, initially set as $[0, 1, 2, 3, 4]$. In Fig. 4(a), agent 0's action is assessed first. It appears valid and remains unchanged, assuming that if agent 2 can execute its current plan, there will be no conflict. Next, in Fig. 4(b), agent 1 is found to be performing an invalid action and is therefore set to stay idle. However, this results in a restricted action affecting agent 3 (Fig. 4(d)), prompting its addition to the list for evaluation. Before this, Fig. 4(c) shows that agent 2 is performing a valid action. When agent 3's action is evaluated, it is changed to 'stay idle,' necessitating a reevaluation of agent 2 after agent 4's actions are considered (as shown in Fig. 4(e)). Upon reconsideration, agent 2's action is deemed restricted (Fig. 4(f)), and it is set to stay idle. This change prompts the addition of agents 0 and 4 back into the consideration chain, as shown in Fig. 4(g). The deadlock is ultimately resolved when all agents are set to 'stay idle.' In this scenario, agents 1, 2, and 0 receive collision penalties due to conflicts with agents 3, 4, and 2 respectively.

5. Attention-based network

The network architecture described encompasses three distinct input components as shown in Fig. 5. The first channel is grid-level observation, essentially the detailed environment within the agent's field of view (FoV), centered around itself. The second channel is a vector that directs the agent towards its goal. The third channel comprises the SVOs of the agent and its partners, which are selected according to Section 4.1. For processing the grid-level observations within the FoV and the vector pointing towards the goal, we utilize

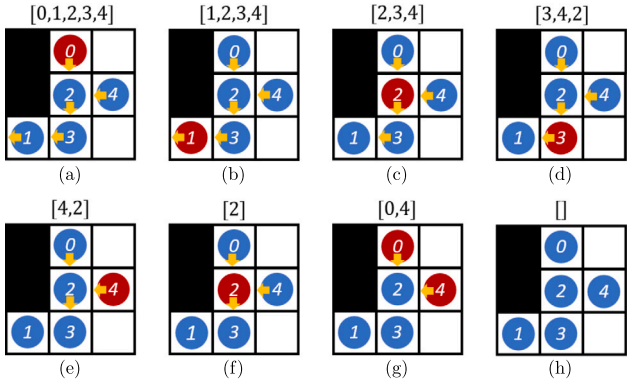


Fig. 4. The SVO-based tie-breaking mechanism example.

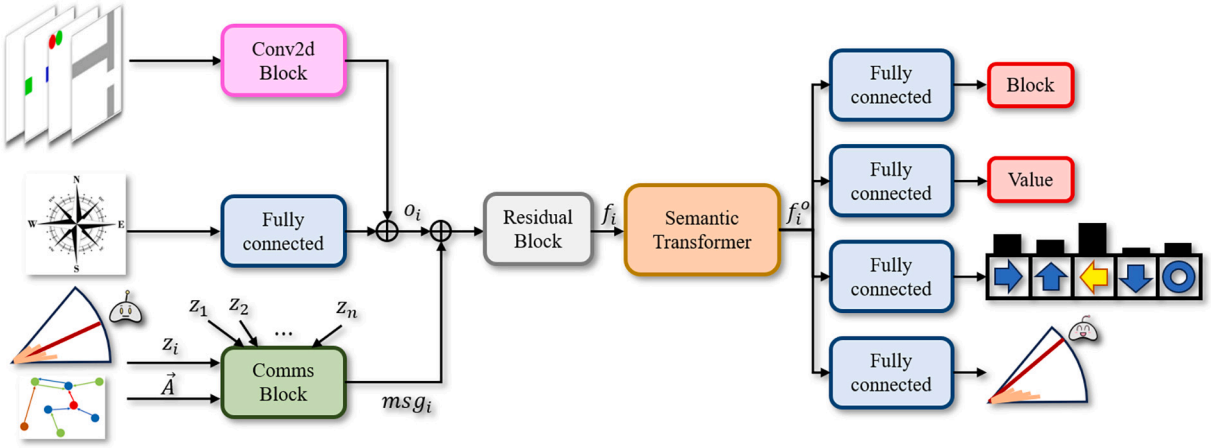


Fig. 5. Overview of the Network of SYLPH.

the efficient encoder as outlined in our prior research [7]. Specifically, the grid-wise observation undergoes processing through two 2D convolutional blocks [69], each containing two convolutional layers followed by a MaxPooling operation. The encoding of the goal vector is accomplished using a fully connected layer. Meanwhile, the SVOs of the ego agent and its partners are encoded using a communication block that is based on a Graph Transformer [38]. Subsequently, the embeddings generated from these three inputs are concatenated and fed through a residual block before being decoded by a semantic transformer. The outputs are then transformed into the agent's action and SVO policies, as well as blocking and value, through different fully connected layers and activation functions. This section will delve into the detailed implementation of the communication block and semantic transformer block.

5.1. Communication block

Our communication block draws inspiration from the Hop2Token aggregation approach detailed in [38]. Unlike methods that aggregate node information based on distance, this paradigm leverages the graph structure to classify nodes by hops. Taking Fig. 7 as an example, the red node represents itself. The blue nodes, which share a direct edge with the red node, are classified as 1-hop nodes. Similarly, the green nodes, which are connected to the blue nodes but not directly to the red node, are categorized as 2-hop nodes. This classification method provides a structured way to model neighborhood relationships beyond simple distance metrics. This approach aligns well with our application scenario, where agents are represented as nodes. The agent and its selected partners are considered to be connected via edges. As shown in Fig. 7, instead of merely aggregating information from direct neighbors, the ego agent performs attention-based information exchange by synthesizing information from nodes at different hop levels. This mechanism effectively utilizes existing knowledge about the importance of other agents, as nodes are pre-classified based on their significance before attention is applied. We integrate this hop-based communication mechanism with the partner selection Algorithm 1 from Section 4.1 to enhance knowledge utilization and information exchange among agents.

In the communication block illustrated in Fig. 6, the inputs include the global agent's social preference SVO z_i and the adjacency matrix $\vec{A}$ of the directed graph $\mathcal{G}_a$, which is constructed according to the partner selection mechanism. The first step involves converting $\vec{A}$, the directed adjacency matrix, into A, its undirected counterpart. Subsequent operations involve raising A to the 0th, 1st, and 2nd powers and symmetric normalization [70,71] (SN), yielding the identity matrices I, $\dot{A}$, and $\ddot{A}$ as multipliers for multi-

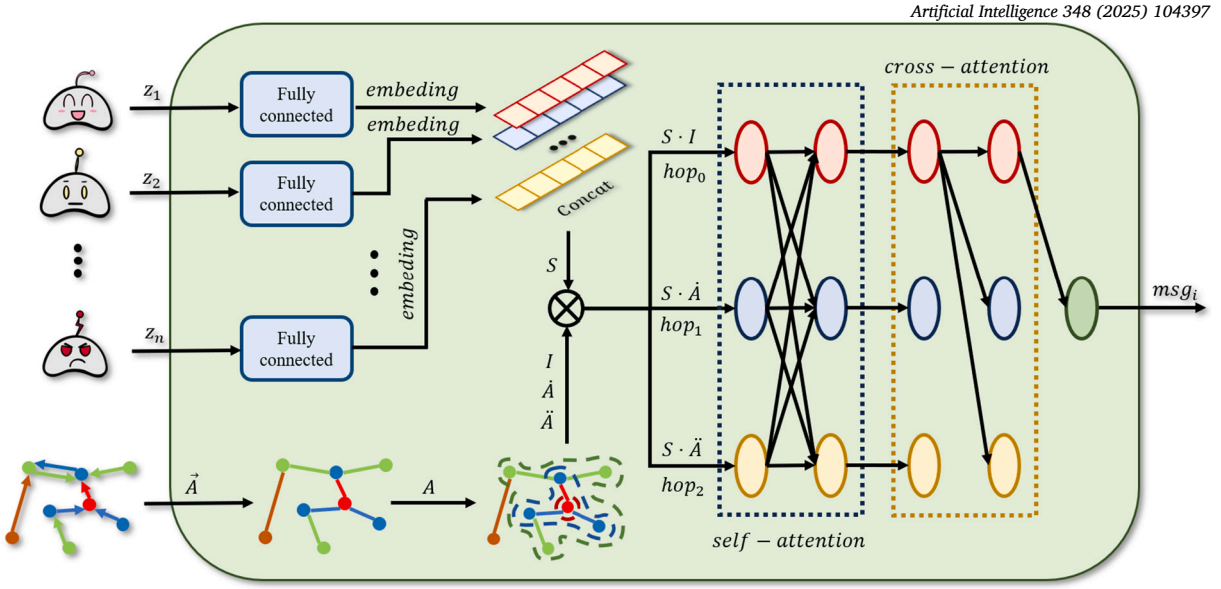


Fig. 6. Details of communication block.

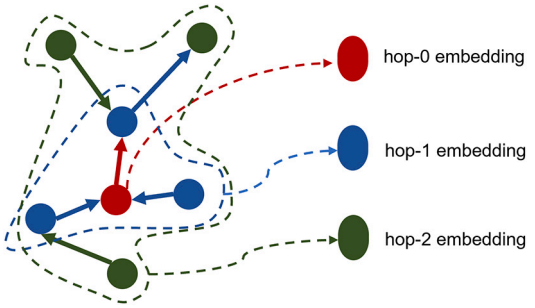


Fig. 7. Multi-hop embedding of nodes. The red node is itself, blue nodes are 1-hop nodes, and green nodes is 2-hop nodes. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

hops. Concurrently, the SVO probability distribution z_i is transformed into SVO embeddings via fully connected layers, serving as vertex features v_i. These vertex features are then combined with multi-hop multipliers to encapsulate the graph's entire feature into a condensed number of multi-hop nodes k ($k = 2$ in practice), significantly streamlining computations ($k \ll n$). Following this feature aggregation, the multi-hop nodes h_i are processed through multi-head self-attention and cross-attention layers. The output features of 0-hop node h_0, emerging from these attention mechanisms, are utilized as the encoder embedding msg_i for the SVO channel, effectively capturing the interactions and dependencies within the graph from agent i's perspective. Formally:

$$\text{Embedding}\begin{cases} A = \vec{A} \vee \vec{A}^T \\ I, \dot{A}, \ddot{A} = SN(A^0, A^1, A^2) \\ S = Concat(FF(z_1), FF(z_2), \cdots, FF(z_n)) \\ H = Concat(h_0, h_1, h_2) = S \otimes I, S \otimes \dot{A}, S \otimes \ddot{A} \end{cases} \tag{12}$$

$$\text{Self-attention}\begin{cases} Q_s^h, K_s^h, V_s^h = W_{Q,s}^h H, W_{K,s}^h H, W_{V,s}^h H \\ A_s^h = Att(Q_s^h, K_s^h, V_s^h) = Softmax(\dfrac{Q_s^h \cdot {K_s^h}^T}{\sqrt{d_k}}) \cdot V_s^h \\ \hat{h}_s = BN(H + Concat(A_s^1, A_s^2, \cdots, A_s^H) W_{O,s}) \\ H_s^o = BN(\hat{h}_s + FF(\hat{h}_s)) \end{cases} \tag{13}$$

