Auton Agent Multi-Agent Syst (2026) 40:23
https://doi.org/10.1007/s10458-026-09747-7

Confidence-Based curricula for multi-agent path finding via reinforcement learning

Thomy Phan[1,2,3] · Joseph Driscoll[4] · Justin Romberg[4] · Sven Koenig[2,3,5]

Received: 10 November 2024 / Accepted: 13 April 2026

Abstract
A wide range of real-world applications can be formulated as *Multi-Agent Path Finding (MAPF)* problem, where the goal is to find collision-free paths for multiple agents with individual start and goal locations. State-of-the-art MAPF solvers are mainly centralized and rely on global information, which limits their scalability and flexibility when facing changes or new maps that require expensive replanning. *Multi-agent reinforcement learning (MARL)* offers an alternative approach to addressing MAPF problems by learning decentralized policies that generalize across a variety of maps. While there exist some prior works that attempt to connect both areas, the proposed techniques are heavily engineered and very complex due to the integration of many mechanisms that limit generality and are expensive to use. We argue that much simpler and more general approaches are needed to enable decentralized MAPF in a sustainable manner at significantly lower cost. In this paper, we propose *Confidence-based Auto-Curriculum for Team Update Stability (CACTUS)* as a lightweight MARL approach to decentralized MAPF. CACTUS defines a simple reverse curriculum scheme, where the goal of each agent is randomly placed within an allocation radius around the agent's start location. The allocation radius increases gradually as all agents improve, which is assessed by a confidence-based measure. In addition, we propose an extension called *Confidence- and Conflict-Based Curriculum Learning with Allocation Radius Adaptation* (C^3LARA), using weighted sampling of goal locations to improve conflict resolution in scenarios of high agent density. We provide a theoretical analysis of the strengths and limitations of CACTUS regarding exploration efficiency, optimality, and multi-agent coordination. We evaluate CACTUS and C^3LARA across various maps of different sizes, obstacle densities, and numbers of agents. Our experiments demonstrate better performance and generalization capabilities than state-of-the-art MARL approaches with less than 600,000 trainable parameters, which is less than 5% of the neural network size of current MARL approaches to decentralized MAPF.

Keywords Multi-Agent path finding · Multi-Agent reinforcement learning · Curriculum learning · Exploration · Multi-Agent credit assignment

Extended author information available on the last page of the article

Published online: 30 April 2026

1 Introduction

A wide range of real-world applications like goods transportation in warehouses [1, 2], smart manufacturing [3, 4], search and rescue missions [5], and traffic management [6–8] can be formulated as *Multi-Agent Path Finding (MAPF)* problem, where the goal is to find collision-free paths for multiple agents with individual start and goal locations. Finding optimal solutions or plans, i.e., a sequence of movement actions from start to goal for each agent, regarding the flowtime or makespan, is NP-hard [9, 10]. Despite the problem complexity, there exist a variety of MAPF solvers that find optimal [11, 12], bounded suboptimal [13, 14], or quick feasible but unbounded suboptimal plans [15, 16] by exploiting structural properties of the MAPF problem via heuristics [11, 13, 15, 17].

Most MAPF solvers use centralized search algorithms and rely on global information, which limits scalability and flexibility when facing changes or new maps that would need expensive replanning [18, 19]. The cost of replanning depends on the computational effort required by the employed search algorithm (w.r.t. the map size and number of agents) and the communication bandwidth required to gather global information about the environment and to distribute the new plans to all agents [20, 21]. Note that even if the computational effort were hypothetically zero, an adequate communication bandwidth is still required to gather global information and send out the plans in a timely manner, which is particularly expensive in large-scale scenarios [20, 22, 23]. Furthermore, the centralization of most search-based MAPF solvers limits their applicability to specific domains, e.g., confined applications like warehouse management [1, 2], which is prohibitive for partially observable and open domains, such as urban transportation or supply chains [24–26]. There exist distributed search approaches that can reduce the computational effort of replanning regarding the number of agents (but not the map size) [27–29]. However, they still require reliable and synchronized neighborhood and multi-round communication to coordinate effectively [28, 29].

Machine learning-based MAPF offers an alternative approach to addressing MAPF problems by learning decentralized agent policies rather than inflexible plans [20, 30]. The policies process histories of local observations to select actions, and are represented by machine learning models, such as deep neural networks with constant inference time, independent of the actual map size, as illustrated in Fig. 2. Such policies can make reactive decisions while generalizing to a variety of scenarios without explicit retraining or communication [19, 30, 31]. Therefore, machine learning-based MAPF is potentially more scalable and flexible than search-based MAPF solvers, being more suitable for partially observable and open domains [19, 20, 32].

We focus on *multi-agent reinforcement learning (MARL)* for decentralized MAPF, using trial-and-error policy learning [33, 34]. In contrast to supervised or imitation learning, which depends on expert data [19, 30], MARL learns from experience obtained via exploration, which mitigates any expert bias, e.g., from fast but unbounded suboptimal MAPF solvers [31, 32, 35], to optimize policies more effectively [36]. State-of-the-art MARL algorithms are based on *centralized training for decentralized execution (CTDE)*, where training takes place in a laboratory or a simulator with access to global information to learn coordinated and generalizing policies that can be executed independently under partial observability afterward [18, 37–39].

MAPF and MARL have been very active research areas in the last few years, with impressive advances on both sides, resulting in a variety of sophisticated algorithms [1, 13, 40, 41]. Despite these advances, both fields have been mainly studied independently of each other. However, MARL could benefit MAPF in various ways:

1. **Efficiency**: The learned policies are decentralized and reactive, therefore alleviating the computational and broadcast communication requirements of centralized MAPF solvers, especially in large-scale scenarios with many agents [39].
2. **Generalization**: The learned policies can generalize over a variety of maps and thus do not require complete retraining or replanning when being used on new maps [19, 20].
3. **Robustness**: The learned policies make decisions based on actual observations, thereby enabling them to react to local changes, i.e., emerging obstacles or new paths, without requiring the replanning of the whole system [3, 26].

On the other hand, MAPF poses an exciting challenge for MARL due to its practical relevance and the following structural aspects [12, 27]:

1. **Sparse Rewards**: MAPF represents a complex navigation problem, in which all agents are rewarded only for reaching their goals. Naive MARL would need exhaustive exploration to obtain informative data, which is time-consuming [42].
2. **Dynamic Constraints**: Agents are not allowed to collide, therefore having temporal constraints in addition to static constraints imposed by obstacles and boundaries of the underlying maps [27].
3. **Coordination**: MAPF requires the coordination of spatially close agents with potentially emergent effects like congestion or circulation [21]. So far, most MARL methods have focused only on coordination at a small scale, though [38, 43].

We believe that addressing MAPF via MARL can provide a fruitful research direction that would benefit both areas. While there are prior works that attempt to connect these areas, the proposed techniques are heavily engineered and very complex, using extremely large neural networks to represent policies, extensively shaped rewards, and centralized MAPF solvers for additional imitation learning [19, 44, 45]. The engineering efforts incorporate substantial human knowledge by focusing on very specific aspects of particular MAPF scenarios, such as congestion [44], blocking [19], communication [45], periodic neighborhood sensing [29], etc.

However, according to *The Bitter Lesson* by Richard Sutton, the Turing Award winner of 2025, *"the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation"* [46]. The lesson refers to a common trend across various fields of artificial intelligence, such as game control [36], natural language processing [47, 48], and computer vision [49, 50], where human knowledge-based engineering often yields short-term benefits, but is superseded by more general approaches due to less human bias (e.g., less heuristic dependencies and parameters) and better scaling w.r.t compute and data (e.g., higher efficiency and parallelization). Inspired by that lesson, we aim for a simpler and more general foundation to enable

MARL-based decentralized MAPF in a sustainable manner with significantly lower costs and without distracting ourselves with engineering details [20, 21].

In this paper, we propose *Confidence-based Auto-Curriculum for Team Update Stability (CACTUS)* as a lightweight MARL approach to decentralized MAPF. CACTUS defines a simple reverse curriculum scheme, where the goal of each agent is randomly placed within an allocation radius around the agent's start location. The allocation radius increases gradually as all agents improve, which is assessed by a confidence-based measure as shown in Fig. 1. Our contributions are as follows:

- We formulate the MAPF problem as a straightforward stochastic game with automatic collision prevention and sparse rewards to solve it in a black-box manner, which is more general and intuitive for standard MARL methods.
- Based on the stochastic game formulation, we propose a simple reverse curriculum scheme that gradually increases the potential distance between the start and goal locations to enhance state-of-the-art MARL techniques that would otherwise likely fail to learn meaningful policies. In addition, we propose an extension called *Confidence- and Conflict-Based Curriculum Learning with Allocation Radius Adaptation* (C^3LARA), using weighted sampling of goal locations to improve conflict resolution in scenarios of high agent density, without additional hyperparameters.
- We provide a theoretical analysis of the strengths and limitations of CACTUS regarding exploration efficiency, optimality, and multi-agent coordination.
- We evaluate CACTUS and C^3LARA across various maps of different sizes, obstacle densities, and numbers of agents. Our experiments demonstrate better performance and generalization capabilities than state-of-the-art MARL approaches with less than 600,000 trainable parameters, which is less than 5% of the neural network size of current MARL approaches to MAPF.

This paper is an extended and revised version of our prior work [51], which was presented at the *23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. We extend the paper with C^3LARA, as listed in the contributions, and theoretical insights about our approach w.r.t. exploration efficiency, optimality, multi-agent coordination, and further limitations. We also provide additional experimental results.

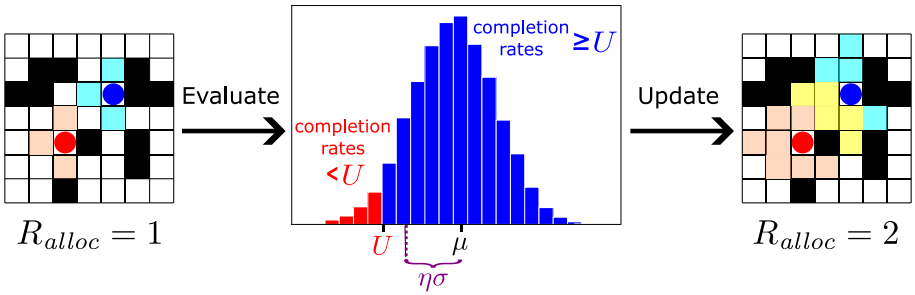

Fig. 1 Curriculum update scheme of CACTUS. The agents (colored circles) are trained and evaluated w.r.t. a goal allocation radius R_{alloc} (shaded areas around the agents). When the average completion rate μ exceeds a curriculum decision threshold U with a certain confidence level such that $\mu - \eta\sigma \geq U$, the allocation radius R_{alloc} is incremented by 1

2 Background

2.1 Multi-Agent Path Finding (MAPF)

We focus on *maps* as undirected unweighted *graphs* $G = \langle \mathcal{V}, \mathcal{E} \rangle$, where vertex set $\mathcal{V}$ contains all possible locations and edge set $\mathcal{E}$ contains all possible transitions or movements between adjacent locations $\{v, u\} \in \mathcal{E}$. The *degree* of a location $v \in \mathcal{V}$, i.e., the number of adjacent locations, is denoted by $\deg(v) = |\{u \in \mathcal{V} | \{v, u\} \in \mathcal{E}\}|$. An *instance* I consists of a map G and a set of *agents* $\mathcal{D} = \{1, ..., N\}$ with each agent $i \in \mathcal{D}$ having a *start location* $v_{\text{start},i} \in \mathcal{V}$ and a *goal location* $v_{\text{goal},i} \in \mathcal{V}$. We assume that $v_{\text{start},i} \neq v_{\text{start},j}$ and $v_{\text{goal},i} \neq v_{\text{goal},j}$ for any agent pair $i, j \in \mathcal{D}$ with $i \neq j$. At every time step t, all agents can move along the edges in $\mathcal{E}$ or wait at their current location $v_{t,i} \in \mathcal{V}$ [9].

MAPF aims to find collision-free plans for all agents. A *plan* $P = \{p_1, ..., p_N\}$ consists of individual paths $p_i = \langle p_{i,0}, ..., p_{i,l(p_i)} \rangle$ per agent $i \in \mathcal{D}$, where $\{p_{i,t}, p_{i,t+1}\} \in \mathcal{E}$, $p_{i,0} = v_{\text{start},i}$, $p_{i,l(p_i)} = v_{\text{goal},i}$, and $l(p_i)$ is the *length* or *travel distance* of path p_i. We consider *vertex conflicts* $\langle i, j, v, t \rangle$ that occur when two agents $i, j \in \mathcal{D}$ occupy the same location $v \in \mathcal{V}$ at time step t and *edge conflicts* $\langle i, j, u, v, t \rangle$ that occur when two agents $i, j \in \mathcal{D}$ traverse the same edge $\{u, v\} \in \mathcal{E}$ in opposite directions at time step t [9]. A plan P is a *solution*, i.e., *feasible* or *collision-free*, when it does not have any vertex or edge conflict. Our goal is to find a solution P^* that minimizes the *flowtime* $\sum_{p \in P} l(p)$ or alternatively the *makespan* $\max_{p \in P} l(p)$. The *completion rate* $\rho_t = \frac{|\{i \in \mathcal{D} | v_{t,i} = v_{\text{goal},i}\}|}{N} \in [0, 1]$ is the fraction of agents which have reached their goals at time step t. An instance I is completely solved when $\rho_T = 1$ at some time step T.

Despite MAPF being an NP-hard problem, there exist a variety of MAPF solvers that find optimal [11, 12], bounded suboptimal [13, 14], or quick but unbounded suboptimal solutions [15, 16]. Most MAPF solvers are centralized and require global information which limits scalability and flexibility regarding changes or new maps that would require expensive replanning and redistribution of plans [20–22]. This also prohibits applicability to partially observable and open domains, such as urban transportation or supply chains [18, 24–26].

2.2 Multi-Agent Reinforcement Learning (MARL)

MARL problems can be formulated as a partially observable *stochastic game* $\mathcal{M} = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \Omega \rangle$, where $\mathcal{D} = \{1, ..., N\}$ is a set of agents, $\mathcal{S}$ is a set of states s_t, $\mathcal{A} = \mathcal{A}_1 \times ... \times \mathcal{A}_N$ is the set of joint actions $a_t = \langle a_{t,1}, ..., a_{t,N} \rangle$, $\mathcal{P}(s_{t+1} | s_t, a_t)$ is the transition probability, $\mathcal{R}(s_t, a_t) = \langle r_{t,1}, ..., r_{t,N} \rangle \in \mathbb{R}^N$ is the joint reward with $r_{t,i}$ being the reward of agent $i \in \mathcal{D}$, $\mathcal{Z}$ is a set of local observations $z_{t,i}$ for each agent i, and $\Omega(s_{t+1}) = z_{t+1} = \langle z_{t+1,1}, ..., z_{t+1,N} \rangle \in \mathcal{Z}^N$ is the subsequent joint observation. Each agent i maintains an action-observation *history* $\tau_{t,i} \in (\mathcal{Z} \times \mathcal{A}_i)^t$. $\pi = \langle \pi_1, ..., \pi_N \rangle$ is the *joint policy* with *local policies* π_i, where $\pi_i(a_{t,i} | \tau_{t,i})$ is the action selection probability of agent i. Local policy π_i can be evaluated with a *value function* $Q_i^\pi(s_t, a_t) = \mathbb{E}_\pi[R_{t,i} | s_t, a_t]$ for all states $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$, where $R_{t,i} = \sum_{k=0}^{T-1} \gamma^k r_{t+k,i}$ is the *return* of agent i, $T > 0$ is the *horizon*, and $\gamma \in [0, 1]$ is the *discount factor*. In cooperative MARL, the goal is to find an *optimal joint policy* $\pi^* = \langle \pi_1^*, ..., \pi_N^* \rangle$ that maximizes the *utilitarian metric* for all states $s_t \in \mathcal{S}$ and joint actions $a_t \in \mathcal{A}$:

$$Q^{\pi}_{\text{tot}}(s_t, a_t) = \sum_{i \in \mathcal{D}} Q^{\pi}_i(s_t, a_t) \tag{1}$$

An optimal joint policy π^* corresponds to the *optimal value function* $Q^* = Q^{\pi^*}_{\text{tot}}$ defined by $Q^* = \max_\pi Q^\pi_{\text{tot}}$ for all states $s_t \in \mathcal{S}$ and joint actions $a_t \in \mathcal{A}$.

2.2.1 Policy gradient MARL

To learn optimal policies π^*_i in large state spaces, function approximators $\hat{\pi}_{i,\theta}$, such as deep neural networks with parameters or weights θ, are trained with gradient ascent on an estimate of $J = \mathbb{E}_\pi[R_{0,i}]$. *Policy gradient methods* use gradients g of the following form [52]:

$$g = A^{\hat{\pi}}_i(s_t, a_t) \nabla_\theta \log \hat{\pi}_{i,\theta}(a_{t,i}|\tau_{t,i}) \tag{2}$$

where $A^{\hat{\pi}}_i(s_t, a_t) = Q^{\hat{\pi}}_i(s_t, a_t) - V^{\hat{\pi}}_i(s_t)$ is the *advantage* of agent i and $V^{\hat{\pi}}_i(s_t) = \mathbb{E}_\pi[R_{t,i}|s_t]$ is its state value function. *Actor-critic* approaches often approximate $\hat{A}_i(\tau_t, a_t) \approx A^{\hat{\pi}_i}_i(s_t, a_t)$ by replacing $Q^{\hat{\pi}}_i(s_t, a_t)$ with $R_{t,i}$ and $V^{\hat{\pi}}_i(s_t)$ with $\sum_{a'_i \in \mathcal{A}_i} \hat{\pi}_{i,\theta}(a'_i|\tau_{t,i}) Q^{\hat{\pi}}_i(s_t, \langle a_{t,1}, ..., a'_i, ..., a_{t,N} \rangle)$ [26, 37, 53]. $Q^{\hat{\pi}}_i$ can be approximated with a *critic* $\hat{Q}_{i,\omega}$ and parameters ω using value-based RL [54, 55].

