Confidence-Based Curriculum Learning for Multi-Agent Path Finding

Thomy Phan
University of Southern California
Los Angeles, USA
thomy.phan@usc.edu

Joseph Driscoll
Georgia Institute of Technology
Atlanta, USA
jdriscoll7@gatech.edu

Justin Romberg
Georgia Institute of Technology
Atlanta, USA
jrom@ece.gatech.edu

Sven Koenig
University of Southern California
Los Angeles, USA
skoenig@usc.edu

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 depend on global information, which limits their scalability and flexibility regarding changes or new maps that would require expensive replanning. *Multi-agent reinforcement learning (MARL)* offers an alternative way by learning decentralized policies that can generalize over 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 general approaches are needed to bring the areas of MARL and MAPF closer together with significantly lower costs. In this paper, we propose *Confidence-based Auto-Curriculum for Team Update Stability (CACTUS)* as a lightweight MARL approach to 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. We evaluate CACTUS in 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.

KEYWORDS

Multi-Agent Path Finding; Multi-Agent Reinforcement Learning; Curriculum Learning

ACM Reference Format:
Thomy Phan, Joseph Driscoll, Justin Romberg, and Sven Koenig. 2024. Confidence-Based Curriculum Learning for Multi-Agent Path Finding. In

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024, IFAAMAS, 9 pages.

1 INTRODUCTION

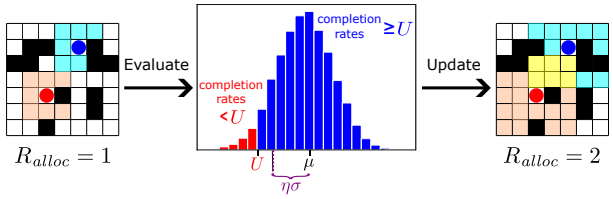

Figure 1: Curriculum update scheme of CACTUS. The agents (colored circles) are trained and evaluated w.r.t. a goal allocation radius R_{alloc} (shaded squares around the agents). When the average completion rate μ exceeds the decision threshold U with a certain confidence level such that $\mu - \eta\sigma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

A wide range of real-world applications like goods transportation in warehouses, search and rescue missions, and traffic management 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 w.r.t. flowtime or makespan is NP-hard [29, 39]. Despite the problem complexity, there exist a variety of MAPF solvers that find optimal [32], bounded suboptimal [5], or quick feasible solutions [16]. Most MAPF solvers are centralized and require global information, which limits scalability and flexibility regarding changes that would need expensive replanning. This also limits applicability to partially observable real-time domains [30].

Multi-agent reinforcement learning (MARL) offers an alternative way by learning decentralized policies that can generalize over a variety of maps and make decisions under partial observability [4, 44]. 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 policies that can be executed independently under partially observability afterwards [18, 28].

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 [5, 14, 17, 46]. 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 reactive and decentralized therefore alleviating the computational and communication requirements of centralized MAPF solvers [28].
(2) **Generalization**: The learned policies can generalize over a variety of maps thus do not require complete retraining or replanning when being used on new maps [30].
(3) **Robustness**: The learned policies make decisions based on actual observations therefore being able to react to local changes, i.e., emerging obstacles or new paths, without requiring replanning of the whole system [24].

On the other hand, MAPF poses an exciting challenge for MARL due to its practical relevance and the following aspects [32, 34]:

(1) **Sparse Rewards**: MAPF represents a complex navigation problem, where all agents are only rewarded for reaching their goals. Naive MARL would need exhaustive exploration to obtain informative data, which is time-consuming [10].
(2) **Dynamic Constraints**: Agents are not allowed to collide therefore having temporal constraints in addition to static constraints imposed by obstacles and boundaries [34].
(3) **Coordination**: MAPF requires coordination of spatially close agents with potentially emergent effects like congestion or circulation. So far, most MARL methods only focus on coordination on a small scale though [18, 51].

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, extensively shaped rewards, and centralized MAPF solvers for imitation learning [6, 30, 48]. We argue that much simpler and general approaches are needed to bring the areas of MARL and MAPF closer together with significantly lower costs.

In this paper, we propose *Confidence-based Auto-Curriculum for Team Update Stability (CACTUS)* as a lightweight MARL approach to 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 start and goal locations to enhance state-of-the-art MARL techniques that would likely fail to learn any meaningful policy otherwise.
- We evaluate CACTUS in 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.