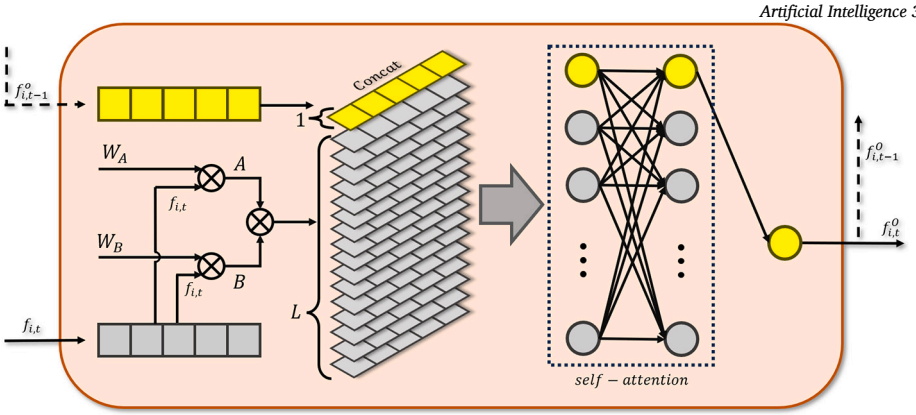


Fig. 8. Details of semantic transformer block.

$$\text{Cross-attention}\begin{cases} Q_c^h, K_c^h, V_c^h = W_{Q,c}^h h_0, W_{K,c}^h H_s^o, W_{V,c}^h H_s^o \\ A_c^h = Att(Q_c^h, K_c^h, V_c^h) = Softmax(\dfrac{Q_c^h \cdot K_c^{hT}}{\sqrt{d_k}}) \cdot V_c^h \\ \hat{h}_c = BN(h_0 + Concat(A_c^h, A_c^h, \cdots, A_c^H) W_{O,c}) \\ h_c^o = BN(\hat{h}_c + FF(\hat{h}_c)) \end{cases} \tag{14}$$

where $FF(\cdot)$ means the fully connected layer; $Concat(\cdot,\cdot)$ indicates the concatenate operator; $BN(\cdot)$ denotes batch normalization.

5.2. Semantic transformer block

In contrast to earlier efforts [44] that applied a Visual Transformer as an encoder to enhance embeddings for observations with a larger FoV, our approach uniquely integrates a Semantic Transformer into our existing Multi-Agent Path Finding (MAPF) framework, supplanting the LSTM component. This transformer serves as a decoder, analyzing and restructuring low-level features from diverse observational channels related to the agent, such as FoV observations, the directional vector to its goal, and the SVOs of agents. Inspired by the tokenizer approach used in image semantic segmentation as discussed in [72], we suggest that the agent's environmental state can also be captured in a similar way. By collecting and reorganizing low-level features, we aim to distill these into a predetermined number of semantic tokens ($L = 16$ in practice) via a spatial attention mechanism. These tokens, embodying high-level semantic concepts, articulate the agent's environmental state context and are subsequently processed by a standard transformer [73] for contextual reasoning. Notably, echoing the temporal reliance inherent in a long-short-term memory (LSTM) cell, our model aspires to imbue the semantic reasoning process with a memory / recurrent mechanism. As illustrated in Fig. 8, an additional memory token, representing the output from the previous timestep's Semantic Transformer module, is incorporated. The cross-attention outcomes between this memory token and the current timestep's generated semantic tokens are harnessed as the present output of the Semantic Transformer module. This refined output informs the generation of the agent's diverse policy actions, melding past insights with current observations to navigate the agent in a clever way. It can be expressed mathematically as follows:

$$S_T = Concat(f^O_{i,t-1}, ((f_{i,t} \cdot W_A) \otimes (f_{i,t} \cdot W_B))). \tag{15}$$

S_T represents a set of tokens that includes L semantic tokens alongside a memory token. It is then fed into a standard Transformer architecture, where it undergoes an update process facilitated by the self-attention mechanism (as shown in Eq. (13)) inherent to Transformers. The self-attention mechanism enables the model to dynamically weigh the importance of each token within S_T relative to the others, thereby refining their representations based on the contextual relationships within the set. Following the processing through the Transformer, the memory token is specifically extracted from the updated set S_T. This memory token serves a dual purpose: it not only represents the output of the semantic transformer block for the current timestep, but also becomes the input for the semantic transformer block in the subsequent timestep. This recurrent integration of the memory token preserves and transfers learned spatio-temporal information, allowing the model to maintain a thread of relevant context throughout the sequence of actions/decisions by both immediate semantic cues and historical knowledge.

6. Experiments

In this section, we give some simulations and experimental validation for the proposed framework and mechanism.

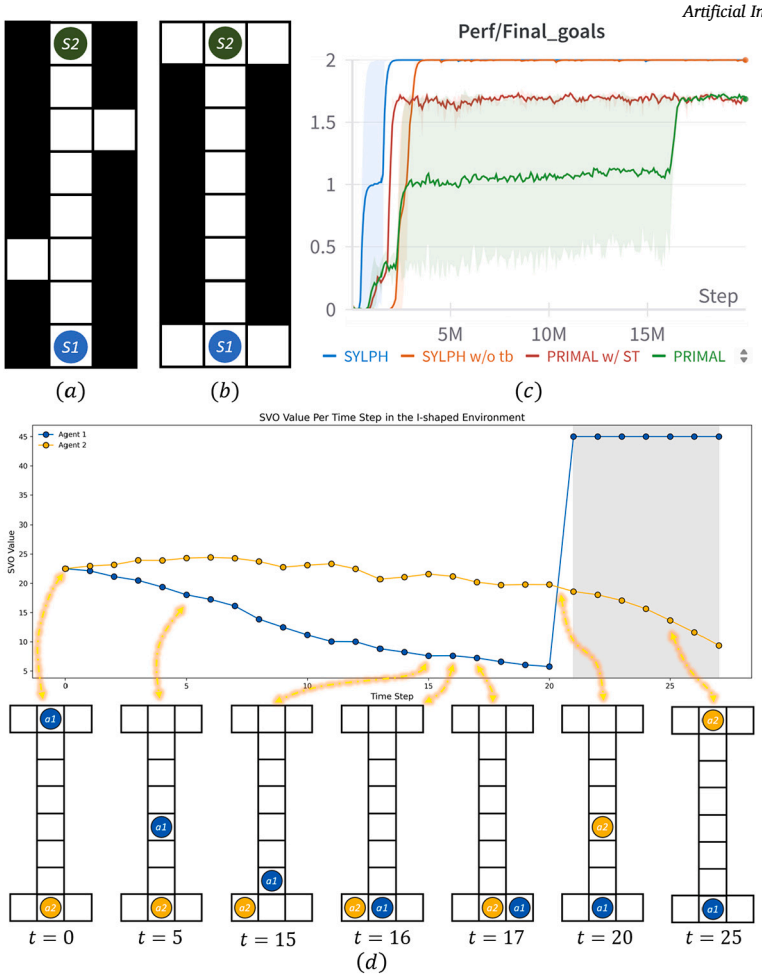


Fig. 9. Two agents symmetric pathfinding case study.

6.1. *Environment type*

This section evaluates the proposed framework's effectiveness across three distinct map types. Firstly, we consider random maps [7] (also known as uniform maps in some papers), a common choice among learning-based planners for both training and testing due to the large variance in its structure and complexity. This variability challenges the model's generalization capabilities, as the unpredictability in obstacle placement necessitates the learning of a diverse array of policies. Secondly, we assess performance on room-like maps [41], which are more structured than random maps and feature elements such as doorways, narrow corridors, and rooms. These structured obstacles compel agents to develop long-horizon planning capabilities to effectively find the path. Furthermore, room-like maps often contain cut vertices that can amplify minor errors into significant team-wide setbacks, highlighting the importance of planner stability. Lastly, given that this paper is dedicated to solving social dilemmas in MAPF problems, we also conduct tests on maze maps [13]. This map type is characterized by numerous long corridors, dead-ends, and various edge cases. Such features pose significant challenges for existing learning-based MAPF planners due to issues such as corridor symmetry and target symmetry [15]. To effectively address this type of problem, agents require a high level of coordination. SYLPH addresses these challenges and achieves better performance by introducing SVO for the agent, effectively breaking the symmetry.

6.2. *Symmetric pathfinding case study*

In our study, we first consider experiments in a completely symmetric environment, a setting inherently susceptible to social dilemmas. Using the configuration employed in [74] (as depicted in Fig. 9(a)), we placed two agents at opposing ends of a narrow corridor which are wide enough to accommodate just one robot at a time, with each agent's starting position serving as the other's goal. The first type of corridors have two recesses capable of fitting one robot each, making the pathfinding environment solvable. These recesses are symmetrically positioned, yet unlike [74], the corridor length and the recesses' locations are subject to random variation in our experiments. Further diversifying our experimental setup, we introduced an `I`-shaped map, presented in Fig. 9(b). Similar to the first environment, the narrow corridor's length within this map remains variable. Throughout training, we maintained

a probabilistic distribution for the occurrence of each map type: 80% ($p_r = 80\%$) for the corridor with recesses (recess map) and 20% ($p_I = 20\%$) for the I-shaped map.

To analyze the expected number of goals reached by PRIMAL[3] and SYLPH under these map probabilities, we consider the empirical training results, as shown in Fig. 9(c): 1) PRIMAL achieves an average of 1.7 goals per episode; 2) SYLPH achieves an average of 2 goals per episode. Given the probabilities of encountering each map type, we express the expected goal count as follows:

$$\begin{aligned} \mathbb{E}(g_{PRIMAL}) &= p_r \times 2 + p_I \times 1 = 1.7 \\ \mathbb{E}(g_{SYLPH}) &= p_r \times 2 + p_I \times 2 = 2 \end{aligned} \tag{16}$$

These equations reflect that PRIMAL struggles with the I-shaped map, whereas SYLPH maintains a consistent performance across both map types. PRIMAL's difficulty with the I-shaped map, despite its competence in the recess map, highlights a fundamental limitation: its agents exhibit symmetrical selfish behavior, which impairs their ability to resolve social dilemmas effectively. This tendency toward self-interest, inherently reinforced in PRIMAL-trained models, hinders high-level collaboration, often leading to live-/deadlocks rather than superior collective outcomes. SYLPH, on the other hand, breaks this symmetry by introducing heterogeneous social preferences (SVOs) among agents, thereby enhancing coordinated maneuvers and improving overall performance. This enables some agents to make more prosocial/selfless choices in such a completely symmetric environment, promoting the achievement of team goals over individual optimization.

For example, in the I-shaped map scenario (Fig. 9(d)), agent 1 initially exhibits more selfish behavior than agent 2, as indicated by its lower SVO value, and therefore proceeds to its goal first. Once agent 1 reaches its goal, their social preferences reverse (as shown in the gray-shaded region), with agent 1 becoming more prosocial than agent 2. The social behaviors exhibited by individuals within the team are closely aligned with their SVO values: more selfish agent tends to prioritize reaching its own goal, while more prosocial agent is inclined to adopt cooperative policy, even this leads to the sacrifice of personal interests. This dynamic adjustment and diversity of social preferences enables SYLPH to effectively resolve the coordination challenges posed by the I-shaped scenario, highlighting its adaptability and potential for addressing complex social dilemmas in symmetric environments.