Alternatively, $\hat{\pi}_{i,\theta}$ can be trained via *proximal policy optimization (PPO)* by iteratively minimizing the following surrogate loss [56]:

$$\mathcal{L}^{\text{PPO}}_i(\theta) = \mathbb{E}[\min\{\hat{A}_i(\tau_t, a_t)\phi_{t,i}(\theta), \hat{A}_i(\tau_t, a_t)\text{clip}(\phi_{t,i}(\theta), 1-\epsilon, 1+\epsilon)\}] \tag{3}$$

where $\phi_{t,i}(\theta) = \frac{\hat{\pi}_{i,\theta}(a_{t,i}|\tau_{t,i})}{\hat{\pi}^{\text{old}}_{i,\theta}(a_{t,i}|\tau_{t,i})}$ is the policy probability ratio and $\epsilon \in [0, 1)$ is a clipping parameter. For simplicity, we omit the parameters θ, ω and write $\hat{\pi}_i$, $\hat{Q}_i$ for the rest of the paper.

2.2.2 Centralized Training Decentralized Execution (CTDE)

For many problems, training takes place in a laboratory or in a simulated environment, where global information is available [37, 39]. Therefore, state-of-the-art MARL algorithms approximate value functions $\hat{Q}_i$, which condition on global states s_t and joint actions a_t, and use them as a critic in (2) or (3) [38, 43]. While the centralized value functions $\hat{Q}_i$ are only required during RL training, the learned policies $\hat{\pi}_i$ fully condition on local histories $\tau_{t,i}$ thus enable decentralized execution. Unlike MAPF, these policies can *generalize* over a variety of scenarios and, thus, ideally, do not need any centralized retraining or replanning for changes or new maps [19].

$\hat{Q}_i$ can be approximated separately for each agent i while integrating global information, which is done in actor-critic MARL algorithms like MAPPO or MADDPG [38, 43]. However, this approach lacks a *multi-agent credit assignment mechanism* for agent teams, where all agents jointly optimize the same objective $Q^{\hat{\pi}}_{\text{tot}}$ (1).

Alternatively, a common value function $\hat{Q}(\tau_t, a_t) \approx Q^{\hat{\pi}}_{\text{tot}}(s_t, a_t)$ can be learned, which is factorized into *local utility functions* $\langle \hat{Q}_1, ..., \hat{Q}_N \rangle$ by using a *factorization operator* Ψ [18, 57]:

$$\hat{Q}(\tau_t, a_t) = \Psi(\hat{Q}_1(\tau_{t,1}, a_{t,1}), ..., \hat{Q}_N(\tau_{t,N}, a_{t,N})) \tag{4}$$

In practice, Ψ is realized with deep neural networks, such that $\langle \hat{Q}_1, ..., \hat{Q}_N \rangle$ can be learned end-to-end via backpropagation by minimizing the mean squared *temporal difference (TD)* error [39, 58]. A factorization operator Ψ is *decentralizable* when being *IGM (Individual-Global-Max) consistent* such that [59]:

$$\text{argmax}_{a_t} \hat{Q}(\tau_t, a_t) = \begin{pmatrix} \text{argmax}_{a_{t,1}} \hat{Q}_1(\tau_{t,1}, a_{t,1}) \\ \vdots \\ \text{argmax}_{a_{t,N}} \hat{Q}_N(\tau_{t,N}, a_{t,N}) \end{pmatrix} \tag{5}$$

There exists a variety of factorization operators Ψ, which are IGM consistent, according to (5), such as:

Value Decomposition Networks (VDN) VDN uses a linear sum to approximate $\hat{Q}(\tau_t, a_t) = \Psi_{\text{VDN}}(\cdot) = \sum_{i \in \mathcal{D}} \hat{Q}_i(\tau_{t,i}, a_{t,i})$ [58].

Monotonic Value Factorization (QMIX) QMIX formulates Ψ_{QMIX} as a nonlinear monotonic combination of $\langle \hat{Q}_i \rangle_{i \in \mathcal{D}}$ with a mixing network, generated by hypernetworks [60]. The mixing network has nonnegative weights to satisfy the constraint $\frac{\delta \hat{Q}}{\delta \hat{Q}_i} \geq 0$ for each agent $i \in \mathcal{D}$.

While VDN and QMIX are restricted to objective functions that are linear or monotonic, respectively, there exist more general and advanced techniques like QPLEX that use nonlinear transformations, e.g., via multi-head attention networks [61].

2.3 Curriculum learning

Curriculum learning is a machine learning paradigm inspired by human learning to master complex tasks through stepwise solving of easier (sub-)tasks, which are sorted by difficulty [62, 63]. The difficulty can depend on various aspects like the complexity of data samples, the objective function, or the learned model [42, 64]. Curriculum learning has been applied to *reinforcement learning (RL)* to solve hard exploration problems with sparse rewards or dynamic constraints [65]. The methods are typically based on self-play [36, 66], task graphs with traversal mechanisms [67], or automatic generation of tasks [68, 69].

A key challenge of curriculum learning is to find or generate a suitable sequence of tasks that are neither too easy nor too difficult for the learner to ensure steady and robust progress [42, 67, 68].

We focus on *reverse curriculum* learning, where we assume explicit goal states as in the MAPF problem (Section 2.1). The curriculum consists of a sequence of tasks, where the (expected) distance between agent and goal gradually increases [42, 70].

3 Related work

3.1 Reverse curriculum generation

Many works on RL-based motion control assume a single goal state, which is easy to specify [71, 72]. A *reverse curriculum* is defined, where the start state is initialized within a short distance to the goal state. The distance gradually increases with the convergence or performance improvement of the agent [42, 70, 73]. Our work is based on reverse curriculum generation, focusing on *multi-agent path finding (MAPF)*. In MAPF, there are *several goal states* that are unique per instance I (which can vary for the same map G, though). We propose a simple *confidence-based approach* to adapt the curriculum using *statistical significance testing* of performance estimates. In contrast to most prior works, which focus on explicitly resettable environments for the start locations, e.g., simulators, we focus on *goal allocation*, which can be realized as an online task (re)assignment without requiring the environment to be resettable.

3.2 Curriculum learning in MARL

Curriculum learning has been widely used in single-agent or two-player zero-sum games to improve convergence speed or performance. While many of these approaches are based on the foundations of multi-agent learning [69, 74, 75], there exist methods particularly designed for MARL based on self-play, agent skills, and population-based training [40, 41, 76]. These methods are typically very complex due to heavily engineered architectures and mechanisms thus requiring a significant amount of compute. As our work focuses on *simple and efficient* MARL approaches to MAPF, we do not consider such resource and tuning-intensive training regimes.

3.3 Machine learning-guided MAPF

Previously, machine learning has been integrated into (centralized) search-based MAPF solvers to guide the heuristic search algorithm, e.g., via node selection or agent prioritization [30, 77–82], to replace common handcrafted heuristics [11, 13, 15, 17]. In this paper, we focus on MARL-based policy learning for *decentralized MAPF* without using any search-based MAPF solver or handcrafted heuristic.

3.4 MARL for decentralized MAPF

MAPF and MARL are very active research areas with remarkable progress in recent years [15, 40, 41]. Both fields have been mainly studied independently of each other, with only a few works attempting to connect them. The first work in this direction is *Pathfinding via Reinforcement and Imitation Multi-Agent Learning (PRIMAL)* [19]. PRIMAL and its successor approaches, PRIMAL2, SCRIMP, etc., are heavily engineered and very complex, using extremely large neural networks to represent policies, extensively shaped rewards, and centralized MAPF solvers for imitation learning to address the challenging aspects of MAPF, i.e., sparse rewards, dynamic constraints, and coordination [19, 29, 44, 45, 83]. Despite their effectiveness in specific scenarios, these approaches are very expensive to use

due to the significant effort required for fine-tuning and the enormous computational and data requirements. Some recent works proposed manually designed curricula to enhance PRIMAL but still rely on very complex architectures and reward functions [84, 85]. Inspired by *The Bitter Lesson* of [46], we aim for a simpler and more general foundation to enable MARL-based decentralized MAPF in a sustainable manner, with significantly lower costs and without distracting ourselves with engineering details. Therefore, we propose a *simple reverse curriculum scheme* to ease applicability and enable faster progress in this direction [20, 21].

4 MAPF as a stochastic game

To apply MARL techniques to MAPF in a general way, we first need to formulate the MAPF problem defined in Section 2.1 as a stochastic game $\mathcal{M}$, according to Section 2.2. Similar to prior works [9, 19, 44, 45], we focus on discrete gridworlds but keep our formulation general, e.g., for our analysis in Section 5.4. An adaptation of our methods to arbitrary graphs, e.g., using graph neural networks, is left for future work.

4.1 Problem formulation

In both settings, the set of agents $\mathcal{D}$ is equivalent. Given a map $G = \langle \mathcal{V}, \mathcal{E} \rangle$, the state space $\mathcal{S}$ is defined by the joint locations of all agents $s_t = \langle v_{t,i}, ..., v_{t,N} \rangle \in \mathcal{S} \subset \mathcal{V}^N$, where each location in s_t is unique such that $v_{t,i} \neq v_{t,j}$ for each agent pair $i, j \in \mathcal{D}$ with $i \neq j$. The individual action space $\mathcal{A}_i$ of each agent i is defined by the maximum degree of map G plus a wait action, such that $|\mathcal{A}_i| = 1 + \max_{v \in \mathcal{V}} \{\deg(\mathrm{v})\}$. In 4-neighborhood gridworlds, as displayed in Fig. 2, each agent would be able to wait or move in all cardinal directions. The state transitions are deterministic, where a valid move action will change the location of the corresponding agent. Attempts to move over non-existent edges or cause collisions, i.e., vertex or edge conflicts, are automatically treated as wait action. The individual reward $r_{t,i}$ is defined by +1 if agent i reaches its goal $v_{\mathrm{goal},i}$, 0 when agent i is staying at its goal

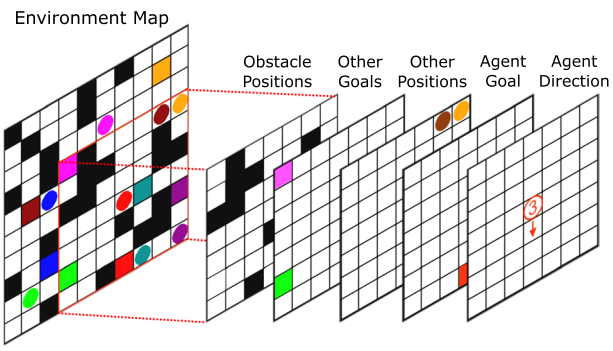

Fig. 2 Example of an individual observation of the red agent in a gridworld. Agents are represented as colored circles, their goals as similarly-colored squares, and obstacles as black squares. Each agent i has a limited *field of view (FOV)* of the environment map, which is centered around its location encoded by five channels: The locations of obstacles, other agents' goals, nearby agents, and the location of the goal $v_{\mathrm{goal},i}$ if within the FOV, and the Manhattan distance and direction of agent i to its goal

location $v_{\text{goal},i}$, and -1 otherwise. Each agent i can partially observe the state s_t through a local neighborhood around its location $v_{t,i}$. For gridworlds, we assume a 7×7 *field of view (FOV)* similar to PRIMAL, which is illustrated in Fig. 2 [19]. The features of an observation $z_{t,i}$, i.e., obstacles, other agents and their goals, and the direction and goal location of agent i, are encoded as a multi-channel image. The direction channel encodes the Manhattan distance to the goal $v_{\text{goal},i}$, and indicates the direction to it.

When the discount factor is $\gamma = 1$, then the negated return $-R_{t,i}$ of each agent i is equivalent to its travel time $l(p_i)$ plus a constant reward of +1 from the beginning to time step t, if $v_{\text{goal},i}$ was reached, and horizon T otherwise. Thus, maximizing $Q^{\pi}_{\text{tot}} = \sum_{i \in \mathcal{D}} Q^{\pi}_i$ in MARL (Section 2.2 and (1)) is equivalent to minimizing the expected flowtime $\mathbb{E}_I[\sum_{p \in P} l(p)]$ in MAPF w.r.t. any instance I (Section 2.1) [86].

4.2 Conceptual discussion

Our problem formulation is simple and general, which allows us to solve it in a black-box manner that is more intuitive for standard MARL methods without extensive engineering [39, 43, 61]. However, the simplicity of our formulation induces a *hard exploration problem* since the reward is sparse, in contrast to PRIMAL and related approaches [19, 44, 45]. In the following, we will discuss some particular formulation decisions and potential alternatives that we plan to investigate further in the future.

4.2.1 Automatic collision prevention

In MAPF, all agents must coordinate to **(1)** navigate to their goals and **(2)** avoid collisions, according to Section 2.1. Learning policies for both requirements directly corresponds to a multi-objective optimization problem, which is known to be challenging for large-scale scenarios, especially in the context of safe reinforcement learning [35, 87]. Addressing collision prevention explicitly in the training regime has led to manual engineering and very complicated machinery [35], such as extensive reward shaping [19], explicit communication [45], overfitting on specific scenarios [44], and the ad hoc invocation of search algorithms [35, 83].

To keep a simple focus and methodology, we concentrate only on the primary MAPF objective of optimizing the flowtime and delegate collision prevention to the environment, e.g., through some low-level control mechanism of the actual robots. The automatic conversion to wait actions also eases our exploration efficiency analysis in Section 5.4.3. Note that collision attempts are still discouraged through time penalties.

The explicit and adequate use of collision penalties depends on the domain (w.r.t. map structure and agent density) and could fundamentally change the primary MAPF objective such that agents deviate substantially from their original task [37, 88]. For example, agents could simply wait indefinitely to avoid collision penalties without reaching their actual goals [23, 58]. Our experimental results in Section 6.2.3 indicate that defining such explicit penalties is non-trivial and can yield poor global behavior.

Alternative automatic measures to prevent collisions include the termination of episodes after a collision [87, 89], shielding of unsafe actions via rules [35], or conformal prediction of other agents' movements [90]. Considering all alternative measures is out of scope for this paper and thus will be considered for future work.

4.2.2 Reward function trade-offs

Since any time step is penalized with -1 anyway (unless an agent reaches or occupies its goal), all agents are discouraged from unnecessary delays, which include collision attempts. Necessary waits are penalized as well, which is in line with the primary MAPF objective, i.e., the flowtime [9, 12]. Unlike prior works [19, 44, 45], we do not need additional handcrafted penalties for particular situations like collisions, blocking, or waiting, as listed in Table 3, which could fundamentally change the primary MAPF objective and lead to unintended side effects [88]. However, the partial observability along with further theoretical assumptions, discussed in Section 5.4, could lead to deadlocks and motivates the distinction between necessary and unnecessary waits in future work.

Furthermore, when optimizing alternative objectives that do not account for wait actions, such as the sum of loss [16], the time penalty must be adjusted to distinguish between necessary and unnecessary waits. However, in practice, such a distinction is not trivial without a suitable predictive measure, such as a MAPF solver or a value function, since the necessity of a wait action is typically determined in hindsight [19]. In such cases, the reward function could be learned itself via meta-learning [91, 92]. Peer incentivization is a decentralized alternative in which agents reward or penalize each other for moving or waiting [23]. Considering all alternative reward approaches is out of scope for this paper and thus will be considered for future work.

5 Confidence-Based curricula for MAPF

5.1 Training scheme

We assume a separate training phase to learn coordinated local policies $\hat{\pi}_i$ for decentralized execution. We train $\hat{\pi}_i$ via policy gradient methods, according to (2) or (3). The critics $\hat{Q}_i$ are trained via CTDE methods to exploit global information during training using either independent learning like MAPPO or value factorization like VDN, QMIX, or QPLEX, as illustrated in Fig. 3 [39, 43, 61]. Since the value factorization-based actor-critic scheme has been used in a variety of prior works [26, 53, 93, 94], we do not claim novelty here, but propose it as a basic approach to train cooperative policies via credit assignment mechanisms [39, 59, 61].