2 BACKGROUND

2.1 Multi-Agent Path Finding

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. 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* $v_{start,i} \in \mathcal{V}$ and a *goal location* $v_{goal,i} \in \mathcal{V}$. We assume that $v_{start,i} \neq v_{start,j}$ and $v_{goal,i} \neq v_{goal,j}$ for any agent pair $i, j \in \mathcal{D}$ with $i \neq j$.

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 $\langle p_{i,t}, p_{i,t+1} \rangle = \langle p_{i,t+1}, p_{i,t} \rangle \in \mathcal{E}$, $p_{i,0} = v_{start,i}$, $p_{i,l(p_i)} = v_{goal,i}$, and $l(p_i)$ is the *length* or *travel distance* of path p_i.

We consider *vertex conlicts* $\langle a_i, a_j, v, t \rangle$ 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 $\langle u, v \rangle = \langle v, u \rangle \in \mathcal{E}$ in opposite directions at time step t [39]. A plan P is a *solution*, i.e., *feasible*, when it does not have any vertex or edge conflict therefore no collisions. Our goal is to find a solution P^* that minimizes the *flowtime* $\sum_{p \in P} l(p)$.

Despite MAPF being an NP-hard problem, there exist a variety of MAPF solvers that find optimal [32], bounded suboptimal [5], or quick feasible solutions [16]. Most MAPF solvers are centralized and require global information which limits scalability and flexibility regarding changes or new maps that would need expensive replanning and redistribution of plans.

2.2 Multi-Agent Reinforcement Learning

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 $s_t \in \mathcal{S}$:

$$Q_{tot}^\pi(s_t, a_t) = \sum_{i \in \mathcal{D}} Q_i^\pi(s_t, a_t) \tag{1}$$

2.2.1 ***Policy Gradient MARL.*** To learn optimal policies π_i^* in large state spaces, function approximators $\hat{\pi}_{i,\theta}$ with parameters 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 [43]:

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

where $A_i^{\hat{\pi}}(s_t, a_t) = Q_i^{\hat{\pi}}(s_t, a_t) - V_i^{\hat{\pi}}(s_t)$ is the *advantage* of agent i and $V_i^{\hat{\pi}}(s_t) = \mathbb{E}_\pi[R_{t,i}|s_t]$ is its state value function. *Actor-critic*

approaches often approximate $\hat{A}_i \approx A_i^{\hat{\pi}_i}$ by replacing $Q_i^{\hat{\pi}}(s_t, a_t)$ with $R_{t,i}$ and $V_i^{\hat{\pi}}$ with $\mathbb{E}_{\pi_i}[Q_i^{\hat{\pi}}]$. $Q_i^{\hat{\pi}}$ can be approximated with a *critic* $\hat{Q}_{i,\omega}$ and parameters ω using value-based RL [20, 49].

Alternatively, $\hat{\pi}_{i,\theta}$ can be trained via *proximal policy optimization (PPO)* by iteratively minimizing the following surrogate loss [31]:

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

where $\phi_{t,i}(\theta) = \frac{\hat{\pi}_{i,\theta}(a_{t,i}|\tau_{t,i})}{\hat{\pi}_{i,\theta}^{old}(a_{t,i}|\tau_{t,i})}$ 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 [28]. 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 critic in Eq. 2 or 3 [18, 51]. While the value functions $\hat{Q}_i$ are only required during training, the learned policies $\hat{\pi}_i$ only condition on local histories $\tau_{t,i}$ thus being independently executable. 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 [30].

$\hat{Q}_i$ can be approximated separately for each agent i while integrating global information, which is done in actor-critic MARL algorithms like MAPPO or MADDPG [18, 51]. However, this approach lacks a *multi-agent credit assignment mechanism* for agent teams, where all agents optimize the same objective Q_{tot} (Eq. 1).