6.3. Comparative experiments

6.3.1. Training setup

The model is trained on a machine equipped with an AMD Ryzen 9 9750X 16-Core Processor, 64 GB of RAM, and an NVIDIA RTX 3090 GPU with 24 GB of memory. During training, our map generator randomly generates environments with uniformly sampled size from $[10, 40]$, including the placement of 8 agents and their corresponding goal positions. For random maps, the obstacle density is uniformly sampled between $[0, 0.3]$; for room-like maps, the obstacle density is approximately 0.3; and for maze maps, the obstacle density is uniformly sampled between $[0, 0.6)$. Each episode terminates when all agents reach their goals or when the maximum time step 256 is exceeded. The learning rate is set to 1×10^{-5}, and the Adam optimizer is used to minimize the loss function. The batch size is configured to 256 to strike a balance between memory usage and convergence speed.

6.3.2. Performance comparison

In our comparison experiments, we primarily assessed three metrics: (1) the *success rate* across 200 instances under a similar configuration, (2) the average *episode length* required to complete these instances, and (3) the average ratio of agents arrive their goals (*arrival rate*) per instance among the 200 instances. The first metric directly assesses the effectiveness of the planner in achieving complete solutions across a wide range of scenarios. A high success rate indicates robustness and reliability, showing that the planner can consistently solve the MAPF problem under varying conditions. The second one reflects the efficiency of the pathfinding algorithm and the quality of the solutions it generates. The last metric is particularly revealing, as it accounts for the performance of individual agents within partially successful episodes. It provides a more detailed picture of a planner's performance, especially in scenarios where not all agents may reach their destinations due to complex interactions or partial failures. For traditional algorithms, success rate and arrival rate often coincide because these methods typically either succeed fully (all agents reach their goals) or fail entirely (one or more agents do not finish). Thus, these metrics tend to reflect the same performance aspect. The learning-based planners might allow for more flexibility in agent behavior, leading to situations where some agents succeed while others do not in the same instance. Therefore, assessing these methods requires a finer-grained metric like the arrival rate to capture the nuances of partial successes and individual agent failures, which the success rate alone might overlook.

To benchmark the performance of SYLPH against other state-of-the-art (SOTA) methodologies in MAPF field, we conducted a series of tests across a spectrum of extreme environment configurations. This rigorous testing was designed to evaluate how well SYLPH and its comparative baselines navigate increasingly complex scenarios, reflected in the type of the environments and the density of agent populations. Specifically, our test environment configuration is as follows:

- Random Maps: Utilized a 32×32 grid world with an obstacle density of 0.2 (*random-32-32-0.2*) and tested with varying numbers of agents: 50, 100, 150, 200, 250, and 300, to observe the scalability of the algorithms under different agent densities.

[3] All references to PRIMAL in this section are modified versions, not the original one. There are two primary differences: firstly, the model is trained using the Proximal Policy Optimization (PPO) framework rather than the Asynchronous Advantage Actor-Critic (A3C); secondly, the model's FoV incorporates heuristic maps designed in DHC.

Table 1
Time consumption of SYLPH and its baselines under different configurations.

Cases	*random*		*room-like*		*maze*	
Configuration	*32-32-0.2-200*		*32-32-0.3-100*		*32-32-0.5-32*	
Time Type	General	Success	General	Success	General	Success
SYLPH	100.944s	79.621s	48.643s	35.533s	5.867s	4.229s
SCRIMP	627.914s	402.702s	422.171s	127.057s	14.635s	9.967s
DCC	218.786s	–	56.918s	36.728s	5.427s	1.891s
EECBS	2.753$s^{\blacktriangle}$	60.001$s^{\blacktriangle}$	1.211$s^{\blacktriangle}$	52.515$s^{\blacktriangle}$	1.544$s^{\blacktriangle}$	4.247$s^{\blacktriangle}$
PIBT	0.259s	0.094s	0.090s	0.055s	0.036s	0.014s
LNS2	1.620$s^{\blacktriangle}$	0.429$s^{\blacktriangle}$	2.635$s^{\blacktriangle}$	2.496$s^{\blacktriangle}$	0.029$s^{\blacktriangle}$	0.029$s^{\blacktriangle}$
LaCAM-py	30.023s	28.120s	30.012s	20.015s	30.010s	29.319s
LaCAM3	30.079$s^{\blacktriangle}$	30.078$s^{\blacktriangle}$	30.084$s^{\blacktriangle}$	30.084$s^{\blacktriangle}$	30.070$s^{\blacktriangle}$	30.070$s^{\blacktriangle}$

- Room-like Maps: Similar in dimension and agent count to the random maps, these environments featured an increased obstacle density (≈ 0.3) and the orderly distribution of obstacles (*room-32-32-0.3*), simulating more structured environments with additional coordination challenges.
- Maze Maps: Given the heightened complexity and obstacle density of approximately 0.5 (*maze-32-32-0.5*), the amount of agents was adjusted to smaller team size: 8, 16, 32, 64, 128, and 256, to test the algorithms' efficiency in highly constrained environments.

To provide a comprehensive comparison, we included both SOTA learning-based MAPF planners and traditional MAPF algorithms as baselines[4]:

- SCRIMP [18]: This is a learning-based MAPF planner recognized as SOTA. It achieves high performance across various environments by leveraging communication learning and a value-based tie-breaking mechanism. However, SCRIMP tends to be time-consuming in densely populated environments.
- DCC [36]: Another SOTA learning-based method, DCC, utilizes an attention mechanism to select communication partners during interactions, effectively reducing the communication load. While DCC is faster than SCRIMP, it does exhibit a lower success rate.
- EECBS [26]: It represents a SOTA bounded suboptimal planner. Typically, setting the suboptimality factor as 1.2 strikes the best balance between path optimality and computational efficiency.
- PIBT [31]: It is a one-step, rule-based planner that achieves sub-optimal solutions at very low computational cost through an adaptive prioritization mechanism.
- LNS2 [29]: This is currently one of the top MAPF planners. It resolves path conflicts through priority-based planning and continuous iteration, achieving near-optimal solutions with extremely high success rates.
- LaCAM-py [32]: LaCAM employs a two-level search strategy to efficiently find solutions. The low-level search resolves positional constraints for individual agents, while the high-level search explores joint position sequences that satisfy those constraints.
- LaCAM3 [75]: This method, also known as Engineering LaCAM*, enhances the performance of LaCAM* through a combination of several improvement techniques, bringing LaCAM3 to the SOTA among traditional MAPF methods.

These baseline comparisons are intended to showcase SYLPH's adaptability and performance across different environmental complexities and agent densities. By testing SYLPH against both cutting-edge learning-based planners and established traditional algorithms, we aimed to highlight the framework's strengths, particularly in scenarios requiring advanced coordination and long-horizon coordination capabilities amidst varying degrees of environmental constraints.

In Fig. 10, the comparative performance of SYLPH across various map configurations, including random, room-like, and maze maps, demonstrates its superior capability over other learning-based planners, including DCC and SCRIMP. The success of SYLPH across these metrics – reliability, scalability, and performance in structured environments – highlights its effective resolution of symmetry conflicts, a common challenge in highly structured scenarios such as room-like and maze maps. This performance advantage is attributed to the diversity of social preferences within the agent population, enabling SYLPH to navigate complex interactions more effectively. Compared to SCRIMP, SYLPH is designed to avoid the costly post-processing procedures required during deployment. Instead, it improves model performance directly through training process, which not only leads to better results but also significantly reduces runtime, as shown in Table 1.[5] In contrast to the local selective communication mechanism used by DCC, SYLPH adopts a global selective communication mechanism. While DCC runs faster in agent sparse environments (maze 32-32-0.5-32), its computational speed drops significantly as agent density increases, which is slower than SYLPH in random 32-32-0.2-200 and room-like 32-32-0.3-100. Additionally, SYLPH consistently outperforms DCC in key performance metrics such as success rate across different scenarios. As shown in Table 1, SYLPH achieves faster inference speed than other learning-based methods while maintaining superior performance (as shown in Fig. 10).

When compared to traditional methods, such as the SOTA bounded-suboptimal MAPF planner EECBS, SYLPH demonstrates significantly better performance on both random and room-like maps, and achieves comparable results on maze maps. Against LaCAM-py, a

[4] We provide all the source of the implementations in Appendix A.3.

[5] The superscript ▲ indicates that the time is C++ time, otherwise it is $Python$ time.

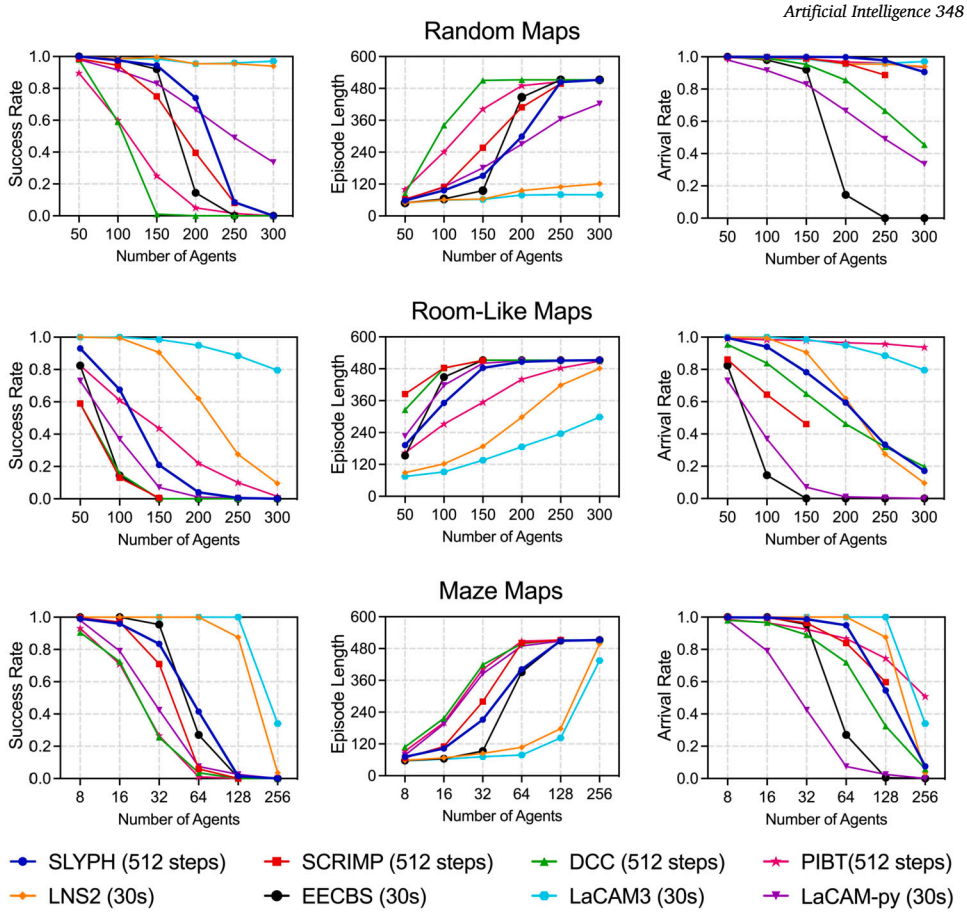


Fig. 10. Comparative results of SYLPH and other baseline algorithms across various maps and configurations, analyzed using three performance indicators: success rate, episode length, and arrival rate.

simplified implementation of LaCAM, SYLPH outperforms it in more structured environments in terms of both success rate and arrival rate. On random maps, SYLPH shows slightly lower success rates than LaCAM-py at only high agent densities (250 and 300 agents), but still achieves a much higher arrival rate. Compared to PIBT, an advanced rule-based one-step planner, SYLPH achieves higher success rates. But PIBT can maintain faster runtime and higher arrival rates. However, when compared with LNS2 and LaCAM3, two of the most reliable state-of-the-art MAPF solvers, SYLPH and all other planners fail to achieve comparable levels of performance. Additionally, learning-based methods are generally slower than traditional methods in terms of execution speed. While algorithm architecture is a primary factor, we believe that programming language also contributes significantly, as C++ is typically 10x to 100x faster than Python for computational tasks.