To address the coordination problem, as mentioned in the introduction, we suggest optimizing individual utilities $\hat{Q}_i \approx Q_i^{\hat{\pi}}$ under consideration of the utilitarian metric $\hat{Q} \approx Q_{\text{tot}}^{\hat{\pi}}$ in (1). For that, the individual utilities $\hat{Q}_i$ can be learned end-to-end through a factorization operator Ψ like VDN, QMIX, or QPLEX to consider multi-agent credit assignment from a cooperative perspective [18, 39, 57, 61]. The factorization operator Ψ approximates the expected sum of individual returns by minimizing the *factorization loss* $\mathcal{L}^{\Psi}$ defined by:

$$\mathcal{L}^{\Psi} = \mathbb{E}\Big[\Big(\Psi(\hat{Q}_1(\tau_{t,1}, a_{t,1}), ..., \hat{Q}_N(\tau_{t,N}, a_{t,N})) - \sum_{i \in \mathcal{D}} R_{t,i}\Big)^2\Big] \tag{6}$$

The local policies $\hat{\pi}_i$ are then trained according to (2) or (3) using *counterfactual advantages* $\hat{A}_i$ defined by [53]:

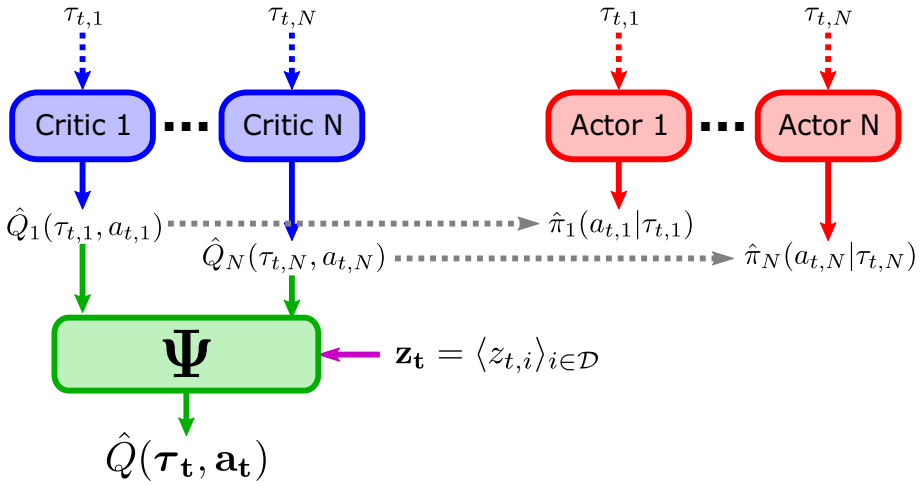

Fig. 3 Common actor-critic scheme as used in various prior works on cooperative MARL [26, 53, 93, 94]. A separate critic is trained for each actor using some centralized factorization operator Ψ like VDN, QMIX, or QPLEX [18, 57]

$$\hat{A}_i(\tau_t, a_t) = R_{t,i} - \sum_{a' \in \mathcal{A}_i} \hat{\pi}_i(a'|\tau_{t,i}) \hat{Q}_i(\tau_{t,i}, a') \tag{7}$$

The advantage $\hat{A}_i$ incentivizes the optimization of travel distances under implicit consideration of other agents through $\hat{Q}_i$ and Ψ w.r.t. (1) and (6).

5.2 Reverse curriculum scheme

The training scheme described above represents a general approach to learning coordinated policies $\hat{\pi}_i$ [26, 53]. However, sparse rewards and dynamic constraints in our MAPF setting (Section 4) pose particular challenges that require a suitable curriculum to learn meaningful policies [42, 70]. Unlike prior works that rely on shaped reward functions with various penalties and expensive expert data for imitation learning, we propose *Confidence-based Auto-Curriculum for Team Update Stability (CACTUS)* to enhance the training scheme of Section 5.1 without significant effort and costs.

Definition 5.1 (CACTUS region and allocation radius) A CACTUS region is defined by $\mathcal{V}^{\text{CACTUS}}(v) = \{u \in \mathcal{V} | \text{DIST}(u, v) \leq R_{\text{alloc}}\}$ for each $v \in \mathcal{V}$, assuming that v is reachable from any surrounding location $u \in \mathcal{V}^{\text{CACTUS}}(v)$ within an allocation radius of $R_{\text{alloc}} \geq 1$ steps, and vice versa. The distance function $\text{DIST} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ denotes a reachability measure, e.g., the geodesic or shortest path distance.

At the beginning of every episode m, each agent i starts at a random location $v_{\text{start},i} \in \mathcal{V}$ and needs to navigate to a randomly assigned goal location $v_{\text{goal},i} \in \mathcal{V}^{\text{CACTUS}}(v_{\text{start},i}) \setminus \{v_{\text{goal},1}, ..., v_{\text{goal},i-1}, v_{\text{start},i}\}$. R_{alloc} characterizes the *potential difficulty* of the generated instances I as larger allocation radii may require the agents to move and explore over longer distances to locate their goals. Thus, our reverse

curriculum scheme starts with a small radius of $R_{\text{alloc}} = 1$, which gradually increases with improving performance measured by the *completion rate* ρ_T^m for episode m.

CACTUS uses a statistical approach to decide whether to increment R_{alloc} or not. After each epoch of E episodes m, we measure the *average completion rate* $\mu = \frac{1}{E}\sum_{m=1}^{E} \rho_T^m$ and its *standard deviation* $\sigma = \sqrt{\frac{1}{E-1}\sum_{m=1}^{E}(\rho_T^m - \mu)^2}$.

Assuming that the completion rates ρ_T^m follow a normal distribution, CACTUS increments R_{alloc} by 1, if $\mu - \eta\sigma \geq U$, where $U \in (0, 1)$ is the *curriculum decision threshold* and $\eta > 0$ is a *deviation factor* to specify the confidence level. For example, if $U = 75\%$ and $\eta = 2$ then R_{alloc} would be incremented only if all agents achieve an average completion rate over 75% with a confidence level of around 97%. This corresponds to a statistical significance test of the currently achievable performance. Note that we only regard *one-tailed tests* here, where we assume no upper limit to the average completion rate of agents (except for $\mu = 100\%$, where σ would be zero). The curriculum update scheme is illustrated in Fig. 1.

The complete formulation of CACTUS is given in Algorithm 1. $\mathcal{G}$ is a set of training maps or a map generator, $\text{DIST} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a reachability distance function, U is the curriculum decision threshold, and η is the deviation factor.

procedure *CACTUS*($\mathcal{G}$, *DIST*, U, η)
  Initialize parameters of $\hat{\pi}_i$, $\hat{Q}_i$ for each agent $i \in \mathcal{D}$ and Ψ
  Set $R_{alloc} \leftarrow 1$
  for epoch $x \leftarrow 1, X$ **do**
    for episode $m \leftarrow 1, E$ **do**
      Randomly select or generate map G from $\mathcal{G}$
      Sample s_0 ▷ Set start locations $v_{start,i}$
      for agent $i \in \mathcal{D}$ **do**
        Set $\tau_{0,i}$ based on $\Omega(s_0)$
        Set $\mathcal{V}_{goal,i} \leftarrow \mathcal{V}^{CACTUS}(v_{start,i}) \setminus \{v_{goal,1}, ..., v_{goal,i-1}, v_{start,i}\}$
        Randomly select goal location $v_{goal,i} \in \mathcal{V}_{goal,i}$
      end for
      for time step $t \leftarrow 0, T-1$ **do**
        for agent $i \in \mathcal{D}$ **do**
          $a_{t,i} \sim \pi_i(\cdot|\tau_{t,i})$
        end for
        $a_t \leftarrow \langle a_{t,1}, ..., a_{t,N} \rangle$
        Execute joint action a_t
        $s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)$
        $z_{t+1} \leftarrow \Omega(s_{t+1})$
        $e_t \leftarrow \langle \tau_t, a_t, r_t, z_{t+1}, \rangle$
        Store experience sample e_t
        $\tau_{t+1} \leftarrow \langle \tau_t, a_t, z_{t+1} \rangle$
      end for
      $\rho_T^m \leftarrow \frac{|\{i \in \mathcal{D} | v_{T,i} = v_{goal,i}\}|}{N}$
    end for
    Train Ψ and $\hat{\pi}_i$, $\hat{Q}_i$ for each agent $i \in \mathcal{D}$ with all e_t
    Calculate average μ and standard deviation σ of all ρ_T^m
    if $\mu - \eta\sigma \geq U$ **then** ▷ Confidence-based performance check
      $R_{alloc} \leftarrow R_{alloc} + 1$
    end if
  end for
  return $\langle \hat{\pi}_1, ..., \hat{\pi}_N \rangle$
end procedure

Algorithm 1 Confidence-Based curriculum learning for MAPF.

5.3 Conflict-Based curriculum scheme

CACTUS enables efficient exploration to learn navigation skills over long distances. Collision avoidance or conflict resolution is addressed indirectly through overlapping CACTUS regions $\mathcal{V}^{\text{CACTUS}}(v_{\text{start},i})$, which grow over time as the allocation radius R_{alloc} increases, as shown in Figs. 1 and 4.

We now propose a variant of CACTUS, called *Confidence- and Conflict-Based Curriculum Learning with Allocation Radius Adaptation* (C^{3}LARA), which uses weighted sampling of goal locations based on overlapping CACTUS regions $\mathcal{V}^{\text{CACTUS}}(v_{\text{start},i})$ to improve conflict resolution in instances of high agent density.

Instead of sampling goal locations uniformly in Line 11 of Algorithm 1, we now assign a probability to each potential location $v \in \mathcal{V}$ proportional to the number of overlapping CACTUS regions that v belongs to. Thus, for each agent $i \in \mathcal{D}$ with a start location $v_{\text{start},i} \in \mathcal{V}$, a CACTUS region $\mathcal{V}^{\text{CACTUS}}(v_{\text{start},i}) \subseteq \mathcal{V}$, and a set of goal locations already assigned to other agents $\{v_{\text{goal},1}, ..., v_{\text{goal},i-1}\}$, we define:

$$\mathbb{P}(v_{\text{goal},i} = v) = \frac{C(v)}{\sum_{u \in \mathcal{V}^{\text{CACTUS}}(v_{\text{start},i}) \setminus \{v_{\text{goal},1}, \ldots, v_{\text{goal},i-1}, v_{\text{start},i}\}} C(u)} \tag{8}$$

where $C(v) = |\{j \in \mathcal{D} | v \in \mathcal{V}^{\text{CACTUS}}(v_{\text{start},j})\}|$ is the *goal candidate weight*, represented by the number of all CACTUS regions containing $v \in \mathcal{V}$. $C(v)$ characterizes the *conflict potential* of v due to an increased chance of encountering other agents [21].

An example is shown in Fig. 4. At the early stages of training, all agents are randomly scattered with small CACTUS regions that rarely overlap. Thus, the distribution of $\mathbb{P}(v_{\text{goal},i} = v)$ is uniform in most cases. With an increasing allocation radius R_{alloc}, according to Algorithm 1, the number of overlaps increases, such that C^{3}LARA biases the goal allocation toward intersection areas of different CACTUS regions. Note that C^{3}LARA does not introduce any additional hyperparameter and, therefore, can be straightforwardly applied to Line 11 in Algorithm 1.

With C^{3}LARA, we can incentivize agents to improve their conflict resolution which potentially improves the generalization to instances of high agent density. As the agents progress, the CACTUS regions will eventually span the whole map, such that the goal

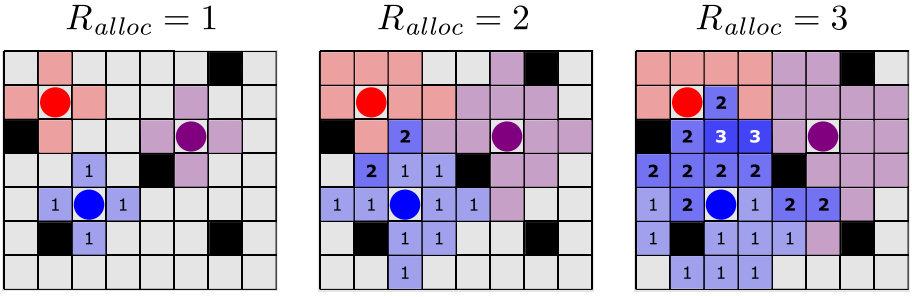

Fig. 4 Example of the goal candidate weighting scheme of C^{3}LARA for $N = 3$ agents (colored circles), focusing on the blue agent. In the beginning, the CACTUS regions around the agents do not overlap, thus the weights are $C(v) = 1$ for most locations $v \in \mathcal{V}^{\text{CACTUS}}(v_{\text{start},i})$. Over time the CACTUS regions will overlap with increasing allocation radius R_{alloc}. All locations v in intersection areas have weights $C(v) > 1$, depending on the number of overlapping regions, according to (8)

location probabilities will revert back to a uniform distribution. To maintain a high conflict potential per location, we need to increase the agent density by increasing the number of agents or decreasing the map size, which we defer to future work.

5.4 Theoretical properties of our curriculum scheme

We now analyze the strengths and limitations of our curriculum scheme from a theoretical perspective, contrasting prior works that are heavily engineered and complex but lack a sound foundation [19, 29, 44, 45, 83]. Our analyses focus on CACTUS and can be extended to C^{3}LARA. For simplicity, we first analyze the optimality and exploration efficiency of CACTUS from a single-agent perspective. We then discuss the conditions to generalize our findings to the multi-agent setting, including open challenges that we defer to future work.

5.4.1 Preliminaries

In the following, we state the main assumptions and definitions (in addition to the CACTUS region of Definition 5.1) on which our analyses will be based.

Assumption 5.1 *We assume a connected and undirected unweighted graph G, and use the geodesic distance or shortest path distance as* $\mathrm{DIST} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ *for* R_{alloc}.

Assumption 5.1 aligns with state-of-the-art works in MAPF [15, 16]. The geodesic distance is required to provide bounds on the reachability of locations within the CACTUS regions.

Definition 5.2 (Random action probability) The probability of randomly selecting an individual action $a_{t,i} \in \mathcal{A}_i$ is $p = \frac{1}{|\mathcal{A}_i|} = \frac{1}{1+\max_{v \in \mathcal{V}}\{\deg(v)\}}$, according to Section 4.

Definition 5.3 (CACTUS exploration policy) An agent at location $v_{t,i}$ controlled by the CACTUS exploration policy π_i^{CACTUS} reaches the goal location $v_{\mathrm{goal},i} \in \mathcal{V}$ within $H \geq R_{\mathrm{alloc}}$ steps with a probability of at least $U \in (0, 1)$, if $v_{t,i} \in \mathcal{V}^{\mathrm{CACTUS}}(v_{\mathrm{goal},i})$. Otherwise, π_i^{CACTUS} behaves randomly by selecting each action $a_{t,i} \in \mathcal{A}_i$ with a probability of $p = \frac{1}{|\mathcal{A}_i|}$, according to Definition 5.2.

π_i^{CACTUS} represents a behavioral policy learned via CACTUS, according to Algorithm 1, where all agents learn to reach their goals within the corresponding CACTUS regions with sufficiently high confidence. The curriculum decision threshold U represents the target completion rate that is ensured through confidence-based or statistical significance testing of the current performance in Line 29 of Algorithm 1.

Assumption 5.2 $\mathcal{V}^{\mathrm{CACTUS}}(v_{\mathrm{goal},i})$ *represents an absorbing region for the policy* π_i^{CACTUS}. *When agent i controlled by* π_i^{CACTUS} *enters* $\mathcal{V}^{\mathrm{CACTUS}}(v_{\mathrm{goal},i})$ *at time step t, it only moves within that region, such that* $v_{t+k,i} \in \mathcal{V}^{\mathrm{CACTUS}}(v_{\mathrm{goal},i})$ *with* $k > 1$. *Any other agent* $j \neq i$ *does not get absorbed by* $\mathcal{V}^{\mathrm{CACTUS}}(v_{\mathrm{goal},i})$.

Due to our curriculum scheme and significance tests in Algorithm 1, we can assume that agents will move toward their goal locations with sufficiently high confidence once they are in a reachable range, i.e., R_{alloc} steps. Note that agents are not absorbed by the CACTUS regions of other agents' goals.

Definition 5.4 (Default exploration policy) The default or random exploration policy π_i^{Default} always selects each action $a_{t,i} \in \mathcal{A}_i$ with a probability of $p = \frac{1}{|\mathcal{A}_i|}$.

We use π_i^{Default} to model the exploration behavior of most standard RL algorithms (Section 2.2.1), as used in prior works on MARL for MAPF [19, 29, 44, 45, 83].

Note that π_i^{CACTUS} and π_i^{Default} represent *behavioral policies* used for exploration, i.e., data acquisition. The *target policies*, e.g., greedy policies w.r.t. $\hat{Q}_i$ can be different from the behavioral policy, depending on the concrete RL algorithm [54, 95].

5.4.2 Single-Agent optimality

For this preliminary analysis, we assume a single-agent setting, i.e., $N = 1$, and thus omit the agent index *i*. Under certain conditions, discussed in Section 5.4.4, this analysis can be generalized to multi-agent settings with $N > 1$ by considering the joint actions $a_t = \langle a_{t,1}, ..., a_{t,N} \rangle$, joint locations $s_t = \langle v_{t,i}, ..., v_{t,N} \rangle$, and joint CACTUS regions $\bigcup_{i \in \mathcal{D}} \mathcal{V}^{\text{CACTUS}}(v_{\text{goal},i})$ of all agents $i \in \mathcal{D}$.

If we can perfectly approximate the value function of the target policy π such that $\hat{Q} = Q^{\pi}$, we can show that our curriculum scheme does not compromise the ability of RL algorithms to converge to the optimal values $Q^*(v, \cdot)$ and policies π^* for all $v \in \mathcal{V}$.

Theorem 1 *Assuming that CACTUS, according to Algorithm 1, converges to the optimal values $Q^*(v, \cdot)$ for all $v_{\text{goal}} \in \mathcal{V}$ and $v \in \mathcal{V}^{\text{CACTUS}}(v_{\text{goal}})$ for an allocation radius of $R_{\text{alloc}} \geq 1$. Then, optimal RL algorithms using CACTUS can converge to the optimal value function Q^* and optimal policies π^* with $R_{\text{alloc}} \to \infty$.*

Proof According to the Bellman principle of optimality [95, 96], all optimal values $Q^*(v, \cdot)$ learned so far are not affected by an expansion of $\mathcal{V}^{\text{CACTUS}}(v_{\text{goal}})$ when incrementing R_{alloc}, since all new locations *u* have a strictly greater distance to v_{goal} and thus a smaller expected return [86]. An example is provided in Fig. 5. As training progresses with $R_{\text{alloc}} \to \infty$ and $\mathcal{V}^{\text{CACTUS}}(v_{\text{goal}}) \to \mathcal{V}$, an optimal RL algorithm will eventually learn the optimal values for the whole map *G* (Assumption 5.1). □

Theorem 1 confirms that any optimal RL algorithm exploring via our CACTUS scheme, according to Algorithm 1, can also converge to the optimal value function Q^* and thus to an optimal policy π^* when gradually incrementing R_{alloc} from 1 to ∞. According to [86] and Section 4.2, our rewards defined in Section 4 are aligned with the original path finding objective thus enabling optimality, in contrast to manual reward shaping [19, 44, 45], which can lead to unintended side effects [88].