Alternatively, a common value function $\hat{Q}(\tau_t, a_t) \approx Q_{tot}(s_t, a_t)$ can be learned, which is factorized into $\langle \hat{Q}_1, ..., \hat{Q}_N \rangle$ as *local utility functions* by using a *factorization operator* Ψ [26, 27]:

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

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 [28, 42]. A factorization operator Ψ is *decentralizable* when satisfying the *IGM (Individual-Global-Max)* such that [37]:

$$argmax_{\mathbf{a_t}} \hat{Q}(\tau_t, a_t) = \begin{pmatrix} argmax_{a_{t,1}} \hat{Q}_1(\tau_{t,1}, a_{t,1}) \\ \vdots \\ argmax_{a_{t,N}} \hat{Q}_N(\tau_{t,N}, a_{t,N}) \end{pmatrix} \quad (5)$$

There exists a variety of factorization operators Ψ, which satisfy Eq. 5 using monotonicity constraints like QMIX [28] or nonlinear transformation like QPLEX or QTRAN [37, 47].

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 [3, 38]. The difficulty can depend on various aspects like the complexity of data samples, the objective function, or the learned model [10, 22]. Curriculum learning has been applied to *reinforcement learning (RL)* to solve hard exploration problems with sparse rewards or dynamic constraints [21]. The methods are typically based on self-play [35, 45], task graphs with traversal mechanisms [33], or automatic generation of tasks [7, 11].

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 [10, 11, 33].

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 [2, 10].

3 RELATED WORK

Reverse Curriculum Generation. Many works on RL-based motion control assume a single goal state, which is easy to specify [1, 19]. 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 [2, 10]. 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 by considering the *uncertainty* of performance estimates.

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 foundations of multi-agent learning [7, 8, 41], there exist methods particularly designed for MARL based on self-play, agent skills, and population-based training [14, 46, 50]. 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.

MARL for MAPF. MAPF and MARL are very active research areas with remarkable progress in recent years [14, 16, 46]. 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 *PRIMAL*. PRIMAL and its successor approaches are heavily engineered and very complex, using extremely large neural networks, 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 [6, 30, 48]. Despite their effectiveness, these approaches are very expensive to use due to significant effort on fine-tuning and 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 [23, 52]. Besides applying MARL to MAPF, there have been other attempts to combine MAPF with machine learning techniques to guide or select centralized search algorithms [12, 13, 15, 25]. Our goal is to provide a suitable foundation to bring the areas of MARL and MAPF closer together with *significantly lower costs*. Therefore, we propose a *simple reverse curriculum scheme* to ease applicability and enable faster progress in this direction.

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 work [6, 30, 39, 48], we focus on discrete gridworlds but try to keep our formulation general. An adaptation of our methods to arbitrary graphs, e.g., using graph neural networks, is left for future work.

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. 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 vertex or edge conflicts, i.e., collisions, are automatically treated as wait action. The individual reward $r_{t,i}$ is defined by +1 if agent i reaches its goal $v_{goal,i}$, 0 when agent i is staying at its goal location $v_{goal,i}$, 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 [30]. 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 $v_{goal,i}$ and indicates the direction to it.

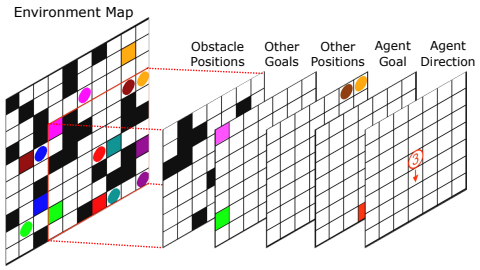

Figure 2: Example for an individual observation of the red agent in a gridworld domain. Agents are represented as colored circles, their goals as similarly-colored squares, and obstacles as black squares. Each agent i has a limited *field of view (FOV)* of the environment map, which is centered around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

When the discount factor is $\gamma = 1$, the negated return $-R_{t,i}$ of each agent i is equivalent to its travel distance $l(p_i)$ from time step t, if $v_{goal,i}$ was reached, and horizon T otherwise. Therefore, maximizing $Q_{tot} = \sum_{i \in \mathcal{D}} Q_i = -\mathbb{E}_I[\sum_{p \in P} l(p)]$ in MARL (Section 2.2 and Eq. 1) would be equivalent to minimizing the expected flowtime in MAPF w.r.t. any instance I on map G (Section 2.1).