These results confirm that SYLPH not only rivals but sometimes surpasses the performance of traditional SOTA MAPF algorithms in highly structured environments. This is a significant achievement for a learning-based framework, demonstrating that SYLPH's socially-aware approach effectively enhances its applicability and effectiveness across diverse and challenging MAPF scenarios. This experimental evidence solidifies SYLPH's position as a new SOTA method in learning-based MAPF, capable of delivering high-performance outcomes where other learning-based methods may struggle.

6.3.3. Paired t-test

In order to rigorously assess the effectiveness of the SVO policy implemented in SYLPH, we conducted a series of paired t-tests to statistically analyze the performance differences between SYLPH and configurations with randomly assigned SVOs. This study was structured to compare SYLPH against three random SVO assignment methods:

- Random SVO Assignment at Each Step: This method, where an agent's SVO is randomly reassigned at every step, resulted in confusion and ineffective decision-making, as the frequent changes prevented the agents from leveraging any consistent strategy, leading to non-convergence.
- Static SVO Assignment for Each Episode: Assigning SVOs at the start of each episode and maintaining them throughout resulted in better stability and some level of task accomplishment.

Table 2
Paired t-test results.

Paired t-test	*random* *32-32-0.2-200*	*room-like* *32-32-0.3-100*	*maze* *32-32-0.5-64*
Static SVO Per Episode	$p < 0.001$	$p < 0.001$	$p < 0.001$
Static SVO Per Partner	$p < 0.001$	$p < 0.001$	$p < 0.001$

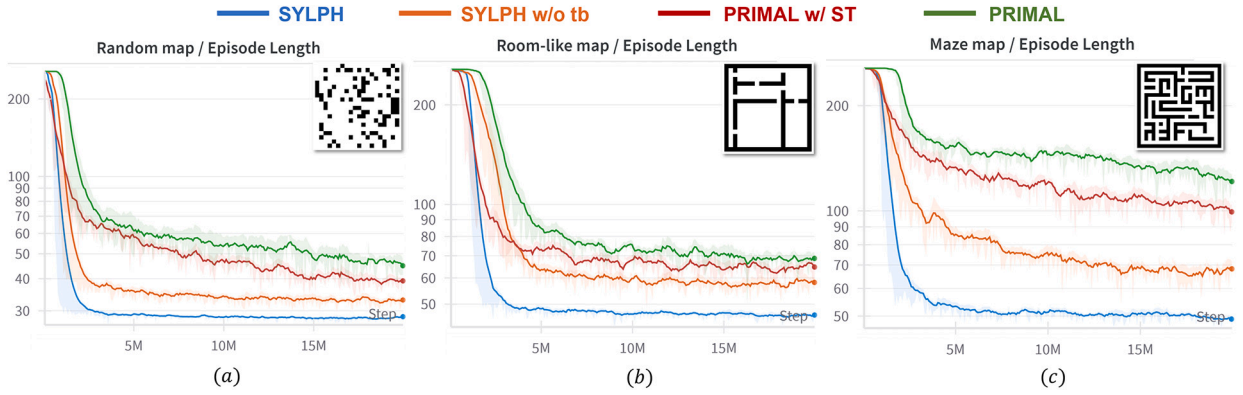


Fig. 11. (*a*), (*b*), and (*c*) represent the ablation experimental results of SYLPH variants in random maps, room-like maps, and maze maps, respectively. Consistent with Section 6.2, PRIMAL here refers to a variant of PRIMAL that adds heuristic maps.

- SVO Update Frequency Matching Partner Switching: Aligning the frequency of SVO updates with the frequency of switching partners shown similar performance as static SVO assignment for each episode.

The experiments were carried out on different map configurations (i.e. *random-32-32-0.2-200*, *room-32-32-0.3-100*, and *maze-32-32-0.5-32*). All of these models were trained with 8 agents.[6] The performance, measured in terms of success rate, was statistically analyzed over 200 instances for each configuration. Specifically, for the random SVO experiments, to prevent any unfairness due to randomness, we conducted the experiment 10 times and used the average success rate as the final result.[7] The results from these tests, as detailed in Table 2, revealed significant statistical differences between the performances of SYLPH and the random SVO methods. Table 2 does not include a comparison between SYLPH and the Random SVO Assignment at Each Step variant, as the latter failed to converge during training, making performance comparison between them does not make sense. For the same reason, we also omit its results from the training curves. The p-values p obtained from the paired t-tests were much lower than the conventional significance threshold (0.05, 0.01, or 0.001), indicating a statistically significant difference in performance favoring SYLPH.

The findings clearly demonstrate that while randomly assigned SVOs might achieve task completion in less structured and sparser scenarios, they lack the scalability and robustness required for handling complex, highly structured environments. Thus, the study conclusively validates the effectiveness of the learned SVO policy within the SYLPH framework, highlighting its critical role in advancing the capabilities of learning-based MAPF solutions.

6.4. MAPF ablation experiments

The core idea of our architecture lies in integrating SVO as a temporally extended variable within the MAPF framework. This integration empowers agents with diverse social preferences, equipping them to navigate and resolve the symmetrical challenges typically presented by social dilemmas. To further enhance our framework, we replaced the LSTM component with a Semantic Transformer (ST), thereby augmenting the agent's spatial inference capabilities. Additionally, we developed a learning-based tie-breaking mechanism, offering a more efficient planner in densely populated environments compared to general post-processing approaches.

To evaluate the impact of these key elements, we tested four SYLPH variants across different map types (random, room-like, and maze maps): 1) The complete SYLPH model incorporating all mentioned components; 2) SYLPH minus the learning-based tie-breaking mechanism (SYLPH w/o tb), retaining SVO as a temporally extended variable; 3) SYLPH stripped of all SVO-related components, including partner selection, SMP3O, and communication block; 4) based on (3), the semantic transformer replaced by LSTM. All ablation studies took place in environments whose size is a random number sampled from $(10, 40)$ with 8 agents. A uniform maximum time limit was set for each episode, capped at 256 steps. All models are trained to 20M steps. The performance enhancements brought by each element were quantified through experiments, with results showcased in Fig. 11. We utilized Episode Length - the timesteps

[6] The training curves can be found at Appendix C.1.
[7] The results can be found at Appendix C.2.

Table 3
More Performance of Ablation Study.

Map Type	Semantic Transformer	Social Behavior	Tie-Breaking	External Reward↑	Blocked Agents↓	Goals Reached ↑
Random Map	–	–	–	-57.490	6.904	7.924
	✓	–	–	-51.462	2.751	7.962
	✓	✓	–	-46.926	0.5562	7.986
	✓	✓	✓	**-39.110**	**0.3345**	**7.999**
Room-like Map	–	–	–	-85.358	3.133	7.87
	✓	–	–	-81.154	2.100	7.905
	✓	✓	–	-74.443	0.4126	7.932
	✓	✓	✓	**-61.210**	**0.2183**	**7.993**
Maze Map	–	–	–	-158.168	27.168	7.382
	✓	–	–	-126.976	14.451	7.557
	✓	✓	–	-90.484	1.709	7.836
	✓	✓	✓	**-69.69**	**0.5189**	**7.977**

required for all agents to achieve their goals - as the performance metric. Notably, across the different map types, incremental additions of SYLPH components yielded significant performance improvements:

- Random Map: Transforming from PRIMAL to PRIMAL w/ Semantic Transformer (ST) reduced episode length by 13.86%. Incorporating SVO decreased episode costs by 29.67% compared with PRIMAL. After the inclusion of the tie-breaking mechanism, SYLPH's episode length is 38.93% lower than PRIMAL.
- Room-like Map: Here, the episode length saw a reduction of 4.71% with PRIMAL w/ ST, 15.93% with SYLPH w/o tb, and 33.02% with the full SYLPH model, all compared to the baseline PRIMAL performance.
- Maze Map: This environment highlighted the most pronounced improvements: 16.35% (PRIMAL w/ ST), 43.13% (SYLPH w/o tb), and 58.88% (SYLPH), showcasing the framework's efficacy in addressing social dilemmas, particularly prevalent due to the maze's long corridors.

These findings underscore the substantial benefits of each component, especially in complex environments like maze maps where social dilemmas always appear. The progression from PRIMAL to the fully-fledged SYLPH model demonstrates the significant role of spatial inference enhancement, SVO diversification, and the learning-based tie-breaking mechanism in elevating model performance across varied and challenging MAPF scenarios.

Table 3 demonstrates the differences among these SYLPH variants across additional performance metrics. Specifically, we introduce three more metrics: external reward, blocked agents,[8] and goals reached. These metrics collectively offer a comprehensive view of each system's efficiency and cooperative capabilities under the constraints of the specified experimental conditions.

- External Reward: This metric quantifies the cumulative reward that an agent receives from the environment over the course of an episode, exclusive of any adjustments or redistributions. A higher external reward is indicative of superior agent performance from RL aspect, reflecting successful task execution within the environment.
- Blocked Agents: This metric measures the number of agents whose paths were obstructed due to the behavior of other agents. It includes agents whose optimal paths are significantly extended because of another agent's presence. A lower value for this metric suggests a policy with stronger spatial awareness and a better ability to reason about the behavior of other agents to preempt collisions/delay.
- Goal Reached: Representing the count of agents (with a maximum possible count of 8) that successfully reach their designated goals within the specified maximum steps (256 in this study), this metric directly reflects the effectiveness of the agents' pathfinding capabilities.

Table 3 illustrates SYLPH's significant improvements across all three metrics in comparison to other variants, across diverse map types. It is worth noting that the reduction in blocked agents highlights SYLPH's advanced planning and social behavior integration. Through the incorporation of social preferences and neighbor selection algorithm, agents are endowed with long-horizon coordination abilities, enabling them to proactively accommodate the movements of other agents and effectively prevent potential conflicts. This proactive coordination behavior is a major focal point of SYLPH. Unlike reactive collision avoidance policy that adjust behaviors based on immediate dilemmas, SYLPH enables agents to formulate and execute a more sophisticated, forward-looking policies. This approach mitigates social dilemmas and conflicts, showcasing the framework's capacity for advanced, anticipatory coordination.

6.5. *Communication range restricted variants*

In this section, we evaluate the performance of the Communication Range Restricted Variants introduced in Section 4.1.2. We consider two communication ranges, 4 and 9, with the partner update threshold $\xi = 1$. These variants are denoted as SYLPH w/ cr = 4

[8] A detailed explanation about this metric can be found at Appendix E.