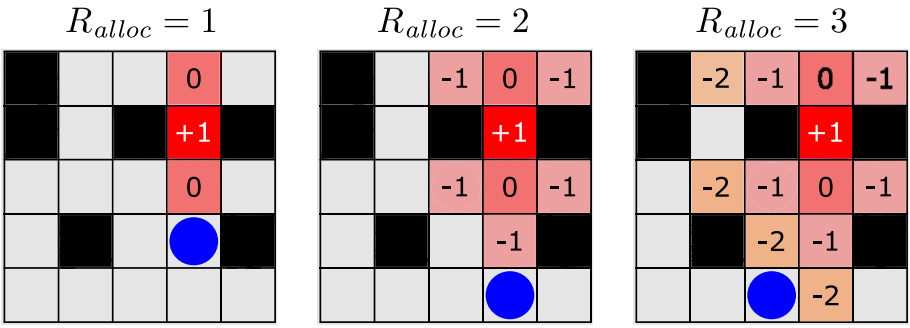

Fig. 5 Value approximation via CACTUS for a single agent (blue circle) and its goal location v_{goal} (red square with +1). Note that the optimal values learned in previous epochs are not affected when incrementing R_{alloc}. Empty cells have a value of zero

5.4.3 Exploration efficiency

For this preliminary analysis, we assume a single-agent setting, i.e., $N = 1$, and thus omit the agent index *i*. Under certain conditions, discussed in Section 5.4.4, this analysis can be generalized to multi-agent settings with $N > 1$ by considering the joint actions $a_t = \langle a_{t,1}, ..., a_{t,N} \rangle$, joint locations $s_t = \langle v_{t,i}, ..., v_{t,N} \rangle$, and joint CACTUS regions $\bigcup_{i \in \mathcal{D}} \mathcal{V}^{\text{CACTUS}}(v_{\text{goal},i})$ of all agents $i \in \mathcal{D}$.

The empirical progress of an RL algorithm depends on its chance of reaching the goal location v_{goal} within T time steps. Failed attempts or episodes that do not lead to v_{goal}, always yield an uninformative return of $-T$, which can lead to poor policies in practice [56, 97–101]. Thus, we will now analyze the exploration efficiency of the behavioral policies π^{CACTUS} and π^{Default}.

Theorem 2 *Given a start location $v_{\text{start}} \in \mathcal{V}$ with $\text{DIST}(v_{\text{start}}, v_{\text{goal}}) = R_{\text{alloc}} + 1$ and a horizon of $T > H$ steps. Then, the CACTUS exploration policy π^{CACTUS} has a higher chance of reaching v_{goal} than the default exploration policy π^{Default} if $U > Cp^T$, where $C \geq 1$ is the number of possible paths from v_{start} to v_{goal} within T steps.*

Proof For π^{CACTUS}, we model our instances *I* as an absorbing Markov chain [102, 103] with three nodes, where v_{start} represents the sole transient node $\mathcal{V}^{\text{Start}}$. All locations $v \in \mathcal{V}^{\text{CACTUS}}(v_{\text{goal}})$ represent an absorbing node $\mathcal{V}^{\text{CACTUS}}$ and all remaining locations $u \notin \mathcal{V}^{\text{CACTUS}}(v_{\text{goal}}) \cup \{v_{\text{start}}\}$ represent the other absorbing node $\mathcal{V}^{X}$. Since *G* is connected, the agent can reach $\mathcal{V}^{\text{CACTUS}}$ and $\mathcal{V}^{X}$ from $\mathcal{V}^{\text{Start}}$ or remain at $\mathcal{V}^{\text{Start}}$ due to waiting or invalid move attempts, as illustrated in Fig. 6. Since each transition has a probability of at least $p = \frac{1}{|\mathcal{A}|}$, the chance P_{CACTUS} of reaching the goal satisfies:

$$P_{\text{CACTUS}} \geq U \sum_{k=1}^{T-H} p^k$$

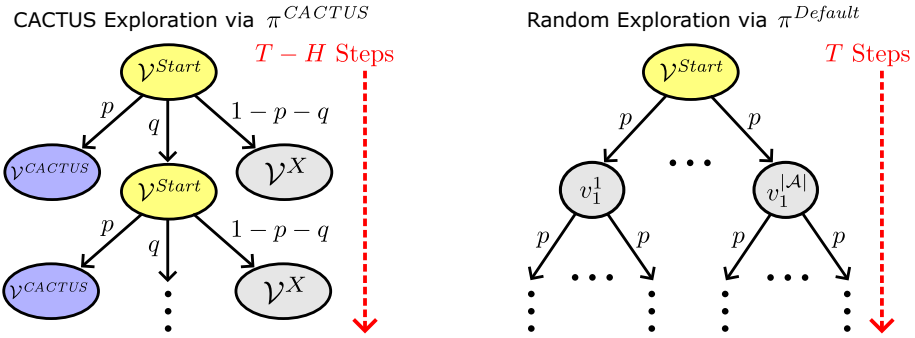

Fig. 6 Illustration of the exploration spaces of π^{CACTUS} and π^{Default}. Left: The exploration behavior of π^{CACTUS} is modeled as an absorbing Markov chain with two absorbing nodes $\mathcal{V}^{\text{CACTUS}}$ (blue) and $\mathcal{V}^{X}$ (gray), respectively. The transition probabilities $p = \frac{1}{|\mathcal{A}|}$ and $q \geq p$ (waiting or invalid move attempts) are used to estimate a lower bound probability of reaching the goal. Right: The exploration space of π^{Default} is exponential w.r.t. the horizon T, where each action is selected with a chance of p

The right-hand side is a lower bound of the true probability P_{CACTUS}. In principle, the agent can still reach v_{goal} from any location in $\mathcal{V}^X$ due to G being connected and undirected, according to Assumption 5.1, which we do not consider for simplicity.

π^{Default} explores completely randomly without any absorbing region. Therefore, its goal-reaching probability is $P_{\text{Default}} = Cp^T = \frac{C}{|\mathcal{A}|^T}$ because the exploration space $|\mathcal{A}|^T$ is exponential w.r.t. the horizon T, as illustrated in Fig. 6 [73, 86].

If $U > Cp^T$, we can now show that $P_{\text{CACTUS}} > P_{\text{Default}}$:

$$P_{\text{CACTUS}} \geq U \sum_{k=1}^{T-H} p^k > Cp^T = P_{\text{Default}} \iff U > \frac{Cp^T}{\sum_{k=1}^{T-H} p^k}$$
$$\iff U > Cp^T$$

Given a finite horizon of $T > H$ steps, $\sum_{k=1}^{T-H} p^k < \sum_{k=1}^{\infty} p^k = \frac{1}{1-p} - 1 = \frac{1}{|\mathcal{A}|-1} \leq 1$, where $|\mathcal{A}| \geq 2$ due to the connectedness of the map G and the wait action, according to Section 4. Thus, we can infer that $\frac{Cp^T}{\sum_{k=1}^{T-H} p^k} > Cp^T$, and confirm $U > Cp^T$. □

In theory, we can estimate C via powers of the adjacency matrix of G to determine an adequate curriculum decision threshold U in Algorithm 1 [104]. In practice, we can assume that C is very small compared to the exploration space $\frac{1}{p^T} = |\mathcal{A}|^T$. For example if $Cp^T \geq p$, then random exploration would find the goal location with a chance of at least 20% in any 4-neighborhood gridworld, which would not be considered a hard exploration problem [40, 97, 98, 100]. Thus, we recommend setting $U > p = \frac{1}{|\mathcal{A}|}$.

5.4.4 Generalization to multi-agent settings and challenges

Our analyses in Sections 5.4.2 and 5.4.3 can be viewed as preliminary steps toward providing guarantees for the multi-agent setting with $N > 1$. The most intuitive approach to generalizing both analyses is to consider the joint actions $a_t = \langle a_{t,1}, ..., a_{t,N} \rangle$, joint locations $s_t = \langle v_{t,i}, ..., v_{t,N} \rangle$, and joint CACTUS regions $\bigcup_{i \in \mathcal{D}} \mathcal{V}^{\text{CACTUS}}(v_{\text{goal},i})$ to formulate a centralized MAPF problem with a single joint controller. In the following, we will discuss the conditions of the generalization and the corresponding challenges that we need to address in the future.

Towards multi-agent optimality To extend Theorem 1 toward multi-agent optimality w.r.t. the flowtime (Section 2.1) and our stochastic game formulation (Section 4), the following conditions must hold:

1. The rewards must be aligned with the MAPF objective, as discussed in Sections 4.1 and 4.2.2 [37, 105].
2. The training instances must be solvable by CACTUS, according to Assumption 5.2. Therefore, the optimal makespan of the training instances must not exceed the allocation radius R_{alloc}. Figure 7 shows a negative example, where the CACTUS regions around the agents' goals would prevent both agents from taking potentially necessary detours in order to solve the task.
3. The problem must be fully observable, i.e., all agents must be able to perceive the whole environment and all agents (1) [106, 107].

Condition **1** is already satisfied by our problem formulation from Section 4, as discussed in Section 4.2 and [86]. Emphasizing other MAPF-relevant aspects through penalties, such as collision avoidance or deadlocks, can change the underlying MARL objective towards more defensive or even lazy behavior, which could deviate substantially from the actually achievable optimal behavior [37, 105].

Condition **2** can be satisfied by generating training instances with optimal makespans that do not exceed R_{alloc}. With that restriction, we can ensure that all agents can theoretically reach their respective goals within R_{alloc} steps. A simple approach towards generating such instances is to simulate a joint random walk of R_{alloc} steps for all agents from their

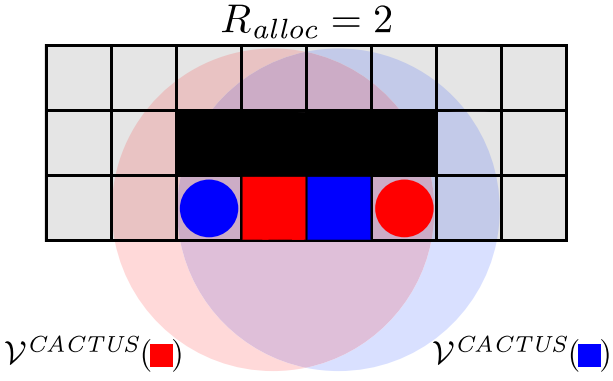

Fig. 7 Example of a solvable instance I with $N = 2$ agents (red and blue circles) and their corresponding goals (red and blue squares) within an allocation radius of $R_{\text{alloc}} = 2$. The shaded circles indicate the absorbing CACTUS regions around the respective goals. According to Assumption 5.2, CACTUS cannot solve this instance because neither agent can leave their CACTUS regions, which prevents them from taking potentially necessary detours to solve the task

start locations $v_{\text{start},i}$ (without any collisions, according to Section 4) and allocate their goals $v_{\text{goal},i}$ at the position of their R_{alloc}th random step, as illustrated in Fig. 8 (right) [21].

Condition **3** can only be satisfied in specific domains, e.g., confined applications like warehouse management [1, 2]. However, in general, the world is messy and uncertain due to noisy sensors [18, 108], limited agent knowledge [23, 37], and unexpected behavior of other agents [3, 26]. Thus, the optimality $Q^* = Q^*_{\text{tot}} = \max_\pi Q^\pi_{\text{tot}}$ of the achievable decentralized behavior π (Section 2.2) is always upper-bounded by the corresponding centralized behavior, where a joint controller acts either under full observability Q^*_{MDP} (MDPs or most search-based MAPF solvers) or partial observability Q^*_{POMDP} (POMDPs with access to all local observations $z_{t,i}$) [107, 108]:

$$Q^* = Q^*_{\text{tot}} \leq Q^*_{\text{POMDP}} \leq Q^*_{\text{MDP}}$$

In theory, we can determine the upper bounds Q^*_{MDP} and Q^*_{POMDP} by solving the corresponding MDP or POMDP formulation, e.g., via the Bellman optimality equation using Dynamic Programming [18, 106, 107]. However, addressing this optimality gap is out of scope for this work and thus will be considered for future work.

Towards multi-agent exploration efficiency To extend Theorem 2 toward multi-agent exploration efficiency, the following conditions must hold:

1. In accordance with the multi-agent optimality, the optimal makespan of the training instances must not exceed the allocation radius R_{alloc}. That means all goals must be reachable within R_{alloc} steps to alleviate the issue of detours, as illustrated in Fig. 7.
2. Given an allocation radius R_{alloc}, the curriculum approach must generate a sufficiently diverse set of instances with a maximum optimal makespan of R_{alloc} to ensure generalizing training with various potential paths; ideally all C joint paths.

Similar to the multi-agent optimality, discussed above, condition **1** can be satisfied by generating training instances with an optimal makespan that does not exceed R_{alloc}, e.g., by simulating joint random walks, as illustrated in Fig. 8 (right) [21].

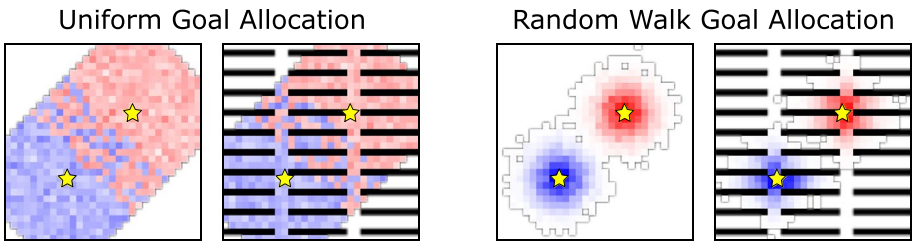

Fig. 8 Illustration of two goal allocation strategies given some allocation radius R_{alloc} and an empty and warehouse corridor map. The shading intensity (blue, red) indicates the empirical frequency of goal allocations. Left: The uniform strategy is used in this paper and allocates goals randomly within their corresponding CACTUS regions. The resulting instances might not be solvable by CACTUS, according to Assumption 5.2. Right: The random walk strategy simulates a joint random walk of all agents from their respective start locations and places the goals at the corresponding R_{alloc}th random step. The resulting instances are always solvable by CACTUS but less diverse

Naive joint random walks can guarantee training instances that are solvable by CACTUS, according to Assumption 5.2. However, these instances are biased towards short-range navigation, with goals typically placed near the start locations, as shown in Fig. 8 (right). To satisfy condition **2**, we need to investigate allocation strategies that can ensure solvability but are also sufficiently diverse like our uniform goal allocation strategy used in Algorithm 1, as illustrated in Fig. 8 (left). Such an investigation is out of scope for this paper and thus will be considered for future work.

5.4.5 Decentralized multi-agent coordination

Under certain conditions, discussed in Section 5.4.4, our analyses in Sections 5.4.2 and 5.4.3 can be generalized to multi-agent systems by considering the joint actions $a_t = \langle a_{t,1}, ..., a_{t,N} \rangle$ and joint locations $s_t = \langle v_{t,i}, ..., v_{t,N} \rangle$. However, most prior works use independent learning, where single-agent RL algorithms are applied to each agent separately [19, 29, 44, 45, 83]. *Independent RL* is not guaranteed to converge in general due to the *non-stationarity* caused by simultaneously learning agents [109, 110]. Even if all agents learn shortest-path policies, the resulting joint policy is not guaranteed to be optimal due to potential conflicts and infeasibility [12, 27, 83].

To alleviate the coordination problem, most prior works rely on extensive engineering by specializing in very specific aspects, such as congestion [44], blocking [19], communication [45], periodic neighborhood sensing [29], etc. While these specializations work empirically well for particular settings, most of them are hardly transferable to new scenarios and may even interfere with each other. For example, rewards may be shaped to avoid corridors. However, if agents can synchronize via communication and find more efficient paths through the corridors, the rewards may be limiting and need re-adjustment. Constantly keeping track and re-adjusting all of these mechanisms is neither tractable nor desirable in the long run, according to *The Bitter Lesson* [46].

Fortunately, modern MARL techniques offer more general ways of addressing multi-agent coordination via credit assignment mechanisms (Section 2.2.2). For example, when the value functions $\hat{Q}_i$ are learned jointly via $\hat{Q} = \sum_{i \in \mathcal{D}} \hat{Q}_i \approx Q^{\hat{\pi}}_{\text{tot}}$ using an IGM consistent factorization operator Ψ, according to (4) and (5), we can ensure that the decentralized maximization of $\hat{Q}_i$ is equivalent to the joint maximization of $\hat{Q}$ and thus effective coordination of the resulting local policies $\hat{\pi}_i$.