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 includes collision attempts. Unlike prior work, we do not need additional penalties for particular situations like collisions, blocking, or waiting which could fundamentally change the actual objective and lead to unintended side-effects [36]

Thus, our problem formulation is simpler and more general, which allows us to solve it in a black-box manner that is more intuitive for standard MARL methods [28, 47, 51]. However, the simplicity of our formulation notably increases difficulty since the reward is sparse in contrast to PRIMAL and related approaches.

5 CONFIDENCE-BASED CURRICULUM

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 Eq. 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 QMIX or QPLEX as illustrated in Fig. 3 [28, 47, 51]. Since the value factorization based actor-critic scheme has been used in a variety of prior work [24, 40], we do not claim novelty here, but propose it as a basic approach to train cooperative policies via credit assignment mechanisms [28, 37, 47].

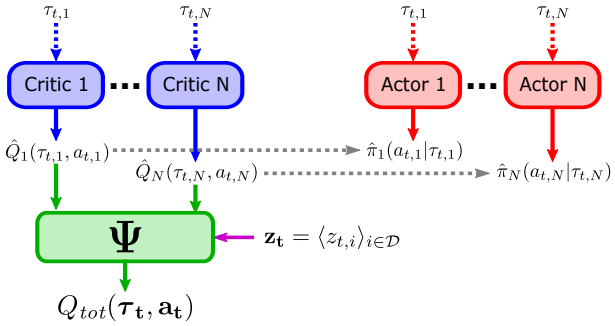

Figure 3: Common actor-critic scheme as used in various prior work on cooperative MARL [24, 40]. A separate critic is trained for each actor using some centralized factorization operator Ψ like QMIX or QPLEX [27].

To address the coordination problem as mentioned in the introduction, we suggest to optimize individual utilities $\hat{Q}_i \approx Q_i^{\hat{\pi}}$ under consideration of the utilitarian metric Q_{tot} in Eq. 1. For that, the individual utilities $\hat{Q}_i$ can be learned end-to-end through a factorization operator Ψ like QMIX or QPLEX to consider multi-agent credit assignment from a cooperative perspective [26–28, 47]. The factorization operator Ψ approximates the expected sum of individual returns by minimizing the *factorization loss* $\mathcal{L}^{\Psi}$ defined by:

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

The local policies $\hat{\pi}_i$ are then trained according to Eq. 2 or 3 using *counterfactual advantages* $\hat{A}_i$ defined by [40]:

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

where the return $R_{t,i}$ represents the negative travel distance $-l(p_i)$ of agent i from time step t according to Section 4. 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. Eq. 1 and 6.

5.2 Reverse Curriculum Scheme

The training scheme described above represents a general approach to learn coordinated policies $\hat{\pi}_i$ [24, 40]. However, sparse rewards and dynamic constraints in our MAPF setting (Section 4) pose particular challenges that require a suitable curriculum to learn meaningful policies [2, 10]. Unlike prior work that relied on complex 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 costs.

At the beginning of every episode m, each agent i starts at a random location $v_{start,i} \in \mathcal{V}$ and needs to navigate to an assigned goal location $v_{goal,i} \in \mathcal{V}$ which is randomly placed within an *allocation radius* R_{alloc}[1] around the start location $v_{start,i}$. R_{alloc} characterizes the *potential difficulty* of generated instances I as larger allocation radii may require the agents to move and explore over longer distances to locate their respective goals. Thus, our reverse curriculum scheme starts with a small allocation radius of $R_{alloc} = 1$ and gradually increments R_{alloc} with improving performance, which is measured by the *completion rate* $\rho_m^{complete} = \frac{N_m^{goal}}{N}$, where $N_m^{goal} = |\{i \in \mathcal{D} | v_{t,i} = v_{goal,i}\}|$ is the number of agents that successfully reached their respective goals in 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* $\mu = \frac{1}{E} \sum_{m=1}^{E} \rho_m^{complete}$ and its *standard deviation* $\sigma = \sqrt{\frac{1}{E-1} \sum_{m=1}^{E} (\rho_m^{complete} - \mu)^2}$.

Assuming that the completion rates $\rho_m^{complete}$ 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%. 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, $DIST : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a vertex distance function, U is the curriculum decision threshold, and η is the deviation factor.