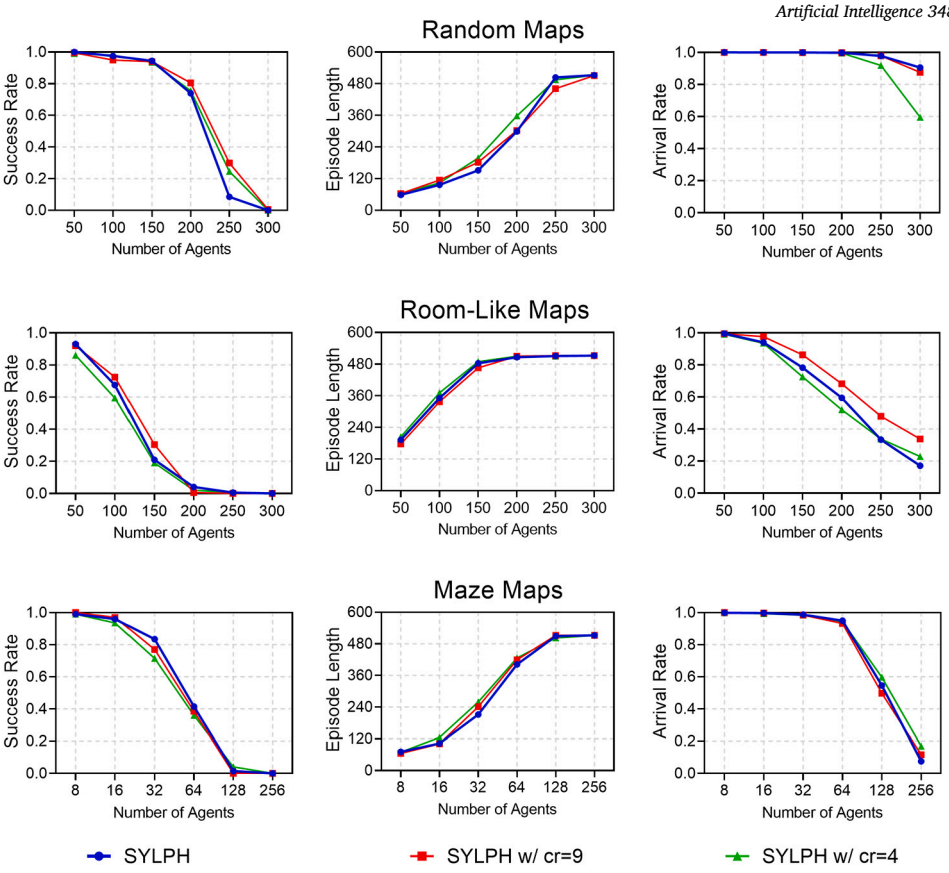


Fig. 12. Performance of SYLPH and its variants with different communication ranges across various configurations in random, room-like, and maze maps.

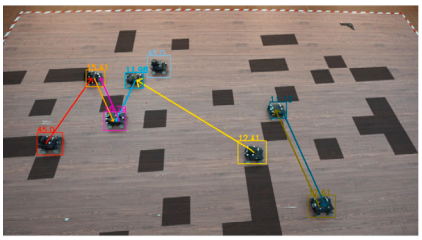
and SYLPH w/ cr = 9, respectively. The objective of comparison is to analyze the impact of global vs. decentralized partner selection across different structural level of maps. Experiments were conducted in the configuration described in Section 6.3, and the results are shown in Fig. 12.

From the results, we can see that the results are consistent with the intuition. In random maps, global partner selection does not provide a significant advantage. Instead, both SYLPH w/ cr = 4 and SYLPH w/ cr = 9 achieve a higher success rate than vanilla SYLPH. This is because random maps lack highly structured components, reducing concerns about potential deadlocks caused by distant agents. Effective coordination with nearby agents is often sufficient to reach the goal. In contrast, in maze maps, the success rates of vanilla SYLPH, SYLPH w/ cr = 9, and SYLPH w/ cr = 4 exhibit a decreasing trend. This is due to the components of highly structured environments, such as long corridors, where agents must anticipate and coordinate with other agents positioned at the far end of the corridor. A broader partner selection range allows agents to prioritize those most relevant for overcoming such structures. Additionally, unlike random maps, where agent interactions are relatively uniform, maze maps exhibit high variance in agent influence, requiring agents to make more precise partner selections. The room-like maps lie between random maps and maze maps in terms of structure. As a result, SYLPH w/ cr = 9 outperforms vanilla SYLPH, which in turn outperforms SYLPH w/ cr = 4. This suggests that the moderate presence of structured components leads to moderate variance in agent importance, making a mid-range communication radius more effective for coordination.

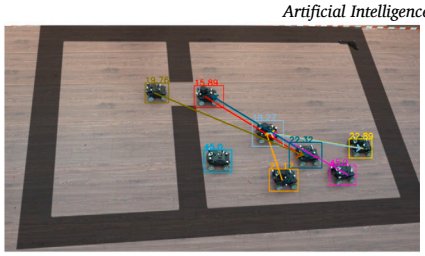
Our findings suggest that when the variance in agent importance is low, agents closer to the ego agent tend to be more crucial for decision-making, making a smaller communication range effective. However, as map structure becomes more complex, the variance in agent importance increases significantly. In such cases, a larger communication range enables agents to accurately identify and prioritize those with the greatest influence on decision-making, leading to better coordination and higher success rates.

6.6. Experimental validation

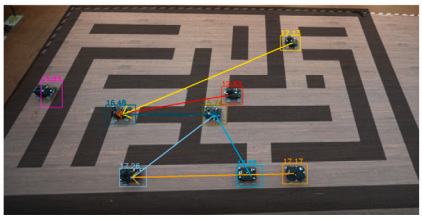
Fig. 13 showcases our implementation of SYLPH with a team of 8 real robots pathfinding through a random map, a room-like map, and a maze map. For these real world pathfinding tasks, we used the standard SYLPH model trained on gridworlds without any additional finetuning/training on-robot. To satisfy our algorithm's assumption that all agents' positions are always perfectly known, we used the *Optitrack Motion Capture System* for precise localization of real robots. We equipped the robots with Mecanum wheels to enable movement in the four cardinal directions. However, disturbances and control inaccuracies can cause deviations from the planned path. To address these issues, we employed an *Action Dependency Graph* (ADG), as proposed by [76], which introduces a



(a) random map



(b) room-like map



(c) maze map

Fig. 13. Experiments with real robots on random map, room-like map, and maze map. In these figures, the black areas represent obstacles, the directed arrows indicate the partner selected by the agent, and the numbers denote the agent's current SVO (ranging from 0 to 45 degrees).

precedence order for agents occupying a cell to prevent execution errors from propagating and disrupting the overall plan. Since our deployment relies on the support of ADG, we require centralized computation to generate the complete paths during deployment, which are then used for distributed execution by the agents.

Our experiments demonstrated that agents could reach their goals quickly and without collisions, with the ADG effectively eliminating execution errors. This highlights the potential of our method for real-world applications. Additional details about our experiment can be found in Appendix D, and the full video is available in the supplementary materials.

7. Conclusion

This paper introduces SYLPH, a socially-aware learning-based MAPF framework designed to address potential social dilemmas by encouraging agents to learn social behaviors. To these ends, we introduce the social value orientation (SVO) as a learnable dynamic choice for agents, to help them make decisions that benefit the group by coupling their individual interests with those of their partners. The resulting different social preferences can break homogeneity in the team, helping couple agents directly in reward space to favor coordinated maneuvers, thereby improving cooperation among agents. With explicit social preferences and advanced communication learning mechanism, agents are able to more effectively reason about each other's behavior, as evidenced by the significant reduction in instances where agents blocked each other in the experiments. Through extensive testing across various map types and agent densities, we showed that SYLPH consistently outperforms other learning-based frameworks like SCRIMP and DCC and, in some scenarios, matches the performance of traditional methods currently considered state-of-the-art. Our framework pushes the performance boundaries of current learning-based MAPF planners and demonstrates that equipping agents with intelligent social behaviors can effectively resolve prevalent social dilemmas in MAPF problems.

Looking ahead, we plan to further refine our learning-based MAPF framework. Under our current training setting, SYLPH's social preference learning enables agents to achieve 100% success rate in various random environments during training. However, some unsolvable edge cases still arise in highly structured room-like and maze maps. In our future works, we will analyze these cases individually, identify their commonalities, and develop mechanisms that train agents to effectively resolve them, thus enhancing overall performance. Additionally, the interpretability of agent behavior is another key area of interest for our future works. We aim to establish a behavior prediction mechanism for agents by further promoting more stable SVO choices. We envision that this advantage may help the team create predictable and interpretable plans, which could be crucial for planning in mixed environments with humans. That is, we envision that SVO may not only help humans understand the intentions of autonomous robots, but may also better incorporate humans in such shared environments by analyzing peoples' social preferences through the use of inverse reinforcement learning, towards improved robotic deployments in populated areas.

CRediT authorship contribution statement

Chengyang He: Writing – review & editing, Writing – original draft, Validation, Project administration, Methodology, Investigation, Conceptualization. **Tanishq Duhan:** Visualization, Methodology. **Parth Tulsyan:** Writing – review & editing, Visualization. **Patrick Kim:** Writing – review & editing, Visualization. **Guillaume Sartoretti:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by the Singapore Ministry of Education (MOE), as well as by an Amazon Research Award.

Appendix A. Implementation details

A.1. Hyperparameters

Table A.4 below presents the hyperparameters used to train the SYLPH model.

A.2. RL reward structure

The reward structure for each agent's step is defined in A.5 below. Similar to our previous works [18,41], agents are penalized at each timestep unless they are on goal to promote faster episode completion. The episode terminates if all agents are on goal at the end of a timestep, or when the number of steps exceeds a pre-defined limit (256 for our model as given in A.4).

The blocking penalty means that if an agent occupies a space that prevents other agents from reaching their goals or significantly lengthens their optimal paths, a penalty is incurred proportional to the number of blockings caused. The specific penalty received by an agent depends on the number of fellow agents it blocks c. The penalty can be represented as $-1 \cdot c$, which have been used in Eq. (B.1). This definition ensures that penalties are not only punitive but also proportionate to the level of inconvenience caused, encouraging agents to consider the broader implications of their decisions on system efficiency and collective goal attainment.

A.3. Baseline implementation sources

In this section, we provide the source code for the baseline models used in the comparison experiments presented in Section 6.3.2:

- SCRIMP: https://github.com/marmotlab/SCRIMP
- DCC: https://github.com/ZiyuanMa/DCC
- EECBS: https://github.com/Jiaoyang-Li/EECBS
- PIBT: https://github.com/Kei18/pypibt
- LNS2: https://github.com/Jiaoyang-Li/MAPF-LNS2
- LaCAM-py: https://github.com/Kei18/py-lacam
- LaCAM3: https://github.com/Kei18/lacam3

Table A.4
Hyperparameters table.

Hyperparameter	Value	Hyperparameter	Value
Number of agents	8	Value coefficient	0.08
Number of SVOs	5	Policy coefficient	10
Maximum episode length	256	Valid coefficient	0.5
FOV size	9	Blocking coefficient	0.5
FOV heuristic	5	Number of epochs	10
World size	(10, 40)	Number of processes	16
Obstacle density	(0.0, 0.3)	Maximum number of timesteps	2e7
Overlap decay	0.95	Minibatch size	16
SVO importance factor	2	Imitation learning rate	0
Learning rate	1e-5	Net size	512
Discount factor	0.95	SVO channel size	512
Gae lamda	0.95	Number of observation channels	9
Clip parameter for probability ratio ϵ	0.2	Goal representation size	12
Gradient clip norm	10	Goal vector length	4
Entropy coefficient	0.01	Number of semantic tokens, L	16

Table A.5
Reward Structure.