We now regard our MAPF objective, according to Section 4 and (1), w.r.t. potential factorization operators Ψ (Section 2.2.2):

Proposition 1 *The sum of values $Q^{\pi}_{\text{tot}}(s_t, a_t) = \sum_{i \in \mathcal{D}} Q^{\pi}_i(s_t, a_t)$ w.r.t. the expected flowtime for all states $s_t \in \mathcal{S}$ and joint actions $a_t \in \mathcal{A}$, according to Section 4 and (1), satisfies the additivity as well as the monotonicity constraint $\frac{\delta Q^{\pi}_{\text{tot}}}{\delta Q^{\pi}_i} \geq 0$.*

Proof The sum of values $Q^{\pi}_{\text{tot}}(s_t, a_t) = \sum_{i \in \mathcal{D}} Q^{\pi}_i(s_t, a_t)$ is a linear operation, where any increase (or decrease) in Q^{π}_i leads to a monotonic increase (or decrease) in Q^{π}_{tot}. □

According to Proposition 1, we can use simple and efficient factorization operators, such as VDN [58] and QMIX [39], as described in Section 2.2.2, to learn coordinated policies

that optimize the sum of values or expected flowtime, according to Section 4 and (6). More general and advanced techniques, such as QPLEX, can also be used, but they increase complexity and computational demand and offer limited prospects for improving performance, as we will demonstrate in Section 6.2.4. Note that the rewards must be aligned with the original MAPF objective, as discussed in Section 4, and cannot be modified arbitrarily without compromising optimality and reliability [88].

The makespan satisfies the monotonicity constraint but not the additivity constraint, which we do not consider further in this paper.

Note that the incorporation of multi-agent coordination techniques, such as credit assignment mechanisms, is necessary but not sufficient to ensure multi-agent optimality of CACTUS, as discussed in Section 5.4.4.

5.4.6 Further limitations

Absorbing CACTUS regions Assumption 5.2 depends on the confidence level, specified by the deviation factor η, that an agent can reach its goal with a chance of at least U, according to Algorithm 1. If the confidence level is too low, there is too much uncertainty about the true goal-reaching probability and, consequently, about the exploration efficiency of π^{CACTUS}. Our experiments in Section 6.2.3 suggest that η should be specified such that the confidence level is at least 95%, i.e., $\eta > 1$.

Distance function To ensure the validity of our exploration efficiency analysis, the distance function $\mathrm{DIST} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ must not underestimate the shortest path length. Using an underestimating distance function for R_{alloc}, such as the Chebyshev distance, cannot guarantee the reachability of a goal location within $H \geq R_{\mathrm{alloc}}$ steps, e.g., due to potential detours, even if the agent is inside the CACTUS region $\mathcal{V}^{\mathrm{CACTUS}}(v_{\mathrm{goal}})$.

Variable number of agents Our curriculum scheme and analyses assume a fixed number of agents N. Therefore, the values learned are only valid for N. While function approximators like deep neural networks can generalize to scenarios with different numbers of agents, as demonstrated in [19, 29, 44, 45, 83], there is no theoretical support for arbitrary scaling yet.

5.5 Conceptual discussion

CACTUS represents a simple reverse curriculum scheme inspired by [42, 70]. Our work focuses on the MAPF problem, where we have multiple agents with different start and goal locations that can vary per instance I. To ensure generalization over many different MAPF instances and maps, our scheme adjusts the random allocation of goals around the random start locations, as illustrated in Fig. 8 (left).

In contrast to prior works [84, 85], CACTUS does not separate learning of different skills like navigation and collision avoidance. As illustrated in Fig. 1, all agents first need to focus on reaching their goals, which are allocated in close proximity within R_{alloc}. With increasing R_{alloc}, the allocation areas of different agents may overlap, as shown in Fig. 4. This automatically causes agents to interact with each other, thus increasing the conflict potential and coordination pressure [21]. The conflict potential can be further amplified with C^3

LARA by sampling goal locations of intersection areas with higher probability, as specified in (8), without additional hyperparameters or engineering.

R_{alloc} is incremented only when the agents can coordinate and reach their goals with sufficiently high confidence, which is checked via statistical significance testing using the hyperparameters U and η. Thus, CACTUS offers an adaptive approach to MAPF problems via MARL without explicit separation of agent skills [40, 84, 85], shaping of rewards [19], or expensive acquisition of expert data [44, 45]. Since the goals are initialized randomly within the allocation radius R_{alloc} around the agents' start locations, they can still be allocated in closer proximity to the agents, which alleviates catastrophic forgetting of easier tasks that the agents have mastered before.

The simplicity of CACTUS allows us to delve into its theoretical properties, in contrast to prior methods, which are heavily engineered and too complicated for proper analysis. By assuming U to be a lower bound probability of reaching goals within their CACTUS regions, as well as absorption within these regions, we can expect considerably higher exploration efficiency for hard exploration problems in the single-agent case, as well as the multi-agent case if the maximum makespan is R_{alloc} and multi-agent coordination techniques like credit assignment mechanisms are used.

While CACTUS can preserve the optimality of single-agent RL algorithms, it cannot yet guarantee optimality for the multi-agent case, due to partial observability and potential instances that cannot be solved by absorbed agents. The former is an open challenge that requires further investigation in future work. The latter can be addressed via goal assignments that ensure reachability of all goals with a maximum makespan of R_{alloc} without any detours outside the absorbing CACTUS regions, as illustrated in Fig. 8 (right). In practice, CACTUS can still perform well compared to prior methods, especially in unstructured maps.

By using suitable function approximators, such as graph neural networks, CACTUS and C^{3}LARA can both be straightforwardly applied to any kind of undirected unweighted graph G beyond gridworlds, as neither our problem formulation in Section 4 nor our methods and theoretical analyses in Section 5 explicitly depends on grid structures, unlike certain heuristics used in search-based MAPF solvers [13, 17]. However, given the prevalence of gridworld benchmarks for MAPF algorithms [9, 19, 111], our experimental evaluation focuses on gridworlds for direct comparability with prior learning- and search-based methods.

6 Experiments

6.1 Setup

6.1.1 Maps and instances

The *training maps* are randomly generated, according to [19], and have different shapes $K \times K$ defined by *map size* $K \in \{10, 40, 80\}$. The *obstacle density* $\delta \in \{0, 0.1, 0.2, 0.3\}$ defines the fraction of non-occupiable locations in the maps. All agents start at random locations with randomly assigned goals, according to an allocation radius $R_{\text{alloc}} \geq 1$. If $R_{\text{alloc}} = \infty$, then the goals can be placed anywhere on the training map.

The *test maps* are provided by [19]. For each map configuration of size K and obstacle density δ, there are 100 pre-generated test instances I with fixed start and goal locations for all agents w.r.t. $R_{\text{alloc}} = \infty$ to ensure a fair comparison between different approaches.

We always set $\gamma = 1$, as suggested in Section 4.

6.1.2 Algorithms and training

We implement PPO for policy learning (3) and VDN, QMIX, QPLEX, and MAPPO for critic learning. We implement a purely RL-based version of PRIMAL using the same reward function as defined in [19] (Table 3). In addition, we employ a naive baseline, called *No Curriculum*, which consists only of PPO and QMIX without any shaped reward. PRIMAL and *No Curriculum* are trained with $R_{\text{alloc}} = \infty$, according to Section 6.1.1. We choose QMIX as our *No Curriculum* baseline for consistency with our default setting. Replacing QMIX with VDN, QPLEX, or MAPPO does not notably affect the performance of this baseline due to the generally limited exploration capabilities of non-curriculum approaches, according to Section 5.4.3.

We train all algorithms on different training maps, as explained in Section 6.1.1, with $N = 8$ agents over 5000 epochs, each consisting of $E = 32$ episodes. The maps of size $K = 10$ are sampled twice as often as the other map sizes, as proposed in [19]. Each episode terminates after all agents reach their goal or after $T = 256$ time steps. All algorithms use parameter sharing, i.e., where all agents use the same policy and individual critic networks $\hat{\pi}_i$ and $\hat{Q}_i$, respectively.

We denote *CACTUS (X)* as CACTUS (or C^{3}LARA, respectively) using MARL algorithm X for critic learning. Unless stated otherwise, CACTUS and C^{3}LARA always use PPO for policy and *X=QMIX* for critic learning.

In addition, we run *CBSH* as a slow but optimal search-based MAPF solver with a runtime limit of 5 minutes [11] and *MAPF-LNS* [15] as a fast but unbounded suboptimal anytime MAPF solver with a runtime limit of 1 minute.

6.1.3 Neural networks and hyperparameters

For CACTUS, C^{3}LARA, and *No Curriculum*, we use deep neural networks to implement $\hat{\pi}_i$ and $\hat{Q}_i$ for each agent i and factorization operator Ψ for QMIX and QPLEX. The neural networks are updated after every $E = 32$ episodes using ADAM with a learning rate of 0.001.

Since all regarded maps are gridworlds, the observations are encoded as multi-channel image, as illustrated in Fig. 2. We implement all neural networks as *multilayer perceptrons (MLP)* and flatten the multi-channel images before feeding them into the networks. $\hat{\pi}_i$ and $\hat{Q}_i$ have two hidden layers of 64 units with ELU activation. The output of $\hat{\pi}_i$ has $|\mathcal{A}_i|$ units with softmax activation. The output of $\hat{Q}_i$ has $|\mathcal{A}_i|$ linear units. The hypernetworks of QMIX, as well as the critic of MAPPO, have two hidden layers of 128 units with ELU activation and one or $|\mathcal{A}_i|$ linear output units, respectively. For PRIMAL, we use the same architecture as proposed in [19].

Unless stated otherwise, CACTUS always uses a threshold of $U = 75\%$ and a deviation factor of $\eta = 2$, which corresponds to a confidence level of about 97% in one-tailed tests.

6.1.4 Computing infrastructure

All training and test runs are performed on a x86_64 GNU/Linux (Ubuntu 18.04.5 LTS) machine with i7-8700 @ 3.2GHz CPU (8 cores) and 64 GB RAM. Due to the simplicity of CACTUS, we do not need any GPU or distributed HPC infrastructure in contrast to [19, 44, 45].

6.2 Results

For each experiment, all respective algorithms are run 10 times to report the average progress and the 95% confidence interval. We evaluate the training progress and generalization of the trained policies with the pre-generated test instances I explained in Section 6.1.1. Since CACTUS and C^{3}LARA progress similarly during training, we report only the learning curves of CACTUS and provide the results of C^{3}LARA in the generalization experiments in Section 6.2.4.

6.2.1 Simplicity of CACTUS

To demonstrate the simplicity of CACTUS, we first quantify the training time and data w.r.t. the number of episodes, the number of trainable parameters, and the reward complexity and compare them with the original PRIMAL, as specified in [19].

An overview is given in Table 1. In almost all aspects, CACTUS requires only 5% or less of the effort of the original PRIMAL, therefore being clearly the simpler and more efficient MARL approach to MAPF. Unlike PRIMAL, CACTUS is only run on CPU while still requiring significantly less training time. Furthermore, CACTUS does not depend on any expert data, i.e., recommendations of a centralized search-based MAPF solver, which saves a considerable amount of compute. While the reward function of PRIMAL requires four penalty terms for very specific situations, CACTUS is trained with a very simple reward function that penalizes any time step unless the goal is reached without manually considering specific cases, as discussed in Section 4.

Figure 9 compares the number of trainable parameters and neural network architectures of PRIMAL and CACTUS. In CACTUS, the critic with the mixing network of QMIX has the majority of trainable parameters, which are only required during training. The actor size is negligible in CACTUS, which enables significantly faster inference than PRIMAL. The network architecture of CACTUS is also much simpler than PRIMAL since it is only based on MLPs and thus does not depend on specialized hardware or significant computational effort for fine-tuning and training [20].

Table 1 Comparison of the original PRIMAL and CACTUS w.r.t. various aspects in our experiments. The last column shows the amount of effort relative to the original PRIMAL. The numbers of the original PRIMAL are from [19]. Unlike [19], we do not use any GPU or expert data for training

	PRIMAL (original [19])	CACTUS	Relative effort to PRIMAL
Training Time	$\approx$ 20 days	$\approx$ 1 day	$\approx 5\%$
# Training Episodes	$\approx$ 3.8 million	160, 000	$\approx 4.2\%$
# Parameters	$\approx$ 13 million	579, 979	$\approx 4.5\%$
# Reward Penalties	4	1	25%

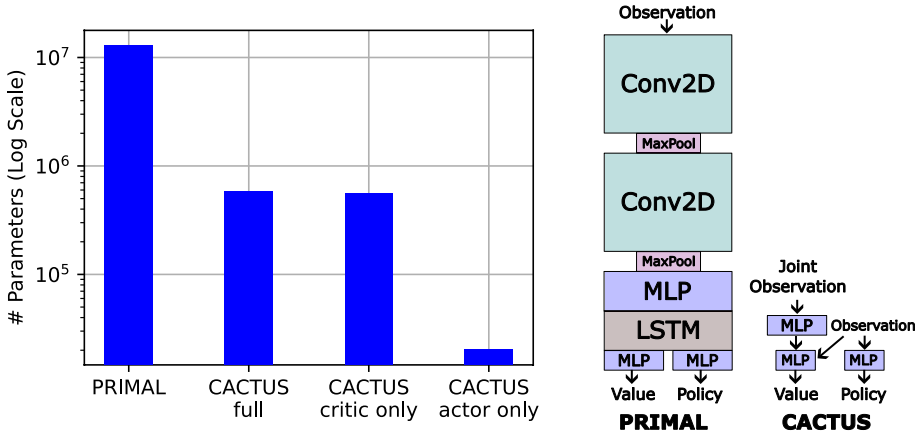

Fig. 9 Left: Comparison of the number of trainable parameters in PRIMAL and CACTUS. Note the logarithmic scale on the y-axis. Right: Schematic network architectures used for PRIMAL and CACTUS. The sizes do not reflect any quantity and are intended only to illustrate the components used for learning

6.2.2 Curriculum learning

We evaluate the effect of CACTUS using QMIX and MAPPO as current state-of-the-art MARL techniques [39, 43, 61]. After every epoch, we measure the average completion rate w.r.t. all test instances I with map size $K \in \{10, 40, 80\}$ as well as obstacle density $\delta \in \{0, 0.1, 0.2, 0.3\}$ for $N = 8$ agents. The results are shown in Fig. 10. *CACTUS (QMIX)* always performs best. PRIMAL outperforms *CACTUS (MAPPO)*. *No Curriculum* fails to learn meaningful policies. However, PRIMAL is barely able to outperform *No Curriculum*

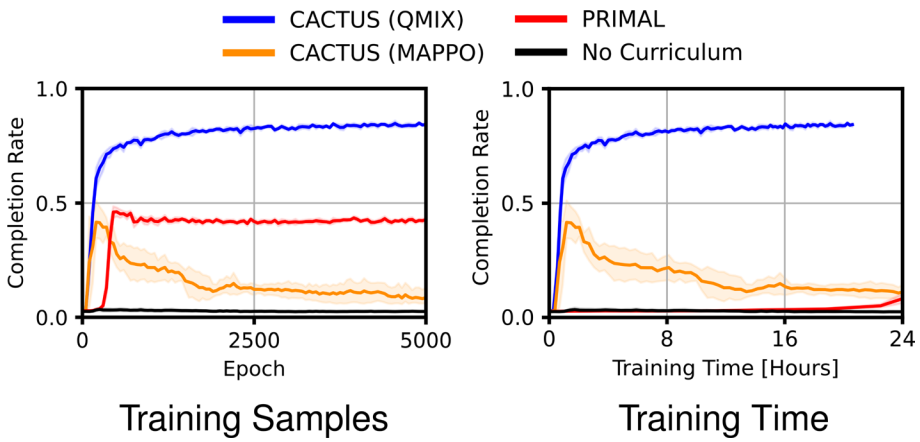

Fig. 10 Average training progress of CACTUS variants, PRIMAL, and a naive MARL baseline without any curriculum w.r.t. training epochs (left) and training time (right). The performance is evaluated on all pre-generated test instances I of [19] with $K \in \{10, 40, 80\}$, $\delta \in \{0, 0.1, 0.2, 0.3\}$, and $N = 8$ agents. Shaded areas show the 95% confidence interval

after 24 hours of training. *CACTUS (QMIX)* is the fastest approach, completing training below 24 hours.

6.2.3 Hyperparameter and ablation study

CACTUS hyperparameters Next, we evaluate the impact of different curriculum decision thresholds $U \in \{0.25, 0.5, 0.75\}$ and deviation factors $\eta = \{1, 2, 3\}$ on CACTUS. After every epoch, we measure the average completion rate w.r.t. all test instances I with map size $K \in \{10, 40, 80\}$ as well as obstacle density $\delta \in \{0, 0.1, 0.2, 0.3\}$ for $N = 8$ agents. For the deviation factor evaluation, we consider CACTUS with $U = 0.25$, since all variants with $U = 0.75$ perform very similarly, as shown in Fig. 10. The results are shown in Fig. 11. CACTUS performs best with $U = 0.75$ and second best with $U = 0.5$. CACTUS with $U = 0.25$ performs best when $\eta = 2$ and second best with $\eta = 3$. All CACTUS variants clearly outperform PRIMAL.