Algorithm 1 Confidence-Based Curriculum Learning for MAPF

procedure *DRIVE*($\mathcal{G}, DIST, U, \eta$)
  Initialize parameters of $\hat{\pi}_i, \hat{Q}_i$ for each agent $i \in \mathcal{D}$ and Ψ
  Set $R_{alloc} = 1$
  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 $\mathcal{V}_{goal,i} \leftarrow \{v \in \mathcal{V} | DIST(v, v_{start,i}) \leq R_{alloc}\}$
        Randomly select goal location $v_{goal,i}$ from $\mathcal{V}_{goal,i}$
        ▷ Goal locations must be unique, i.e, $v_{goal,i} \neq v_{goal,j}$ if $i \neq j$
      for time step $t \leftarrow 0, T-1$ **do**
        for agent $i \in \mathcal{D}$ **do**
          $a_{t,i} \sim \pi_i(\cdot | \tau_{t,i})$
        $a_t \leftarrow \langle a_{t,1}, ..., a_{t,N} \rangle$
        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
        $\tau_{t+1} \leftarrow \langle \tau_t, a_t, z_{t+1} \rangle$
      $\rho_m^{complete} \leftarrow |\{i \in \mathcal{D} | v_{t,i} = v_{goal,i}\}| / N$
    Train Ψ and $\hat{\pi}_i, \hat{Q}_i$ for each agent $i \in \mathcal{D}$ with all e_t
    Calculate μ and σ with all $\rho_m^{complete}$
    if $\mu - \eta\sigma \geq U$ **then** ▷ Curriculum update decision
      $R_{alloc} \leftarrow R_{alloc} + 1$
  return $\langle \hat{\pi}_1, ..., \hat{\pi}_N \rangle$

5.3 Conceptual Discussion

CACTUS represents a simple reverse curriculum scheme inspired by prior work [2, 10]. 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 a variety of MAPF instances and maps, our scheme adjusts the random allocation of goals around the random start locations.

In contrast to prior work [23, 52], 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 respective goals, which are allocated in close proximity within R_{alloc}. With increasing R_{alloc}, the allocation areas of different agents may overlap, automatically causing agents to interact with each other thus increasing coordination pressure. R_{alloc} is only incremented when agents are able to coordinate and reach their goals with sufficiently high confidence, which is checked with the hyperparameters U and η. Thus, CACTUS offers an adaptive approach to solving MAPF problems via MARL without requiring explicit separation of agent skills [23, 46, 52], extensive engineering of rewards, or expensive acquisition of expert data [6, 30, 48].

Since the goals are randomly initialized within 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.

[1] A vertex distance measure is required, which can depend on the number of edges, for example. In this paper, we measure the distance between two positions (x_1, y_1) and (x_2, y_2) by $max\{|x_1 - x_2|, |y_1 - y_2|\}$ for two-dimensional environments.

6 EXPERIMENTAL SETUP

6.1 Maps and Instances

The *training maps* are randomly generated according to [30] 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_{alloc}. If $R_{alloc} = \infty$, then the goals can be placed anywhere on the training map.

The *test maps* are provided by [30]. 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 to ensure a fair comparison between different MARL approaches.

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

6.2 Algorithms and Training

We implement PPO[2] for policy learning according to Eq. 3 and QMIX, QPLEX, and MAPPO for critic learning. We implement a purely RL-based version of PRIMAL using the same reward function as defined in [30]. In addition, we employ a naive baseline, called *No Curriculum*, only consisting of PPO and QMIX[3] without any shaped reward. PRIMAL and *No Curriculum* are trained with $R_{alloc} = \infty$ (Section 6.1).

We train all algorithms on different training maps as explained in Section 6.1 with $N = 8$ agents for 5000 epochs consisting of $E = 32$ episodes. The maps of size $K = 10$ are sampled twice as often as the other map sizes as proposed in [30]. 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 network $\hat{\pi}_i$ and $\hat{Q}_i$ respectively.

We denote *CACTUS (X)* as CACTUS using MARL algorithm X for critic learning. Unless stated otherwise, CACTUS always uses PPO for policy and *X=QMIX* for critic learning.

In addition, we run *CBSH* as a slow but optimal MAPF solver with a runtime limit of 5 minutes [9] and *MAPF-LNS* [16] as a fast anytime MAPF solver with a runtime limit of 1 minute.

6.3 Neural Networks and Hyperparameters

For CACTUS 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 perceptron (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 [30].

[2]Code available at github.com/thomyphan/rl4mapf.