Action	Reward
Move (up/down/left/right)	-0.3
Stay (off goal)	-0.3
Stay (on goal)	0.0
Collision	-2
Block	-1

Appendix B. Claim and proof

Let there be an arbitrary value a and b selected from a finite set, and an arbitrary value c such that:

$$\begin{aligned} &a \in \{-c, -2, -0.3, 0\}; \\ &b \in \{-c, -2, -0.3, 0\}; \\ &c \in \mathbb{R} \ \text{ and } \ c \geq 1; \end{aligned} \tag{B.1}$$

where a and b mean the ego agent and its partner's external rewards, R_i^{ex} and R_p^{ex}. According to Table A.5, we can figure out that -0.3 is the moving and staying idle (off goal) cost; 0 is the penalty for agent standing on its goal; -2 denotes the collision penalty; and $-c$ is the blocking reward. Furthermore, we define a function $f(x)$ as:

$$f(x) = a \cdot \cos x + b \cdot \sin x, \quad x \in [0°, 45°]; \tag{B.2}$$

Remark. *The function $f(x)$ models the equation of R_i^a defined in Eq. (4).*

Theorem. *The function $f(x)$ is monotonically non-increasing x for any valid values of a and b.*

Proof. For the sake of contradiction, assume that the value of the function $f(x)$ is monotonically increasing. In other words, assume that x is increasing from 0° to 45° for any value a and b. Then, it must be true that:

$$f'(x) = -a \cdot \sin x + b \cdot \cos x > 0 \tag{B.3}$$

Considering the domain of x and the permissible values of a and b, it must be true that:

$$\begin{aligned} -a \cdot \sin x &\geq 0 \\ b \cdot \cos x &\leq 0 \end{aligned} \tag{B.4}$$

Hence, for the assumption to hold:

$$-a \cdot \sin x > b \cdot \cos x \tag{B.5}$$

Rearranging the inequality,

$$\begin{aligned} -a \cdot \frac{\sin x}{\cos x} &> b \\ -\tan x &> \frac{b}{a} \end{aligned} \tag{B.6}$$

But for given domain of x,

$$-\tan x \leq 0 \tag{B.7}$$

If the above is true, it must be true that:

$$\begin{aligned} 0 \geq -\tan x &> \frac{b}{a} \\ \therefore 0 &> \frac{b}{a} \end{aligned} \tag{B.8}$$

Which is impossible from the possible values of a and b as they are both less than or equal to 0. Therefore, the assumption is false and the function $f(x)$ is not monotonically increasing.

By contradiction, it must be true that the function $f(x)$ is monotonically non-increasing. In other words, the function $f(x)$ is consistent or decreasing when x is increasing from 0° to 45° for any valid value of a and b. □

Appendix C. Random SVO results

C.1. Training results

The experiments in this section is to verify the effectiveness and efficiency of the trained SVO policy provided by SYLPH. To clarify the efficiency of SYLPH, we assigned random SVOs to all agents in the comparative experiment, which also served to enhance population diversity. Specifically, three comparative experiments were conducted to update the agents' SVO at different frequencies: 1) the SVO is set at the beginning of the episode and remains unchanged throughout, 2) the SVO remains unchanged until the agent switches partners, and 3) the SVO is randomly reset at each step.

The outcomes of these experimental setups are depicted in Fig. C.14. Notably, the policy of resetting the SVO at each step resulted in confusion among agents about their SVO policy, preventing the training from converging to an effective action policy. This indicates

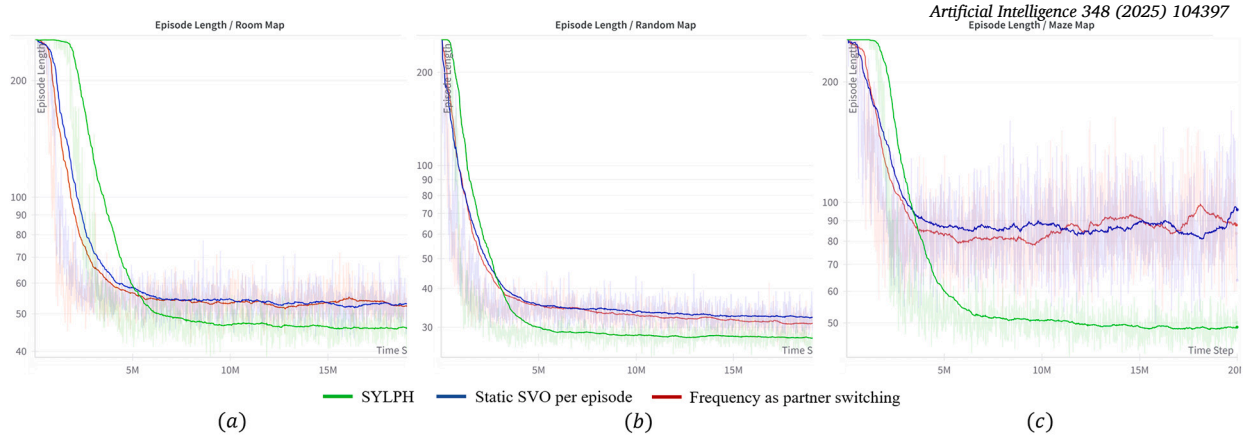


Fig. C.14. Training curves for SYLPH and two random SVO assignment which updates with varying frequencies across room-like maps, random maps, and maze maps.

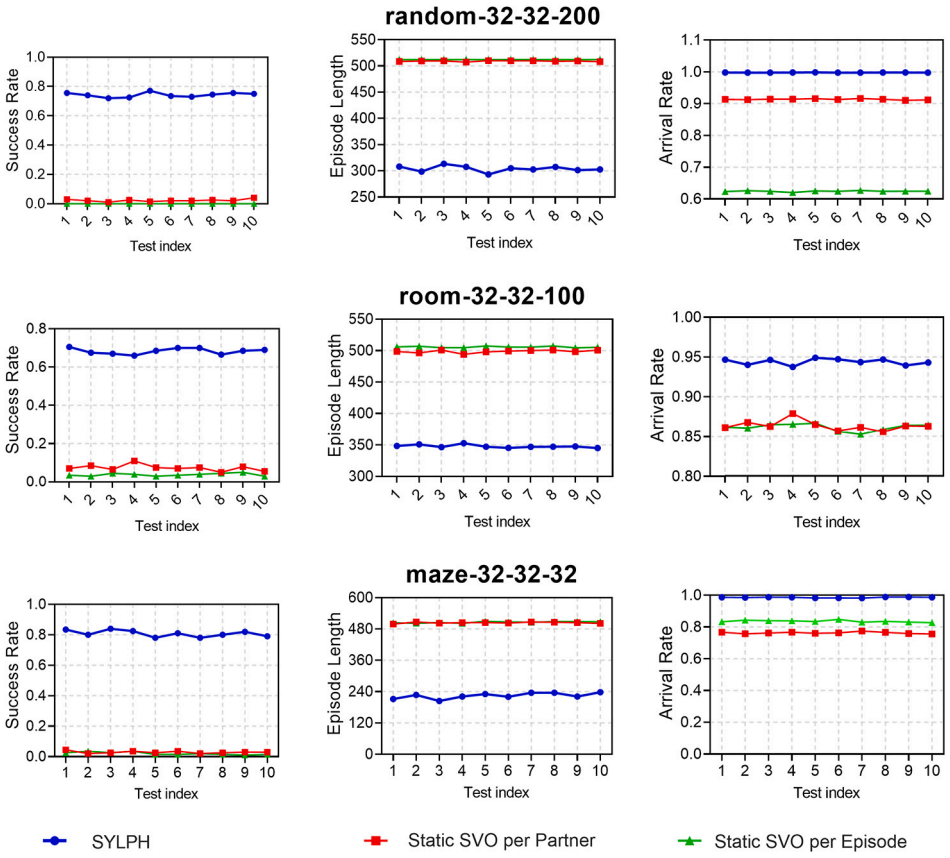


Fig. C.15. Comparison of the results of SYLPH and variants of its whose random SVO with different update frequencies, run 10 times on three different types of test sets. The results are used to calculate the p-value of the paired-t test in Section 6.3.3.

the disruptive impact of high-frequency random SVO changes on agent behavior and decision-making. In contrast, the other two methods of random SVO assignment allowed agents to learn effective action policies, although performance metrics slightly lower than those achieved by the SVO policy specifically learned by SYLPH in random and room-like maps. This slight difference in performance might be attributed to the limited number of agents (only 8) involved in the training, which constrains the extent and variety of potential social dilemmas and conflicts within these less complex environments. Consequently, the added value of adaptive social behaviors in these settings is somewhat restricted. However, a different trend was observed in the training outcomes on maze maps, which are characterized by high obstacle density and highly structured obstacle distribution. Here, even with just eight agents, the structured environment teems with numerous social dilemmas and conflicts. Under these conditions, the SVO policy implemented by SYLPH demonstrated clear advantages, leading to significant performance improvements.

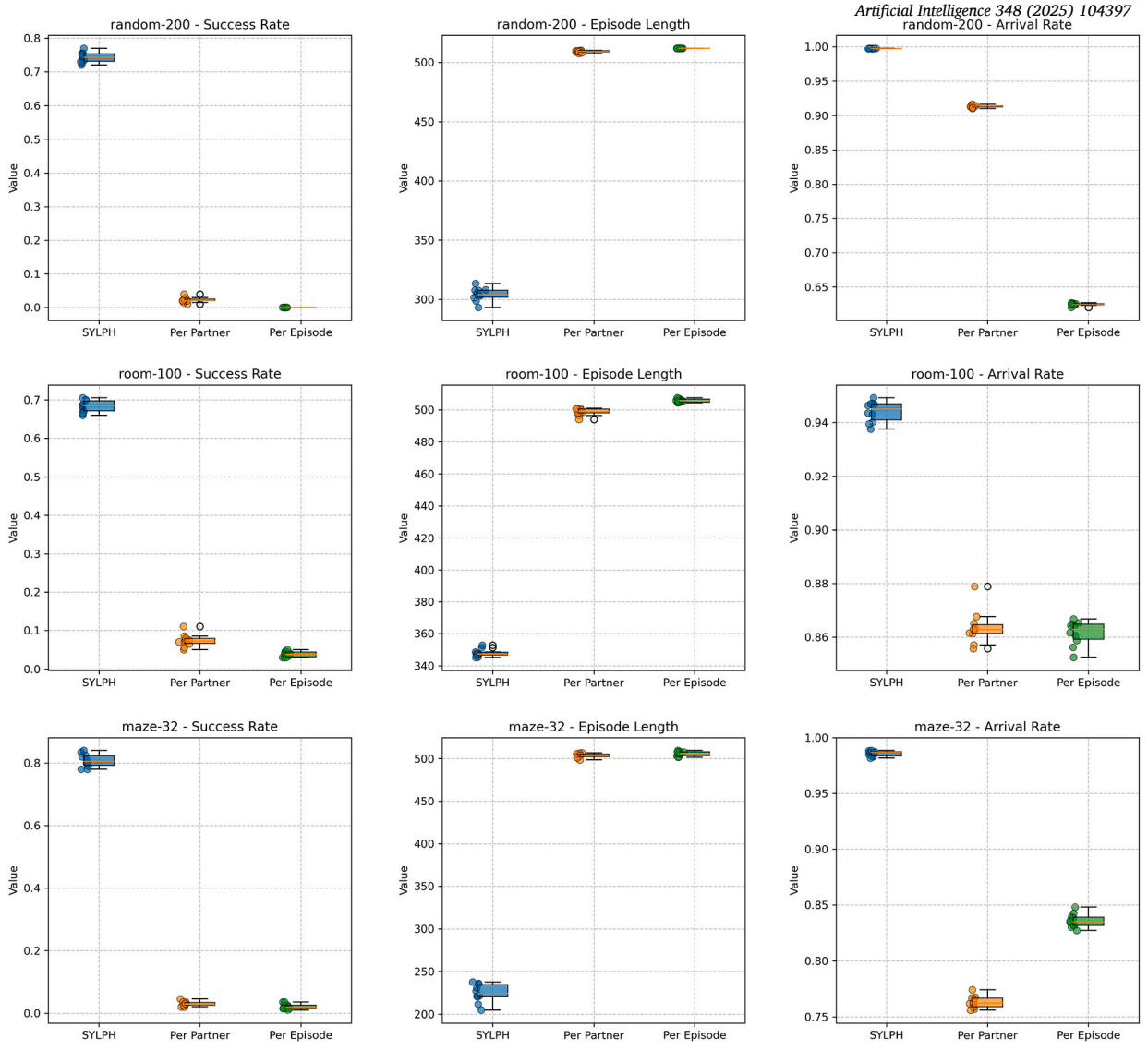


Fig. C.16. A mustache plot showing the statistical results of SYLPH and its two variants, Static SVO per Partner and Static SVO per Episode, each run 10 times on *random-32-32-0.2-200*, *room-32-32-0.3-100*, and *maze-32-32-0.5-32*.