Training with manual curriculum schedules We also compare our confidence-based curriculum update rule from Section 5.2 and Algorithm 1 with alternative manual curricula with a hyperparameter $K \geq 1$. We consider fixed-interval updates to the allocation radius R_{alloc} after K epochs, and doubling-interval updates, starting with K epochs and doubling the interval after each radius update. The latter is motivated by the growing CACTUS regions, which could require more exploration time. The results are shown in Fig. 12 (left). None of the manual curricula outperforms CACTUS. The fixed-interval baselines require $K \geq 100$ epochs between the radius updates for notable effectiveness. The doubling-interval baselines generally underperform the fixed-interval baselines, indicating that the policies overfit to the current curriculum stage, whereas our confidence-based rule grows the CACTUS regions at an adequate pace without optimistic overshooting (i.e., increasing R_{alloc} too fast) and premature overfitting (i.e., increasing R_{alloc} too slowly)

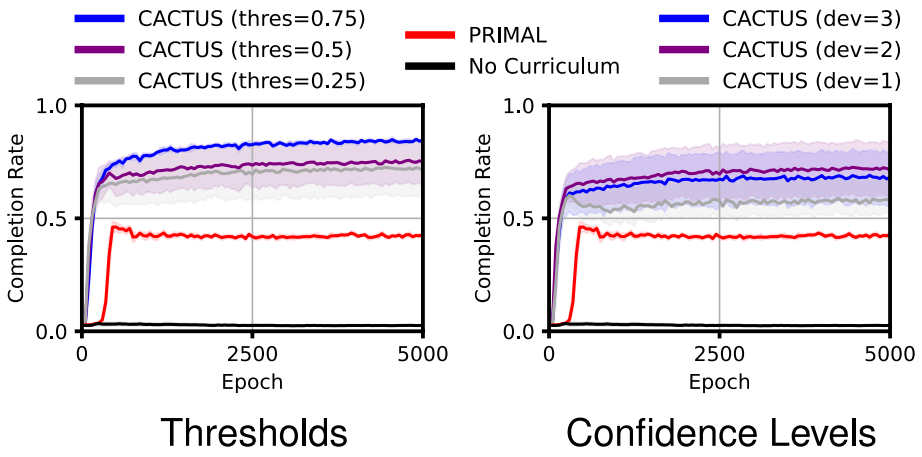

Fig. 11 Average training progress of CACTUS variants, PRIMAL, and a naive MARL baseline without any curriculum w.r.t. different curriculum decision thresholds U (left) and deviation factors η (right). The performance is evaluated on all pre-generated test instances I of [19] with $K \in \{10, 40, 80\}$, $\delta \in \{0, 0.1, 0.2, 0.3\}$, and $N = 8$ agents. The right plot shows CACTUS variants with $U = 0.25$. Shaded areas show the 95% confidence interval

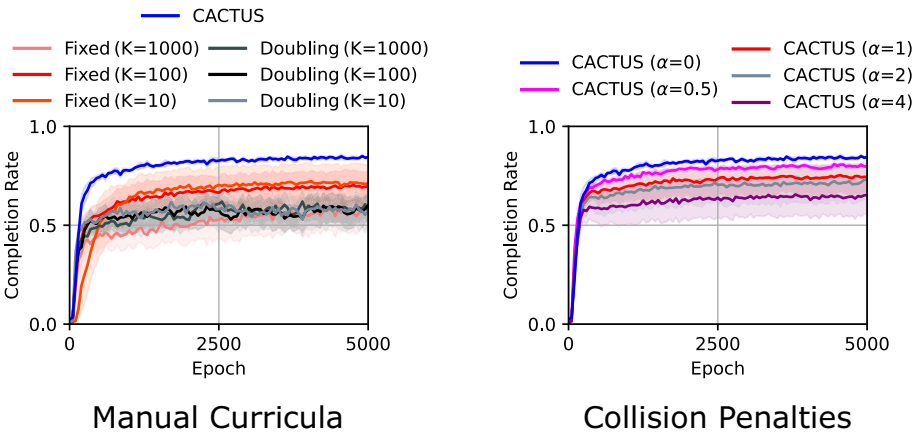

Fig. 12 Left: Average training progress of CACTUS and alternative manual curricula with a hyperparameter $K \in \{10, 100, 1000\}$. We consider fixed-interval updates to the allocation radius R_{alloc} after K epochs, and doubling-interval updates, starting with K epochs. Right: Average training progress of CACTUS variants with different penalty values $\alpha \in \{0, 0.5, 1, 2, 4\}$ for collisions in addition to the time penalty from Section 4

Training with collision penalties In addition, we compare CACTUS using our reward definition from Section 4.1 with variants that use explicit collision penalties $\alpha \geq 0$ in addition to the time penalty from Section 4. The results are shown in Fig. 12 (right). Using explicit collision penalties leads to degrading performance, indicating more defensive or even lazy behavior from the learned policies, as agents tend to prioritize collision avoidance over reaching their goals when the collision penalty α is sufficiently high (Section 4.2).

6.2.4 Generalization of CACTUS and C^{3}LARA

Next, we evaluate the generalization capabilities of the policies trained with CACTUS, C^{3}LARA, PRIMAL, and *No Curriculum*. All policies are trained for 5000 epochs of 32 episodes each before being evaluated on all test instances I with map size $K \in \{40, 80\}$ and obstacle density $\delta \in \{0, 0.1, 0.2, 0.3\}$ with different numbers of agents N. Note that all policies have only been trained with $N = 8$ agents, according to Section 6.1.2. We also report the average performance of the centralized search-based MAPF solvers *CBSH* and *MAPF-LNS*.

State-of-the-Art comparison The average completion rates of CACTUS, C^{3}LARA, PRIMAL, *No Curriculum*, and the centralized search-based MAPF solvers are shown in Fig. 13. CACTUS and C^{3}LARA generalize best compared to PRIMAL and *No Curriculum*, with C^{3}LARA achieving slightly higher completion rates when N increases. In test instances with $\delta \leq 10\%$, C^{3}LARA consistently achieves an average completion rate of over 80% when scaling up to $N = 64$ agents, while CACTUS achieves at least 60%. PRIMAL is always outperformed by CACTUS and C^{3}LARA but consistently outperforms *No Curriculum*. All approaches perform poorly when the agent number is $N \geq 256$ or $\delta > 20\%$. MAPF-LNS always achieves a perfect completion rate of 1, while CBSH only achieves competitive completion rates when $N \geq 16$.

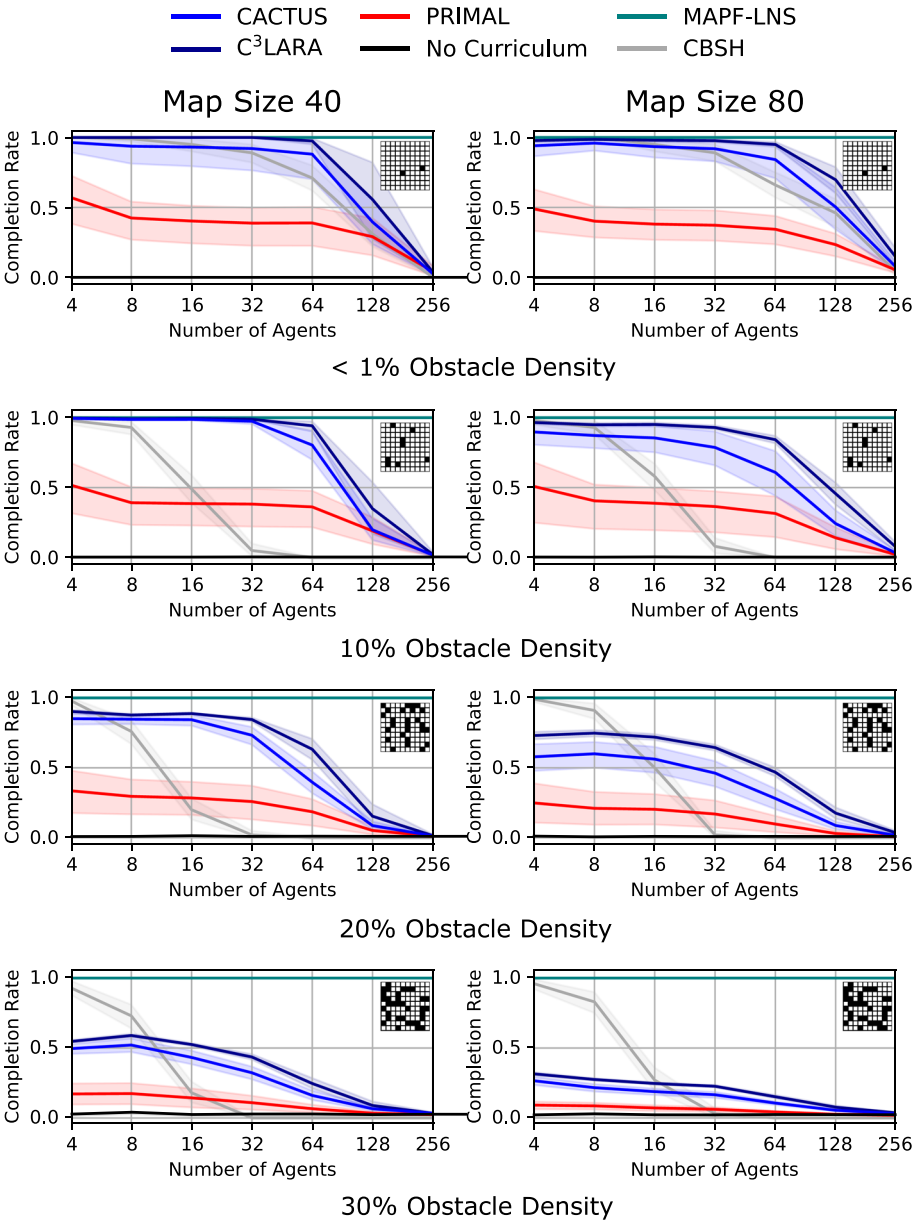

Fig. 13 Average completion rate of CACTUS and C^{3}LARA as well as MARL and MAPF baselines of all test instances I of size $K \in \{40, 80\}$ and obstacle density $\delta \in \{0, 0.1, 0.2, 0.3\}$ w.r.t. different numbers of agents N. The icons in the top-right corner of each plot show a 10×10 sub-grid example to illustrate the obstacle density. Shaded areas show the 95% confidence interval

The average episode lengths (capped at horizon $T = 256$) of CACTUS, C^{3}LARA, PRIMAL, *No Curriculum*, and the centralized search-based MAPF solvers are shown in Fig. 14 for all test instances I of size $K = 40$. Unsolved instances are always terminated after T time steps. CACTUS and C^{3}LARA achieve the lowest average episode lengths of all MARL approaches, with C^{3}LARA consistently achieving lower episode lengths than CACTUS. PRIMAL achieves longer episode lengths with higher variance due to completing fewer episodes, especially when $N \leq 64$. *No Curriculum* is the worst-performing MARL approach, always achieving the maximum episode length of $T = 256$. MAPF-LNS performs similarly to the MARL approaches when $\delta < 1\%$, but its average episode length increases faster in other settings. Despite MAPF-LNS always achieving a perfect completion rate of 1, as shown in Fig. 13, the resulting plans require a larger horizon T for completion when $N \geq 16$. CBSH always achieves a competitive or lower episode length when $N \leq 8$ and $\delta \geq 20\%$.

Alternative value factorization The average completion rates of CACTUS variants using VDN, QMIX, and QPLEX, as well as PRIMAL, are shown in Fig. 15 for all test instances I of size $K = 40$. *CACTUS (VDN)* and *CACTUS (QMIX)* achieve the highest completion rates in most cases, with *CACTUS (QPLEX)* outperforming PRIMAL when $N \leq 128$.

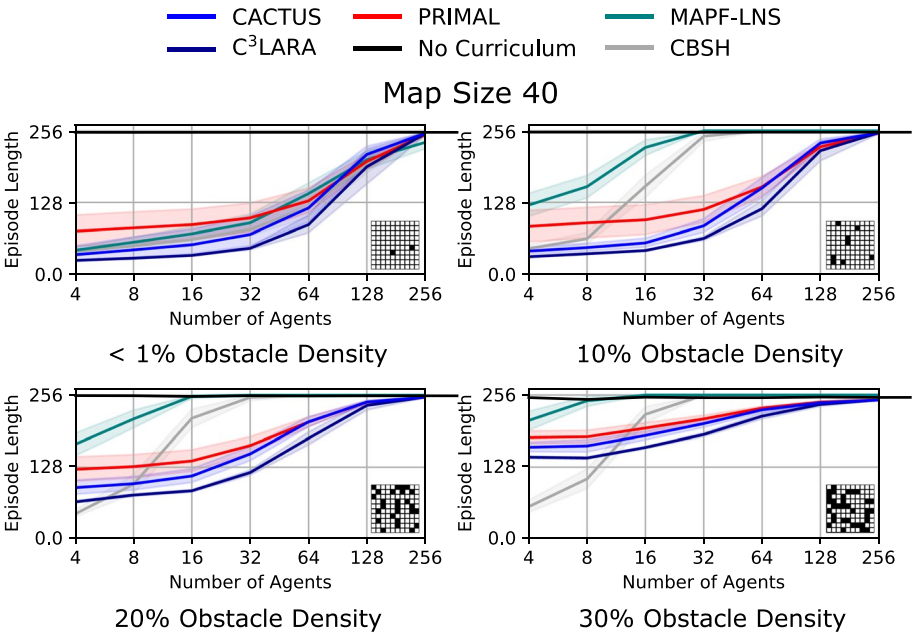

Fig. 14 Average episode length (capped at horizon $T = 256$) of CACTUS and C^{3}LARA as well as MARL and MAPF baselines of all test instances I of size $K = 40$ and obstacle density $\delta \in \{0, 0.1, 0.2, 0.3\}$ w.r.t. different numbers of agents N. The icons in the top-right corner of each plot show a 10×10 subgrid example to illustrate the obstacle density. Shaded areas show the 95% confidence interval

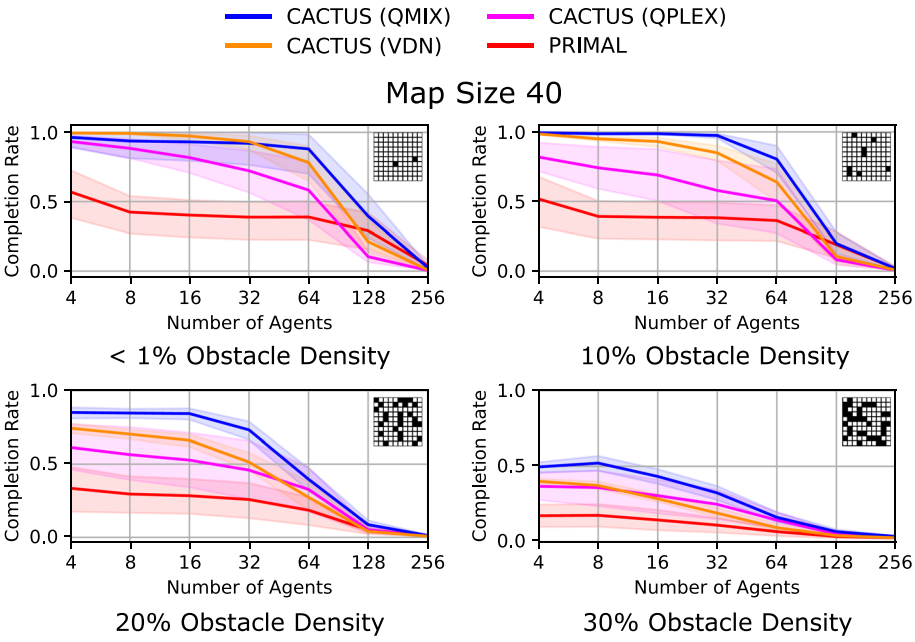

Fig. 15 Average completion rate of CACTUS variants and PRIMAL of all test instances I of size $K = 40$ and obstacle density $\delta \in \{0, 0.1, 0.2, 0.3\}$ w.r.t. different numbers of agents N. The icons in the top-right corner of each plot show a 10×10 sub-grid example to illustrate the obstacle density. Shaded areas show the 95% confidence interval

6.2.5 Limitations in structured maps

Training progress in structured maps Furthermore, we tested CACTUS, PRIMAL, and *No Curriculum* in structured maps with rooms and narrow corridors, such as mazes. While training was conducted on randomly generated maps, as explained in Section 6.1.1, we evaluated the training progress using all maze and room maps of the common MAPF benchmark from [9]. The results are shown in Fig. 16. Compared to the results for unstructured maps in Section 6.2.2, all approaches perform significantly worse, with none of them achieving an average completion rate above 25%. CACTUS consistently outperforms all other approaches, which generally fail to complete more than 10% of all agent tasks.

Evaluation on structured small-scale instances Finally, we evaluated the learned behaviors of CACTUS, C^{3}LARA, and PRIMAL in more depth using a set of small-scale test instances with $N = 2$ agents, as depicted in Fig. 17. We tested all policies evaluated in Section 6.2.4, ensuring that none had encountered these test instances during training, according to our setup described in Section 6.1. The test instances are inspired by prior literature:

- **A**: Both agents must swap positions [112].
- **B**: Both agents must coordinate to move through a bottleneck node [33].
- **C**: The blue agent must move into the alcove to let the red agent pass [113].