[3]We choose QMIX for consistency with our default setting. Replacing QMIX with QPLEX or MAPPO does not notably affect the performance of this baseline.

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.

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 [6, 30, 48].

7 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 trained policies with the pre-generated test instances I explained in Section 6.1.

7.1 Simplicity of CACTUS

To demonstrate the simplicity of CACTUS, we first quantify the training time, the training 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 [30].

An overview is given in Table 1. In almost all aspects, CACTUS only requires 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 MAPF solver, which saves a significant amount of compute. While the reward function of PRIMAL requires four penalties for very specific situations, CACTUS is trained with a very simple reward function that penalizes any time step unless the goal is reached without considering any specific case (Section 4).

Table 1: Comparison of the original PRIMAL and CACTUS w.r.t. various numbers in our experiments. The last column provides the amount of effort relative to the original PRIMAL. The numbers of the original PRIMAL are from [30]. Unlike [30], we do not use any GPU or expert data for training.

	PRIMAL (original [30])	CACTUS	Rel. to PRIMAL
Training Time	≈ 20 days	≈ 1 day	≈ 5%
# Training Episodes	≈ 3.8 million	160,000	≈ 4.2%
# Parameters	≈ 13 million	579,979	≈ 4.5%
# Reward Penalties	4	1	25%

Fig. 4 compares the number of trainable parameters and neural network architectures of PRIMAL and CACTUS. In CACTUS, the critic with the mixing network 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 thus does not depend on specialized hardware or significant computational effort for fine-tuning and training.

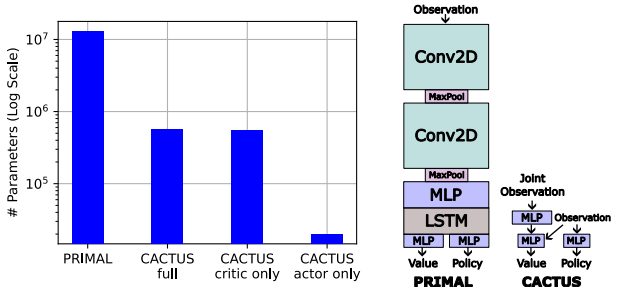

Figure 4: *Left*: Comparison of the number of trainable parameters in PRIMAL and CACTUS. Note the logarithmic scale on the y-axis. *Right*: The schematic network architectures used for PRIMAL and CACTUS. The sizes do not reflect any quantity and only illustrate the components used for learning.

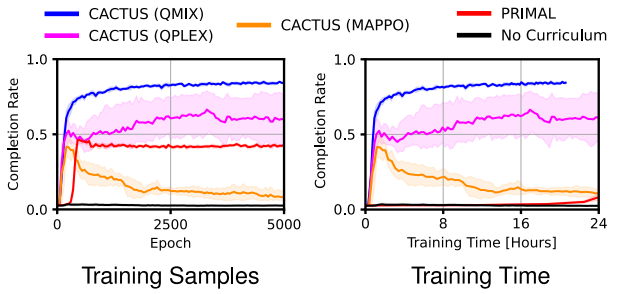

Figure 5: Average training progress of CACTUS variants, PRIMAL, and a naive MARL baseline without any curriculum w.r.t. training epochs (left) and training time (right). The performance is evaluated on all pre-generated test instances I of [30] with $K \in \{10, 40, 80\}$, $\delta \in \{0, 0.1, 0.2, 0.3\}$, and $N = 8$ agents. Shaded areas show the 95% confidence interval.

7.2 Curriculum Learning

We evaluate the effect of CACTUS using QMIX, QPLEX, and MAPPO as current state-of-the-art MARL techniques [28, 47, 51]. 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. 5. *CACTUS (QMIX)* and *CACTUS (QPLEX)* perform best. PRIMAL always outperforms *CACTUS (MAPPO)*. *No Curriculum* fails to learn any meaningful policy. However, PRIMAL is barely able to outperform *No Curriculum* after 24 hours of training. *CACTUS (QMIX)* completes training below 24 hours, while *CACTUS (QPLEX)* and *CACTUS (MAPPO)* require slightly more training time than 24 hours.

7.3 CACTUS Hyperparameters

Next, we evaluate the impact of different 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 similar as shown in Fig. 5.