The experiments validate the superiority of the SVO policy trained under the SYLPH framework over randomly assigned SVO polices.

C.2. Execution results

In this section, we evaluate two random SVO update methods, *Static SVO Per Episode* and *Static SVO Per Partner*, by running them 10 times in three environments: *random-32-32-0.2-200*, *room-32-32-0.3-100*, and *maze-32-32-0.5-32*. The results are presented in Fig. C.15.

We compute the average success rates of *Static SVO Per Episode*, *Static SVO Per Partner*, and SYLPH across 10 runs and calculate their p-values to determine whether there is a significant difference between the random SVO variants and SYLPH. As shown in Fig. C.14 and Fig. C.15, while both random SVO update methods achieve acceptable performance during training, they exhibit poor scalability in denser test environments. Additionally, we show a mustache plot (Fig. C.16) here to illustrate the performance gaps of these variants compared to vanilla SYLPH.

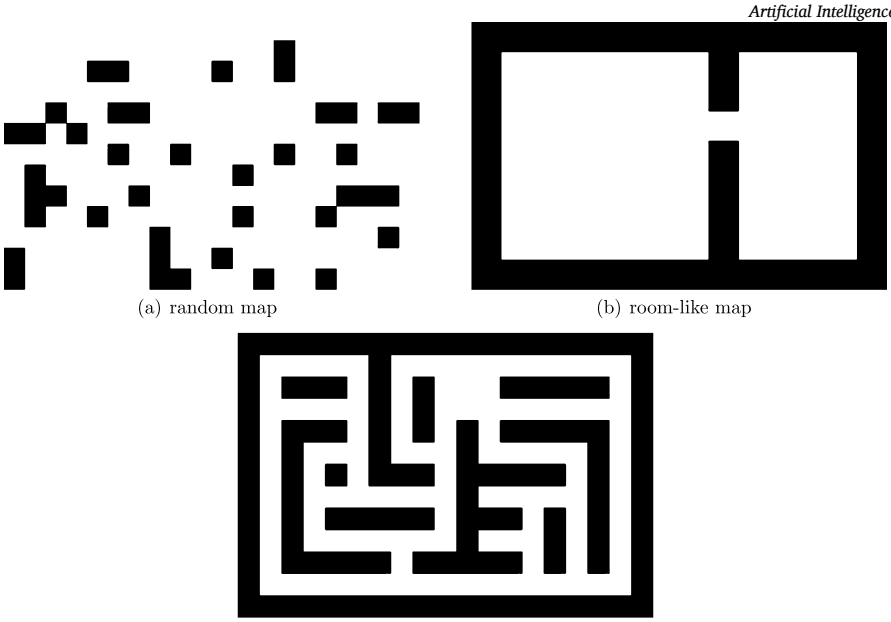


Fig. D.17. Maps for real robot experiments.

Appendix D. Engineering deployment

D.1. Setup

Fig. D.17 illustrates the random map, room-like map, and maze map used in our experiment. When mapped to the real world, each cell has a side length of $0.3m$, which is slightly larger than the size of the agent to ensure that the agent occupies only one cell on the map. We utilized 8 robots equipped with Mecanum wheels, each robot measuring approximately $0.23m \times 0.2m$. We used the *OptiTrack Motion Capture System* to get the accurate positions of these 8 robots. The configuration of the agents' starting and goal positions was randomly generated. In the experiment, the robots were aware of the virtual positions of obstacles and were programmed to avoid these areas. However, the real environment did not contain physical obstacles which can prevent interference with the line of sight of the *OptiTrack* motion capture system.

D.2. Action dependency graph

SYLPH generates paths assuming each agent operates perfectly at every step in a discrete map. However, due to the imperfect nature of robots and the continuous environment, directly executing these planned paths in the real world is impractical. For example, a planner might instruct agent `A` to move to agent `B`'s current position while agent `B` moves to a new cell. Executing this plan directly could result in collisions due to localization errors, delays in motor control, or differences in velocities.

To prevent such issues and ensure the feasibility of our joint set of actions, we adopt the method proposed by [76] by constructing an *Action Dependency Graph* (ADG). The ADG establishes a precedence order for agents occupying a cell, meaning that faster-moving agents will wait for others if the planned path requires them to occupy the cell afterward. This mechanism ensures that execution errors do not propagate and disrupt the overall plan. In the above example, the ADG ensures that agent `A` will wait for `B` to vacate its current cell before moving in, thus preserving the integrity of the planned path. Such a mechanism ensures that the planned path is executed securely between agents, though it may introduce slight delays.

D.3. Execution

In order to deploy the model, we use a centralized solver but a distributed execution method to validate the result of SYLPH. We run SYLPH on a centralized desktop with access to both the current and goal positions of all robots, enabling it to compute the complete paths each agent must follow to reach its destination. After calculation, the central desktop iterates through all agents, their actions into *Task* objects and constructs the ADG from these tasks, with each node/task of the ADG having the following attributes:

```
object Task {
    taskID;       // Unique identifier for the task
    robotID;      // ID of the robot assigned to the task
    action;       // Action to be performed
```

```
    startPos;     // Initial position of the robot
    endPos;       // Position after action completion
    time;         // Scheduled time for the action
    dependencies; // All inter-robot Tasks that need to be
                   completed before this task can be enqueued
    status;       // Current status: staged, enqueued, or
                   completed
};
```

Each task can have one of three statuses: *STAGED*, *ENQUEUED*, or *DONE*. Initially, tasks are set to *STAGED*. When all *dependencies* of the task are completed, the status changes to *ENQUEUED*, meaning readiness for execution. The task status updates to *DONE* once the agent reaches the specified *endPos*.

At the ROS execution level, a *central node* is responsible for generating the ADG, as well as distributing tasks to robots. Each robot is equipped with a *robot node*, which handles receiving tasks from the central node, extracting goals from these tasks, uses a PID controller to reach those goals, and global communication to publish the completion of their goals. Once the ADG is computed by the *central node*, it assigns each robot its respective tasks. Since each task includes not only the start and goal positions but also task dependencies, the robot can execute its path in a fully distributed manner, relying solely on global communication to publish task completions. When a robot completes a task, it broadcasts this completion via global communication, allowing all robots that depend on this task to update its status to *DONE*. As outlined in the ADG paper, before enqueuing a task, the *robot node* verifies that the previous task is either *ENQUEUED* or *DONE* and that all dependent tasks are marked as *DONE*.

This approach not only ensures that robots wait for delayed agents when necessary but also grants them the flexibility to complete their tasks without interruption if no dependencies exist. As a result, the execution burden is effectively distributed after the ADG is computed by the central node.

Appendix E. Details of the 'blocked agents' metric

We consider that the ego agent blocks other agents if either of the following two conditions occurs (see Algorithm 4 for details):

- The ego agent selects the stay idle action, and if treated as an obstacle, it prevents another agent from reaching its goal.
- The ego agent selects the stay idle action, and if treated as an obstacle, it significantly increases the optimal path length required for another agent to reach its goal.

Algorithm 4: Count the number of blocked agents.

Input: Ego agent's position p_i; ego agent's action a_i; other agent's position p_j; other agent's goal g_j; map $\mathcal{G}$
Output: Number of blocked agents by ego agent.

Set the cell of p_i as obstacle in map $\mathcal{G}$ to build $\mathcal{G}'$;
Initialize the number of blocked agents $n_b = 0$;
if *ego agent's action* a_i *is stay idle* **then**
  foreach *agent* j, *where* $j \neq i$ **do**
    if $\exists P_j = A^*(G, p_j, g_j) \wedge \nexists P'_j = A^*(G', p_j, g_j)$ **then**
      $n_b = n_b + 1$;
    else if $\exists P_j \wedge len(P'_j) > len(P_j) + \epsilon$ **then**
      $n_b = n_b + 1$;
    else
      Pass
    end
  end
end

Data availability

Our full training/testing code is available from the url at the end of our abstract.

References

[1] R. Stern, N.R. Sturtevant, A. Felner, S. Koenig, H. Ma, T.T. Walker, J. Li, D. Atzmon, L. Cohen, T.S. Kumar, et al., Multi-agent pathfinding: definitions, variants, and benchmarks, in: Twelfth Annual Symposium on Combinatorial Search, 2019.
[2] J. Li, A. Tinka, S. Kiesel, J.W. Durham, T.S. Kumar, S. Koenig, Lifelong multi-agent path finding in large-scale warehouses, Proc. AAAI Conf. Artif. Intell. 35 (2021) 11272–11281.
[3] B. Wang, Z. Liu, Q. Li, A. Prorok, Mobile robot path planning in dynamic environments through globally guided reinforcement learning, IEEE Robot. Autom. Lett. 5 (2020) 6932–6939.
[4] S. Polydorou, A Learning-Based Approach for Distributed Planning and Coordination of Airport Surface Movement Operations, 2021.
[5] J. Li, E. Lin, H.L. Vu, S. Koenig, et al., Intersection coordination with priority-based search for autonomous vehicles, Proc. AAAI Conf. Artif. Intell. 37 (2023) 11578–11585.