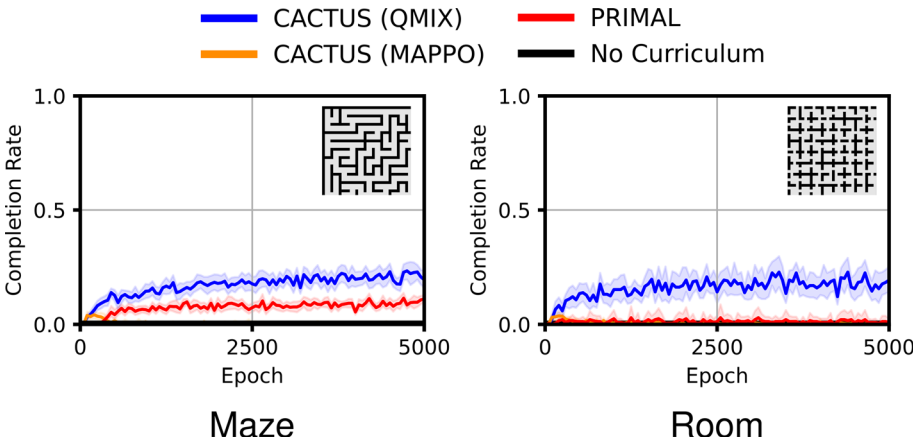

Fig. 16 Average training progress of CACTUS variants, PRIMAL, and a naive MARL baseline without any curriculum in structured maps. The performance is evaluated on all maze and room maps provided by [9], respectively

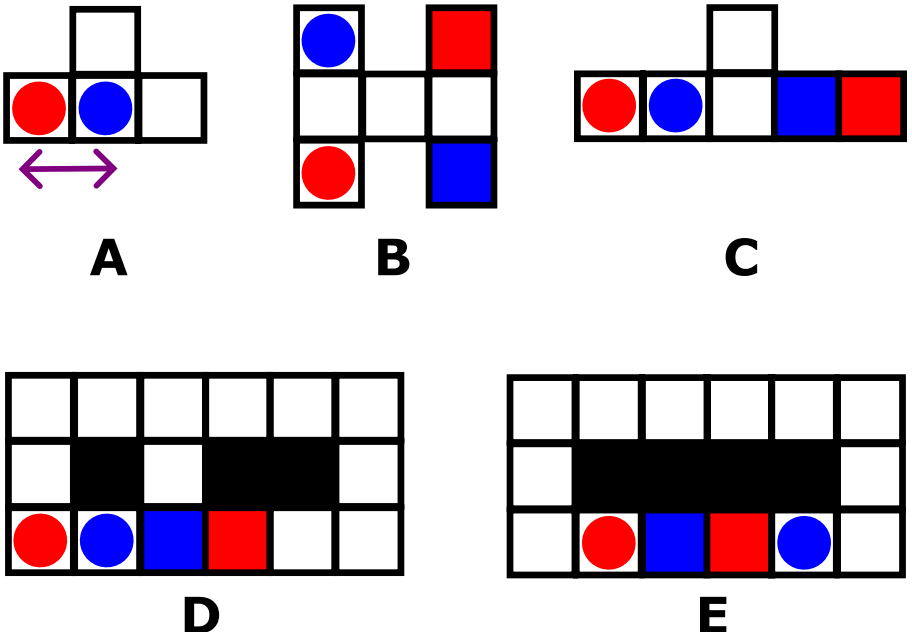

Fig. 17 Different small-scale test instances with $N = 2$ agents for evaluating the learned behaviors of CACTUS, C^{3}LARA, and PRIMAL in structured scenarios

- **D**: The blue agent has to let the red agent pass. Alternatively, the red agent could make a detour to reach its goal.
- **E**: One of the agents has to make a detour to reach its goals, inspired by Fig. 7.

Table 2 Average success rates, i.e., the rate of both agents reaching their goals, with 95% confidence intervals of the learned behaviors of CACTUS, C^3LARA, and PRIMAL (Section 6.2.4) on the small-scale test instances from Fig. 17. The best results per instance are highlighted in boldface and blue

Instance	CACTUS	C^3LARA	PRIMAL
A	0.99 ± 0.59	$\mathbf{1.0 \pm 0}$	0.99 ± 0.22
B	0.65 ± 0.38	$\mathbf{0.69 \pm 0.45}$	0.66 ± 0.15
C	0.70 ± 0.41	$\mathbf{0.79 \pm 0.52}$	0.75 ± 0.16
D	0.30 ± 0.18	$\mathbf{0.42 \pm 0.28}$	0.41 ± 0.09
E	$\mathbf{0.36 \pm 0.21}$	0.33 ± 0.22	0.34 ± 0.08

The average success rates, i.e., the rate of both agents reaching their goals, with 95% confidence intervals are displayed in Table 2. All approaches achieve an average success rate of at least 65% in the instances A, B, and C. However, all approaches achieve poor success rates (below 50%) in the instances D and E, which entail conflicts at goal locations, thus requiring detours or yielding. Such scenarios are common in structured maps, such as mazes or rooms, and are particularly difficult for CACTUS and C^3LARA, as discussed in Section 5.4.4. C^3LARA always achieves the highest average success rate except in the instance E, where CACTUS performs best. Since these test instances have very short navigation distances, PRIMAL can compete with CACTUS and C^3LARA due to less necessary exploration.

6.3 Discussion

Our results confirm the necessity of adequate curricula, as standard MARL methods without any curriculum would likely fail to learn meaningful policies in our MAPF setting, according to our analysis in Section 5.4.3 and our problem formulation in Section 4. This is due to the sparse rewards and dynamic constraints of the problem formulation. The shaped reward function of PRIMAL is helpful in improving performance over standard MARL. Because of the many handcrafted penalty terms listed in Table 3, PRIMAL policies are rather conservative, which leads to a performance plateau that cannot be overcome without imitation learning on suitable expert data. In particular, collision penalties encourage defensive or even lazy behavior, as discussed in Section 4.2 and demonstrated in our experimental results from Section 6.2.3.

However, CACTUS with value factorization clearly outperforms PRIMAL with 95% less training time and learnable parameters without any additional reward shaping or expert data. CACTUS with MAPPO performs poorly, suggesting the importance of adequate credit assignment mechanisms, such as value factorization, for multi-agent coordination, according to our analysis in Section 5.4.5. The comparison of VDN, QMIX, and QPLEX confirms that VDN and QMIX are sufficient because our MAPF objective is additive and monotonic. Since VDN does not use additional function approximators, e.g., neural networks, it cannot exploit global state information like QMIX, thus being less expressive. QPLEX, on the other hand, appears too complicated for our MAPF objective, requiring more data and training time to achieve competitive performance. Note that the linearity of our objective does not imply that MAPF problems are easy to solve, as they remain NP-hard. We argue that exploration poses the *actual challenge*, which is effectively addressed by CACTUS.

CACTUS is robust w.r.t. the choice of hyperparameters, as any configuration of the curriculum decision threshold $U \geq 25\% > p = \frac{1}{|\mathcal{A}_i|}$ and deviation factor $\eta \geq 1$ outperforms PRIMAL, as shown in Fig. 11, supporting our recommendation for U in Section 5.4.3. Choosing a confidence level that is too low could result in less effective policies due to higher uncertainty about the decision threshold U, as discussed in Section 5.4.6, while choosing a confidence level that is too high, e.g., using $\eta > 3$, could result in slow curriculum updates, where agents may overfit on easy tasks. The adaptive curriculum updates of CACTUS are more effective than alternative manual curriculum schedules, as demonstrated in Section 6.2.3. This is due to an adequate pace of growing CACTUS regions, without optimistic overshooting (i.e., increasing R_{alloc} too fast) and premature overfitting (i.e., increasing R_{alloc} too slowly).

Despite the relatively restrictive time and data budget compared to the original PRIMAL, CACTUS, with value factorization, generalizes quite well over different numbers of agents N and map sizes K. CACTUS with QMIX scales up to instances with 8 to 16 times more agents than used during training, in contrast to standard MARL, which generally fails to learn meaningful policies in the MAPF problem. C^{3}LARA is able to outperform CACTUS when the agent density is sufficiently high due to the increased conflict potential in its weighted goal sampling scheme. This is further supported by our small-scale study in Section 6.2.5, where C^{3}LARA outperforms CACTUS in most test instances, in which agents must swap places or temporarily move aside (e.g., from their own goal location) to let another agent pass.

CACTUS and C^{3}LARA struggle in maps with high obstacle density $\delta \geq 20\%$, where they are outperformed by the optimal CBSH algorithm in small-scale instances with $N \leq 8$ agents. In such cases, as well as in structured maps like mazes or rooms, the agents must take longer detours to avoid conflicts and reach their goals. According to Section 5.4.6, the absorbing property of the CACTUS regions prevents such detours, resulting in worse average performance and scalability.

7 Conclusion

We presented CACTUS as a lightweight MARL approach to decentralized MAPF. CACTUS defines a simple reverse curriculum scheme, where the goal of each agent is randomly placed within an allocation radius around the agent's start location. The allocation radius increases gradually as all agents improve, which is assessed by a confidence-based measure. In addition, we proposed C^{3}LARA as an extension, using weighted sampling of goal locations to improve conflict resolution in scenarios of high agent density, without additional hyperparameters or engineering.

We analyzed the theoretical properties of CACTUS, showing that it can significantly improve exploration efficiency in hard exploration domains, in contrast to prior works that use random exploration. CACTUS can be neatly combined with value factorization techniques, such as VDN or QMIX, since our MAPF objective is additive and monotonic. While CACTUS can preserve the optimality in single-agent RL settings, it cannot guarantee optimality for the multi-agent setting yet, due to partial observability and potential instances that cannot be solved by agents that are absorbed by their corresponding CACTUS regions. This

can be addressed via diverse goal assignments that ensure the reachability of all goals with a maximum makespan within the absorbing CACTUS regions.

Our experimental results support our theoretical findings and demonstrate the simplicity, efficiency, effectiveness, and generalization capabilities of CACTUS compared to alternative MARL approaches and centralized search-based MAPF solvers. Inspired by *The Bitter Lesson* by Sutton [46], CACTUS and C^{3}LARA demonstrate how simple and well-defined curricula can enhance MARL techniques for MAPF in a general way without relying on extremely large neural networks, extensively shaped reward functions, or centralized search-based MAPF solvers for imitation learning. Therefore, we hope to provide a suitable and more general foundation for faster and sustainable progress in MARL-based decentralized MAPF at significantly lower cost.

Appendix A: Reward function comparison

PRIMAL uses a rather complex reward function with different penalty terms for very specific situations like time steps, wait actions, collisions, or blocking of other agents [19]. An A* algorithm is run to determine a blocking situation, which further increases the (computational) complexity of the approach. While the reward definition works relatively well on the training and test maps of Section 6.1.1, the reward terms might require completely different weightings for other scenarios – which need to be determined on a case-by-case basis. Therefore, the generality of the PRIMAL reward is limited. Furthermore, the manual reward design can fundamentally change the original MAPF objective and lead to unintended side effects [88].

CACTUS and C^{3}LARA, on the other hand, use a very simple reward function that is directly aligned with the MAPF objective, according to [86] and Section 4. Table 3 shows the reward terms of PRIMAL and CACTUS (and C^{3}LARA).

Appendix B: Hyperparameters

All common hyperparameters used by all MARL approaches in the experiments, as reported in Section 6.2 in the paper, are listed in Table 4. The final values were chosen based on a coarse grid search which finds a tradeoff between performance and computation w.r.t. *CACTUS (QMIX)*. We directly adopted the final values in Table 4 for all other approaches and domains from Sections 6 and 6.2.

Table 3 Reward terms of PRIMAL, CACTUS, and C^{3}LARA

	PRIMAL [19]	CACTUS/C^{3}LARA
Completion reward	+20	+1
Time penalty	−0.3	−1
Wait penalty	−0.5	✗
Collision penalty	−2	✗
Blocking penalty	−2	✗

Table 4 Common hyperparameters and their respective final values used by all algorithms evaluated in the paper. We also list the numbers that have been tried during development of the paper

H.Param.	Final Value	Numbers/Range	Description
E	32	{16, 32, 128, 256}	Number of episodes per epoch or batch.
X	5000	{1000, 2000, 5000, 10000}	Number of epochs. E was gradually increased to assess the stability of the learning progress until convergence.
α	0.001	{0.001}	Learning rate. We used the default value of ADAM in `torch` without further tuning.
Clip norm	1	{1,∞}	Gradient clipping parameter. Using a clip norm of 1 leads to better performance than disabling it with ∞.
λ	1	{0, 1}	Trace parameter for TD(λ) learning.
$\lvert\tau_{t,i}\rvert$	1	{1, 5, 10}	Local history length. It was set to 1 to reduce computation because the other values did not significantly improve performance.

Appendix C: Neural network architectures

We coarsely tuned the neural network architectures from Section 6.1.3 in the paper w.r.t. performance and computation for $\hat{\pi}_i$ and $\hat{Q}_i$. The number of hidden layers was varied between 1, 2, and 3, where 2 is the minimum number of layers that worked reliably well. The number of units is inspired by prior work [39, 57] and set to 64 without any tuning. Using ELU or ReLU activation did not make any significant difference for any neural network thus we stick to ELU throughout the experiments.

Acknowledgements The research of T.P., J.D., and J.R. was supported by the National Science Foundation (NSF) under grant number 2112533. The research of S.K. was supported by NSF under grant numbers 2544613, 2434916, and 2321786, as well as gifts from Amazon Robotics and the Donald Bren Foundation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government. We thank Han Zhang for the initial discussion on conflict-based curricula.

Author Contributions T.P. designed and implemented the concepts of CACTUS. T.P. and J.D. designed and implemented the concepts of C3LARA. T.P., J.D., J.R., and S.K. discussed the concepts. T.P. and S.K. provided and discussed related work. T.P. and S.K. designed and conducted the experiments. T.P., J.D., J.R., and S.K. discussed the results and visualized the data. All authors contributed to writing the manuscript.

Funding The research of T.P., J.D., and J.R. was supported by the National Science Foundation (NSF) under grant number 2112533. The research of S.K. was supported by NSF under grant numbers 2544613, 2434916, and 2321786, as well as gifts from Amazon Robotics and the Donald Bren Foundation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

Data Availability No datasets were generated or analysed during the current study.

Materials Availability Not applicable.

Code availability Our code is available at https://github.com/thomyphan/rl4mapf. The centralized search-b ased MAPF solvers CBSH and MAPF-LNS are from the public repositories of [11] and [15], respectively.

Declarations

Competing Interests The authors declare no competing interests.

Ethics Approval and Consent to Participate Not applicable.

References

1. Li, J., Tinka, A., Kiesel, S., Durham, J. W., Kumar, T. S., & Koenig, S. (2021). Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *AAAI Conference on Artificial Intelligence* (vol. 35, pp. 11272–11281).
2. Zhang, Y., Fontaine, M. C., Bhatt, V., Nikolaidis, S., & Li, J. (2023). Multi-Robot Coordination and Layout Design for Automated Warehousing. In *International Joint Conference on Artificial Intelligence* (pp. 5503–5511).
3. Phan, T., Gabor, T., Sedlmeier, A., Ritz, F., Kempter, B., Klein, C., Sauer, H., Schmid, R., Wieghardt, J., Zeller, M., & Linnhoff-Popien, C. (2020). Learning and Testing Resilience in Cooperative Multi-Agent Systems. In *19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '20 (pp. 1055–1063). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC.
4. Zhang, Y., Fontaine, M. C., Bhatt, V., Nikolaidis, S., & Li, J. (2023). Arbitrarily Scalable Environment Generators via Neural Cellular Automata. In *Advances in Neural Information Processing Systems* (pp. 57212–57225).
5. Jennings, J. S., Whelan, G., & Evans, W. F. (1997). Cooperative Search and Rescue with a Team of Mobile Robots. In *1997 8th International Conference on Advanced Robotics. Proceedings. ICAR'97* (pp. 193–200).
6. Ho, F., Geraldes, R., Gonçalves, A., Rigault, B., Sportich, B., Kubo, D., Cavazza, M., & Prendinger, H. (2022). Decentralized multi-agent path finding for uav traffic management. *IEEE Transactions on Intelligent Transportation Systems, 23*(2), 997–1008.
7. Li, J., Hoang, T. A., Lin, E., Vu, H. L., & Koenig, S. (2023). Intersection Coordination with Priority-Based Search for Autonomous Vehicles. *AAAI Conference on Artificial Intelligence, 37*(10), 11578–11585.
8. Šišlák, D., Volf, P., & Pěchouček, M. (2011). Agent-Based Cooperative Decentralized Airplane-Collision Avoidance. *IEEE Transactions on Intelligent Transportation Systems, 12*(1), 36–46.
9. Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., Barták, R., & Boyarski, E. (2019). Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *International Symposium on Combinatorial Search* (vol. 10, pp. 151–158)
10. Ratner, D., & Warmuth, M. (1986). Finding a Shortest Solution for the NxN Extension of the 15-Puzzle is Intractable. In *5th AAAI National Conference on Artificial Intelligence* (pp. 168–172).
11. Felner, A., Li, J., Boyarski, E., Ma, H., Cohen, L., Kumar, T. S., & Koenig, S. (2018). Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. In *International conference on automated planning and scheduling* (vol. 28, pp. 83–87)
12. Sharon, G., Stern, R., Felner, A., & Sturtevant, N. (2012). Conflict-Based Search For Optimal Multi-Agent Path Finding. *AAAI Conference on Artificial Intelligence, 26*(1), 563–569.
13. Cohen, L., & Koenig, S. (2016). Bounded Suboptimal Multi-Agent Path Finding using Highways. In *International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 3978–3979).
14. Li, J., Ruml, W., & Koenig, S. (2021). EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding. *AAAI Conference on Artificial Intelligence, 35*, 12353–12362.
15. Li, J., Chen, Z., Harabor, D., Stuckey, P. J., & Koenig, S. (2021). Anytime Multi-Agent Path Finding via Large Neighborhood Search. In *International joint conference on artificial intelligence* (pp. 4127–4135).