[6] H. Ma, J. Yang, L. Cohen, T. Kumar, S. Koenig, Feasibility study: moving non-homogeneous teams in congested video game environments, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 13, 2017, pp. 270–272.
[7] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T.S. Kumar, S. Koenig, H. Choset, Primal: pathfinding via reinforcement and imitation multi-agent learning, IEEE Robot. Autom. Lett. 4 (2019) 2378–2385.
[8] Q. Lin, H. Ma, Sacha: soft actor-critic with heuristic-based attention for partially observable multi-agent path finding, IEEE Robot. Autom. Lett. (2023).
[9] Z. Yan, C. Wu, Neural neighborhood search for multi-agent path finding, in: The Twelfth International Conference on Learning Representations, 2024.
[10] C. Ferner, G. Wagner, H. Choset, Odrm* optimal multirobot path planning in low dimensional search spaces, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 3854–3859.
[11] H. Ma, D. Harabor, P.J. Stuckey, J. Li, S. Koenig, Searching with consistent prioritization for multi-agent path finding, Proc. AAAI Conf. Artif. Intell. 33 (2019) 7643–7650.
[12] M. Tan, Multi-agent reinforcement learning: independent vs. cooperative agents, in: Proceedings of the Tenth International Conference on Machine Learning, 1993, pp. 330–337.
[13] M. Damani, Z. Luo, E. Wenzel, G. Sartoretti, Primal _2: pathfinding via reinforcement and imitation multi-agent learning-lifelong, IEEE Robot. Autom. Lett. 6 (2021) 2666–2673.
[14] H. Guan, Y. Gao, M. Zhao, Y. Yang, F. Deng, T.L. Lam, Ab-mapper: attention and bicnet based multi-agent path planning for dynamic environment, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2022, pp. 13799–13806.
[15] J. Li, D. Harabor, P.J. Stuckey, H. Ma, G. Gange, S. Koenig, Pairwise symmetry reasoning for multi-agent path finding search, Artif. Intell. 301 (2021) 103574.
[16] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, D. Rus, Social behavior for autonomous vehicles, Proc. Natl. Acad. Sci. USA 116 (2019) 24972–24978.
[17] R. Veerapaneni, Q. Wang, K. Ren, A. Jakobsson, J. Li, M. Likhachev, Improving learnt local mapf policies with heuristic search, in: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 34, 2024, pp. 597–606.
[18] Y. Wang, B. Xiang, S. Huang, G. Sartoretti, Scrimp: scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding, arXiv preprint arXiv:2303.00605, 2023.
[19] J. Gao, Y. Li, X. Li, K. Yan, K. Lin, X. Wu, A review of graph-based multi-agent pathfinding solvers: from classical to beyond classical, Knowl.-Based Syst. (2023) 111121.
[20] G. Wagner, H. Choset, M*: a complete multirobot path planning algorithm with performance bounds, in: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2011, pp. 3260–3267.
[21] G. Sharon, R. Stern, A. Felner, N.R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, Artif. Intell. 219 (2015) 40–66.
[22] S.-H. Chan, J. Li, G. Gange, D. Harabor, P.J. Stuckey, S. Koenig, Ecbs with flex distribution for bounded-suboptimal multi-agent path finding, in: Proceedings of the International Symposium on Combinatorial Search, vol. 12, 2021, pp. 159–161.
[23] J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, Addison-Wesley Longman Publishing Co., Inc., 1984.
[24] G. Wagner, H. Choset, Subdimensional expansion for multirobot path planning, Artif. Intell. 219 (2015) 1–24.
[25] M. Barer, G. Sharon, R. Stern, A. Felner, Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem, in: Proceedings of the International Symposium on Combinatorial Search, vol. 5, 2014, pp. 19–27.
[26] J. Li, W. Ruml, S. Koenig, Eecbs: a bounded-suboptimal search for multi-agent path finding, Proc. AAAI Conf. Artif. Intell. 35 (2021) 12353–12362.
[27] J. Li, Z. Chen, D. Harabor, P.J. Stuckey, S. Koenig, Anytime multi-agent path finding via large neighborhood search, in: International Joint Conference on Artificial Intelligence 2021, Association for the Advancement of Artificial Intelligence (AAAI), 2021, pp. 4127–4135.
[28] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: International Conference on Principles and Practice of Constraint Programming, Springer, 1998, pp. 417–431.
[29] J. Li, Z. Chen, D. Harabor, P.J. Stuckey, S. Koenig, Mapf-lns2: fast repairing for multi-agent path finding via large neighborhood search, Proc. AAAI Conf. Artif. Intell. 36 (2022) 10256–10265.
[30] D. Silver, Cooperative pathfinding, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 1, 2005, pp. 117–122.
[31] K. Okumura, M. Machida, X. Défago, Y. Tamura, Priority inheritance with backtracking for iterative multi-agent path finding, Artif. Intell. 310 (2022) 103752.
[32] K. Okumura, Lacam: search-based algorithm for quick multi-agent pathfinding, Proc. AAAI Conf. Artif. Intell. 37 (2023) 11655–11662.
[33] H. Tang, F. Berto, 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.
[34] Q. Li, F. Gama, A. Ribeiro, A. Prorok, Graph neural networks for decentralized multi-robot path planning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 11785–11792.
[35] Q. Li, W. Lin, Z. Liu, A. Prorok, Message-aware graph attention networks for large-scale multi-robot path planning, IEEE Robot. Autom. Lett. 6 (2021) 5533–5540.
[36] Z. Ma, Y. Luo, J. Pan, Learning selective communication for multi-agent path finding, IEEE Robot. Autom. Lett. 7 (2021) 1455–1462.
[37] W. Li, H. Chen, B. Jin, W. Tan, H. Zha, X. Wang, Multi-agent path finding with prioritized communication learning, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE, 2022, pp. 10695–10701.
[38] J. Chen, K. Gao, G. Li, K. He, Nagphormer: a tokenized graph transformer for node classification in large graphs, arXiv preprint arXiv:2206.04910, 2022.
[39] W. Kim, J. Park, Y. Sung, Communication in multi-agent reinforcement learning: intention sharing, in: International Conference on Learning Representations, 2020.
[40] Z. Ding, T. Huang, Z. Lu, Learning individually inferred communication for multi-agent cooperation, Adv. Neural Inf. Process. Syst. 33 (2020) 22069–22079.
[41] C. He, T. Yang, T. Duhan, Y. Wang, 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. 14576–14582.
[42] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, D. Zhao, Mapper: multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2020, pp. 11748–11754.
[43] Z. Ma, Y. Luo, 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.
[44] L. Virmani, Z. Ren, S. Rathinam, H. Choset, Subdimensional expansion using attention-based learning for multi-agent path finding, arXiv preprint arXiv:2109.14695, 2021.
[45] H. Tang, F. Berto, Z. Ma, C. Hua, K. Ahn, J. Park, Himap: learning heuristics-informed policies for large-scale multi-agent pathfinding, arXiv preprint arXiv:2402.15546, 2024.
[46] R. Veerapaneni, A. Jakobsson, K. Ren, S. Kim, J. Li, M. Likhachev, Work smarter not harder: simple imitation learning with cs-pibt outperforms large scale imitation learning for mapf, arXiv preprint arXiv:2409.14491, 2024.
[47] A. Andreychuk, K. Yakovlev, A. Panov, A. Skrynnik, Mapf-gpt: imitation learning for multi-agent pathfinding at scale, Proc. AAAI Conf. Artif. Intell. 39 (2025) 23126–23134.
[48] C.G. McClintock, S.T. Allison, Social value orientation and helping behavior 1, J. Appl. Soc. Psychol. 19 (1989) 353–362.
[49] R.O. Murphy, K.A. Ackermann, M.J. Handgraaf, Measuring social value orientation, Judgm. Decis. Mak. 6 (2011) 771–781.
[50] D. Zhang, J. Xue, Y. Cui, Y. Wang, E. Liu, W. Jing, J. Chen, R. Xiong, Y. Wang, Zero-shot transfer learning of driving policy via socially adversarial traffic flow, arXiv preprint arXiv:2304.12821, 2023.

[51] Z. Dai, T. Zhou, K. Shao, D.H. Mguni, B. Wang, H. Jianye, Socially-attentive policy optimization in multi-agent self-driving system, in: Conference on Robot Learning, PMLR, 2023, pp. 946–955.
[52] K.R. McKee, I. Gemp, B. McWilliams, E.A. Duéñez-Guzmán, E. Hughes, J.Z. Leibo, Social diversity and social preferences in mixed-motive reinforcement learning, arXiv preprint arXiv:2002.02325, 2020.
[53] U. Madhushani, K.R. McKee, J.P. Agapiou, J.Z. Leibo, R. Everett, T. Anthony, E. Hughes, K. Tuyls, E.A. Duéñez-Guzmán, Heterogeneous social value orientation leads to meaningful diversity in sequential social dilemmas, arXiv preprint arXiv:2305.00768, 2023.
[54] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J.Z. Leibo, N. De Freitas, Social influence as intrinsic motivation for multi-agent deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 3040–3049.
[55] C. Li, T. Wang, C. Wu, Q. Zhao, J. Yang, C. Zhang, Celebrating diversity in shared multi-agent reinforcement learning, Adv. Neural Inf. Process. Syst. 34 (2021) 3991–4002.
[56] E. Hughes, J.Z. Leibo, M. Phillips, K. Tuyls, E. Dueñez-Guzman, A. García Castañeda, I. Dunning, T. Zhu, K. McKee, R. Koster, et al., Inequity aversion improves cooperation in intertemporal social dilemmas, Adv. Neural Inf. Process. Syst. 31 (2018).
[57] B. Eysenbach, A. Gupta, J. Ibarz, S. Levine, Diversity is all you need: learning skills without a reward function, arXiv preprint arXiv:1802.06070, 2018.
[58] A. Sharma, S. Gu, S. Levine, V. Kumar, K. Hausman, Dynamics-aware unsupervised discovery of skills, arXiv preprint arXiv:1907.01657, 2019.
[59] S. He, J. Shao, X. Ji, Skill discovery of coordination in multi-agent reinforcement learning, arXiv preprint arXiv:2006.04021, 2020.
[60] H. Ma, W. Hönig, T.S. Kumar, N. Ayanian, S. Koenig, Lifelong path planning with kinematic constraints for multi-agent pickup and delivery, Proc. AAAI Conf. Artif. Intell. 33 (2019) 7651–7658.
[61] A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, A. Panov, Learn to follow: decentralized lifelong multi-agent pathfinding via planning and learning, arXiv preprint arXiv:2310.01207, 2023.
[62] R. Chandra, R. Maligi, A. Anantula, J. Biswas, Socialmapf: optimal and efficient multi-agent path finding with strategic agents for social navigation, IEEE Robot. Autom. Lett. (2023).
[63] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, J. Wang, Mean field multi-agent reinforcement learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 5571–5580.
[64] S.D. Han, J. Yu, Optimizing space utilization for more effective multi-robot path planning, in: 2022 International Conference on Robotics and Automation (ICRA), IEEE, 2022, pp. 10709–10715.
[65] Z. Chen, D. Harabor, J. Li, P.J. Stuckey, Traffic flow optimisation for lifelong multi-agent path finding, Proc. AAAI Conf. Artif. Intell. 38 (2024) 20674–20682.
[66] R. Lowe, Y.I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Adv. Neural Inf. Process. Syst. 30 (2017).
[67] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347, 2017.
[68] J. Morag, R. Stern, A. Felner, Adapting to planning failures in lifelong multi-agent path finding, in: Proceedings of the International Symposium on Combinatorial Search, vol. 16, 2023, pp. 47–55.
[69] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.
[70] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907, 2016.
[71] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016).
[72] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, P. Vajda, Visual transformers: token-based image representation and processing for computer vision, arXiv preprint arXiv:2006.03677, 2020.
[73] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[74] M. Bettini, A. Shankar, A. Prorok, Heterogeneous multi-robot reinforcement learning, arXiv preprint arXiv:2301.07137, 2023.
[75] K. Okumura, Engineering lacam*: towards real-time, large-scale, and near-optimal multi-agent pathfinding, arXiv preprint arXiv:2308.04292, 2023.
[76] W. Hönig, S. Kiesel, A. Tinka, J.W. Durham, N. Ayanian, Persistent and robust execution of mapf schedules in warehouses, IEEE Robot. Autom. Lett. 4 (2019) 1125–1131.