16. Okumura, K. (2023). Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding. In *International joint conference on artificial intelligence*.
17. Li, J., Harabor, D., Stuckey, P. J., et al. (2021). Pairwise Symmetry Reasoning for Multi-Agent Path Finding Search. *Artificial Intelligence,* 103574.
18. Phan, T., Ritz, F., Altmann, P., Zorn, M., Nüßlein, J., Kölle, M., Gabor, T., & Linnhoff-Popien, C. (2023). Attention-Based Recurrence for Multi-Agent Reinforcement Learning under Stochastic Partial Observability. In *40th International Conference on Machine Learning*.
19. Sartoretti, G., Kerr, J., Shi, Y., Wagner, G., Kumar, T. S., Koenig, S., & Choset, H. (2019). PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters, 4*(3), 2378–2385.
20. Phan, T., Phan, T., & Koenig, S. (2025). Generative Curricula for Multi-Agent Path Finding via Unsupervised and Reinforcement Learning. *Journal of Artificial Intelligence Research, 82*, 2471–2534.
21. Phan, T., & Koenig, S. (2026). Spatially Grouped Curriculum Learning for Multi-Agent Path Finding. *AAAI Conference on Artificial Intelligence (AAAI), 40*(35), 29642–29650.
22. Tanenbaum, A. S., & Van Steen, M. (2007). Distributed Systems: Principles and Paradigms. Prentice-Hall .
23. Phan, T., Sommer, F., Ritz, F., Altmann, P., Nüßlein, J., Kölle, M., Belzner, L., et al. (2024). Emergent Cooperation from Mutual Acknowledgment Exchange in Multi-Agent Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems*.
24. Yan, Z., Zheng, H., & Wu, C. (2024). Multi-Agent Path Finding for Cooperative Autonomous Driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 12361–12367).
25. Zheng, H., Yan, Z., & Wu, C. (2024). Multi-Agent Path Finding for Mixed Autonomy Traffic Coordination. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 958–965).
26. Phan, T., Belzner, L., Gabor, T., Sedlmeier, A., Ritz, F., & Linnhoff-Popien, C. (2021). Resilient Multi-Agent Reinforcement Learning with Adversarial Value Decomposition. *AAAI Conference on Artificial Intelligence,* (13).
27. Silver, D. (2005). Cooperative Pathfinding. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 1*(1), 117–122.
28. Okumura, K., Machida, M., Défago, X., & Tamura, Y. (2022). Priority Inheritance with Backtracking for Iterative Multi-Agent Path Finding. *Artificial Intelligence, 310*, 103752.
29. Skrynnik, A., Andreychuk, A., Yakovlev, K., & Panov, A. (2024). Decentralized Monte Carlo Tree Search for Partially Observable Multi-Agent Pathfinding. *AAAI Conference on Artificial Intelligence, 38*, 17531–17540.
30. Alkazzi, J.-M., & Okumura, K. (2024). A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding. *IEEE Access, 12*, 57390–57409.
31. Andreychuk, A., Yakovlev, K., Panov, A., & Skrynnik, A. (2025). MAPF-GPT: Imitation Learning for Multi-Agent Pathfinding at Scale. *Proceedings of the AAAI Conference on Artificial Intelligence, 39*(22), 23126–23134.
32. Jiang, H., Wang, Y., Veerapaneni, R., Duhan, T., Sartoretti, G., & Li, J. (2025). Deploying Ten Thousand Robots: Scalable Imitation Learning for Lifelong Multi-Agent Path Finding. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1–7).
33. Buşoniu, L., Babuška, R., & De Schutter, B. (2010). Multi-Agent Reinforcement Learning: An Overview. *Innovations in Multi-Agent Systems and Applications-1*, 183–221.
34. Tan, M. (1993). Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *10th International Conference on Machine Learning* (pp. 330–337).
35. Veerapaneni, R., Jakobsson, A., Ren, K., Kim, S., Li, J., & Likhachev, M. (2025). Work Smarter Not Harder: Simple Imitation Learning with CS-PIBT Outperforms Large-Scale Imitation Learning for MAPF. In *2025 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 10229–10236).
36. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the Game of Go without Human Knowledge. *Nature, 550*(7676), 354–359.
37. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual Multi-Agent Policy Gradients. *AAAI Conference on Artificial Intelligence*, *32*(1).
38. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems* (pp. 6379–6390).
39. Rashid, T., Samvelyan, M., Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2018). QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *35th International Conference on Machine Learning* (pp. 4295–4304).
40. Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., & Georgiev, P. (2019). Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature*, 1–5.

41. Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. (2019). Human-Level Performance in 3D Multiplayer Games with Population-based Reinforcement Learning. *Science, 364*(6443).
42. Florensa, C., Held, D., Wulfmeier, M., Zhang, M., & Abbeel, P. (2017). Reverse Curriculum Generation for Reinforcement Learning. In *Conference on Robot Learning* (pp. 482–495). PMLR
43. Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., & Wu, Y. (2022). The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems*, *35*.
44. Damani, M., Luo, Z., Wenzel, E., & Sartoretti, G. (2021). PRIMAL $_2$: Pathfinding via Reinforcement and Imitation Multi-Agent Learning-Lifelong. *IEEE Robotics and Automation Letters, 6*(2), 2666–2673.
45. Wang, Y., Xiang, B., Huang, S., & Sartoretti, G. (2023). SCRIMP: Scalable Communication for Reinforcement-and Imitation-Learning-Based Multi-Agent Pathfinding. In *2023 International Conference on Autonomous Agents and Multiagent Systems* (pp. 2598–2600).
46. Sutton, R. (2019). The Bitter Lesson. *Incomplete Ideas (Blog), 13*(1), 38.
47. Brown, T., Mann, B., Ryder, N., Subbiah, M., et al. (2020). Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (vol. 33, pp. 1877–1901). Curran Associates, Inc .
48. Liu, A., Feng, B., Xue, B., Wang, B., et al. (2024). Deepseek-v3 Technical Report. arXiv preprint arXiv:2412.19437
49. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (2002). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.
50. Esser, P., Kulal, S., Blattmann, A., et al. (2024). Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. ICML'24. JMLR.
51. Phan, T., Driscoll, J., Romberg, J., & Koenig, S. (2024). Confidence-Based Curriculum Learning for Multi-Agent Path Finding. In *23rd International Conference on Autonomous Agents and Multiagent Systems* (pp. 1558–1566).
52. Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems* (pp. 1057–1063).
53. Su, J., Adams, S., & Beling, P. (2021). Value-Decomposition Multi-Agent Actor-Critics. AAAI *Conference on Artificial Intelligence*, *35*.
54. Watkins, C. J., & Dayan, P. (1992). Q-Learning. *Machine Learning, 8*(3–4), 279–292.
55. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-Level Control through Deep Reinforcement Learning. *Nature*, *518*(7540).
56. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347
57. Phan, T., Ritz, F., Belzner, L., Altmann, P., Gabor, T., & Linnhoff-Popien, C. (2021). VAST: Value Function Factorization with Variable Agent Sub-Teams. *Advances in Neural Information Processing Systems, 34*, 24018–24032.
58. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., & Graepel, T. (2018). Value-Decomposition Networks for Cooperative Multi-Agent Learning based on Team Reward. In *International Conference on Autonomous Agents and Multiagent Systems (Extended Abstract)*.
59. Son, K., Kim, D., Kang, W. J., Hostallero, D. E., Yi, Y. (2019). QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *36th International Conference on Machine Learning* (pp. 5887–5896).
60. Ha, D., Dai, A. M., & Le, Q. V. (2017). HyperNetworks. *International Conference on Learning Representations*.
61. Wang, J., Ren, Z., Liu, T., Yu, Y., & Zhang, C. (2020). QPLEX: Duplex Dueling Multi-Agent Q-Learning. *International Conference on Learning Representations*.
62. Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. In *26th International Conference on Machine Learning*.
63. Soviany, P., Ionescu, R. T., Rota, P., & Sebe, N. (2022). Curriculum Learning: A Survey. *International Journal of Computer Vision, 130*(6).
64. Narvekar, S., Sinapov, J., Leonetti, M., & Stone, P. (2016). Source Task Creation for Curriculum Learning. *International Conference on Autonomous Agents and Multiagent Systems*.
65. Narvekar, S., Peng, B., Leonetti, M., Sinapov, J., Taylor, M. E., & Stone, P. (2020). Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *The Journal of Machine Learning Research*, *21*(1).
66. Tesauro, G. (1995). Temporal Difference Learning and TD-Gammon. *Communications of the ACM, 38*(3), 58–68.

67. Silva, F. L. D., & Costa, A. H. R. (2018). Object-Oriented Curriculum Generation for Reinforcement Learning. In *17th International Conference on Autonomous Agents and Multiagent Systems* (pp. 1026–1034).
68. Gabor, T., Sedlmeier, A., Kiermeier, M., Phan, T., Henrich, M., Pichlmair, M., Kempter, B., Klein, C., Sauer, H., Schmid, R., & Wieghardt, J. (2019). Scenario Co-Evolution for Reinforcement Learning on a Grid World Smart Factory Domain. In *Genetic and Evolutionary Computation Conference* (pp. 898–906).
69. Dennis, M., Jaques, N., Vinitsky, E., Bayen, A., Russell, S., Critch, A., & Levine, S. (2020). Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design. *Advances in Neural Information Processing Systems*, *33*.
70. Asada, M., Noda, S., Tawaratsumida, S., & Hosoda, K. (1996). Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning. *Machine Learning, 23*, 279–303.
71. McAleer, S., Agostinelli, F., Shmakov, A., & Baldi, P. (2018). Solving the Rubik's Cube with Approximate Policy Iteration. *International Conference on Learning Representations*.
72. Agostinelli, F., McAleer, S., Shmakov, A., & Baldi, P. (2019). Solving the Rubik's Cube with Deep Reinforcement Learning and Search. *Nature Machine Intelligence, 1*(8), 356–363.
73. Uchendu, I., Xiao, T., Lu, Y., Zhu, B., Yan, M., Simon, J., Bennice, M., Fu, C., Ma, C., Jiao, J., Levine, S., & Hausman, K. (2023). Jump-Start Reinforcement Learning. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, & J. Scarlett (Eds.), *40th International Conference on Machine Learning. Proceedings of Machine Learning Research* (vol. 202, pp. 34556–34583). PMLR
74. Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., & Fergus, R. (2018). Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. *International Conference on Learning Representations*.
75. Du, Y., Abbeel, P., & Grover, A. (2021). It Takes Four to Tango: Multiagent Self Play for Automatic Curriculum Generation. *International Conference on Learning Representations*.
76. Wu, J., Yang, T., Hao, X., Hao, J., Zheng, Y., Wang, W., & Taylor, M. E. (2023). PORTAL: Automatic Curricula Generation for Multiagent Reinforcement Learning. In *International Conference on Autonomous Agents and Multiagent Systems (Extended Abstract)* (pp. 2460–2462).
77. Huang, T., Koenig, S., & Dilkina, B. (2021). Learning to Resolve Conflicts for Multi-Agent Path Finding with Conflict-Based Search. *AAAI Conference on Artificial Intelligence, 35*, 11246–11253.
78. Huang, T., Li, J., Koenig, S., & Dilkina, B. (2022). Anytime Multi-Agent Path Finding via Machine Learning-Guided Large Neighborhood Search. In *36th AAAI Conference on Artificial Intelligence (AAAI)* (pp. 9368–9376).
79. Kaduri, O., Boyarski, E., & Stern, R. (2020). Algorithm Selection for Optimal Multi-Agent Pathfinding. *International Conference on Automated Planning and Scheduling, 30*, 161–165.
80. Phan, T., Huang, T., Dilkina, B., & Koenig, S. (2024). Adaptive Anytime Multi-Agent Path Finding Using Bandit-Based Large Neighborhood Search. *AAAI Conference on Artificial Intelligence*.
81. Phan, T., Zhang, B., Chan, S.-H., & Koenig, S. (2025). Anytime Multi-Agent Path Finding with an Adaptive Delay-Based Heuristic. *AAAI Conference on Artificial Intelligence (AAAI), 39*, 23286–23294.
82. Phan, T., Chan, S.-H., & Koenig, S. (2026). Truncated Counterfactual Learning for Anytime Multi-Agent Path Finding. *AAAI Conference on Artificial Intelligence (AAAI), 40*(35), 29633–29641.
83. Skrynnik, A., Andreychuk, A., Nesterova, M., Yakovlev, K., & Panov, A. (2024). Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning. *AAAI Conference on Artificial Intelligence, 38*, 17541–17549.
84. Zhao, C., Zhuang, L., Huang, Y., & Liu, H. (2023). Curriculum Learning Based Multi-Agent Path Finding for Complex Environments. In *2023 International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE
85. Pham, P., & Bera, A. (2023). Crowd-Aware Multi-Agent Pathfinding with Boosted Curriculum Reinforcement Learning. arXiv preprint arXiv:2309.10275
86. Koenig, S., & Simmons, R. G. (1993). Complexity Analysis of Real-Time Reinforcement Learning. *AAAI Conference on Artificial Intelligence, 93*, 99–105.
87. Garcıa, J., & Fernández, F. (2015). A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research, 16*(1), 1437–1480.
88. Skalse, J., Howe, N., Krasheninnikov, D., & Krueger, D. (2022). Defining and Characterizing Reward Gaming. *Advances in Neural Information Processing Systems, 35*, 9460–9471.
89. Leike, J., Martic, M., Krakovna, V., Ortega, P. A., Everitt, T., Lefrancq, A., Orseau, L., & Legg, S. (2017). AI Safety Gridworlds. arXiv preprint arXiv:1711.09883
90. Strawn, K. J., Phan, T., Wang, E., Ayanian, N., Koenig, S., & Lindemann, L. (2025). Multi-Agent Path Finding Among Dynamic Uncontrollable Agents with Statistical Safety Guarantees. arXiv preprint arXiv:2507.22282

91. Xu, Z., Hasselt, H. P., & Silver, D. (2018). Meta-Gradient Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2396–2407.
92. Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020). Agent57: Outperforming the Atari Human Benchmark. In *Proceedings of the 37th International Conference on Machine Learning* (pp. 507–517).
93. Wang, Y., Han, B., Wang, T., Dong, H., & Zhang, C. (2021). DOP: Off-Policy Multi-Agent Decomposed Policy Gradients. *International Conference on Learning Representations*.
94. Zhang, T., Li, Y., Wang, C., Xie, G., & Lu, Z. (2021). FOP: Factorizing Optimal Joint Policy of Maximum-Entropy Multi-Agent Reinforcement Learning. In M. Meila, & T. Zhang (Eds.), *38th International Conference on Machine Learning. Proceedings of Machine Learning Research* (vol. 139, pp. 12491–12500). PMLR
95. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction. MIT Press.
96. Bellman, R. (1957). Dynamic programming. Princeton University Press.
97. Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2019). Diversity is All You Need: Learning Skills without a Reward Function. *International Conference on Learning Representations*.
98. Eysenbach, B., & Levine, S. (2022). Maximum Entropy RL (Provably) Solves Some Robust RL Problems. *International Conference on Learning Representations*.
99. Osband, I., Van Roy, B., Russo, D. J., & Wen, Z. (2019). Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research, 20*(124), 1–62.
100. Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2018). Parameter Space Noise for Exploration. *International Conference on Learning Representations*.
101. Schmidhuber, J. (1991). Curious Model-Building Control Systems. In *IEEE International Joint Conference on Neural Networks* (pp. 1458–1463). IEEE
102. Grinstead, C. M., & Snell, J. L. (2012). Introduction to Probability. *American Mathematical Society*.
103. Kemeny, J. G., & Snell, J. L. (1960). *Finite Markov Chains*. Van Nostrand Publishing Company.
104. Wilson, R. J. (2010). Introduction to Graph Theory. Longman.
105. Devlin, S., & Kudenko, D. (2011). Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems. In *The International Conference on Autonomous Agents and Multiagent Systems* (pp. 225–232). ACM
106. Boutilier, C. (1996). Planning, Learning and Coordination in Multiagent Decision Processes. In *6th Conference on Theoretical Aspects of Rationality and Knowledge* (pp. 195–210).
107. Oliehoek, F. A., Spaan, M. T. J., & Vlassis, N. (2008). Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research, 32*(1), 289–353.
108. Oliehoek, F. A., & Amato, C. (2016). A Concise Introduction to Decentralized POMDPs. Springer.
109. Hernandez-Leal, P., Kaisers, M., Baarslag, T., & De Cote, E. M. (2017). A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. arXiv Preprint arXiv:1707.09183
110. Laurent, G. J., Matignon, L., & Le Fort-Piat, N. (2011). The World of Independent Learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems, 15*(1), 55–64.
111. Skrynnik, A., Andreychuk, A., Borzilov, A., Chernyavskiy, A., Yakovlev, K., & Panov, A. (2025) POGEMA: A Benchmark Platform for Cooperative Multi-Agent Pathfinding. In *International Conference on Learning Representations (ICLR)*.
112. Luna, R., & Bekris, K. E. (2011). Push and Swap: Fast Cooperative Path-Finding with Completeness Guarantees. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. IJCAI'11* (pp. 294–300). AAAI Press.
113. Hoenig, W., Kumar, T. K., Cohen, L., Ma, H., Xu, H., Ayanian, N., & Koenig, S. (2016). Multi-Agent Path Finding with Kinematic Constraints. *Proceedings of the International Conference on Automated Planning and Scheduling, 26*(1), 477–485.

Authors and Affiliations

Thomy Phan[1,2,3] · Joseph Driscoll[4] · Justin Romberg[4] · Sven Koenig[2,3,5]

✉ Thomy Phan
thomy.phan@uni-bayreuth.de

Joseph Driscoll
jdriscoll7@gatech.edu

Justin Romberg
jrom@ece.gatech.edu

Sven Koenig
sven.koenig@uci.edu

[1] Department of Computer Science, University of Bayreuth, Bayreuth 95447, Germany

[2] Thomas Lord Department of Computer Science, University of Southern California, Los Angeles, CA 90007, USA

[3] Department of Computer Science, University of California, Irvine, Irvine, CA 92697, USA

[4] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

[5] School of Science and Technology, Örebro University, Örebro SE-70182, Sweden