The results are shown in Fig. 6. 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.

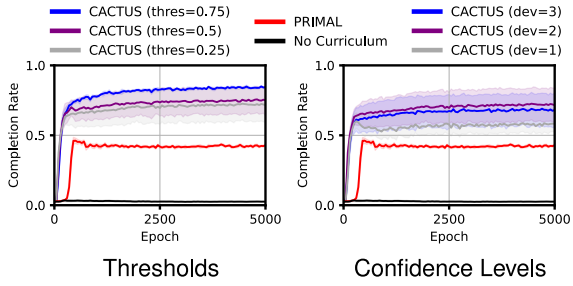

Figure 6: Average training progress of CACTUS variants, PRIMAL, and a naive MARL baseline without any curriculum w.r.t. different decision thresholds U (left) and deviation factors η (right). The performance is evaluated on all pre-generated test instances I of [30] with $K \in \{10, 40, 80\}$, $\delta \in \{0, 0.1, 0.2, 0.3\}$, and $N = 8$ agents. The right plot shows CACTUS variants with $U = 0.25$. Shaded areas show the 95% confidence interval.

7.4 Generalization

Finally, we evaluate the generalization capabilities of policies trained with CACTUS using QMIX, QPLEX, and MAPPO as well as PRIMAL, and *No Curriculum*. All policies are trained for 5000 epochs consisting of 32 episodes before being evaluated on all test instances I with map size $K \in \{40, 80\}$ and obstacle density $\delta \in \{0, 0.2\}$ with different numbers of agents N. Note that all policies have only been trained with $N = 8$ (Section 6.2). We also report the average performance of the centralized MAPF solvers *CBSH* and *MAPF-LNS*.

The generalization results w.r.t. different numbers of agents N are shown in Fig. 7. *CACTUS (QMIX)* generalizes best compared to all other MARL approaches. In test instances with low obstacle density, *CACTUS (QMIX)* always achieves an average completion rate over 70% when scaling up to $N = 64$ agents. If $\delta = 0$, then *CACTUS (QMIX)* is able to complete at least 50% of all agents when scaling up to $N = 128$. *CACTUS (QPLEX)* always outperforms PRIMAL on average except in instances with low obstacle density and map size $K = 40$, where the number of agents exceeds 32. PRIMAL is always outperformed by *CACTUS (QMIX)* but consistently outperforms *CACTUS (MAPPO)* and *No Curriculum*. All approaches perform poorly when the agent number is $N \geq 256$ or $\delta = 0.2$.

Compared to the centralized MAPF solvers, *CACTUS (QMIX)* can only outperform *CBSH*, when the obstacle density δ is low. *MAPF-LNS* is the best performing approach, always achieving a completion rate of 100% in all test instances.

7.5 Limitation in Structured Maps

In addition, we tested CACTUS and the learning baselines in structured maps with rooms and narrow corridors such as mazes. While

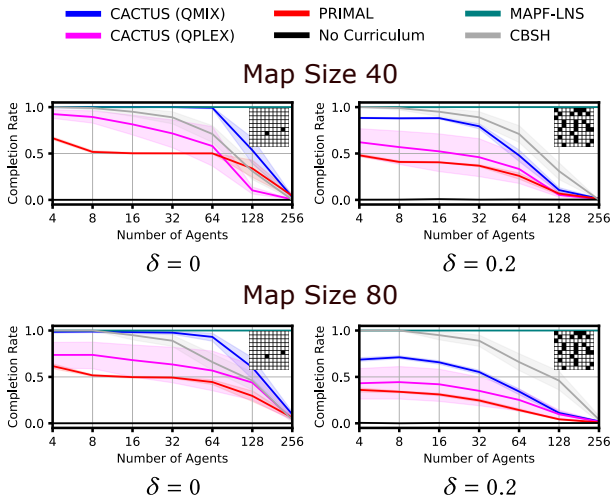

Figure 7: Average generalization performance of CACTUS variants as well as MARL and MAPF baselines to all test instances I of size $K \in \{40, 80\}$ and obstacles density $\delta \in \{0, 0.1, 0.2, 0.3\}$ w.r.t. different numbers of agents N. The icons on the top-right of each plot show a 10×10 sub-grid example to illustrate the obstacle density. Shaded areas show the 95% confidence interval.

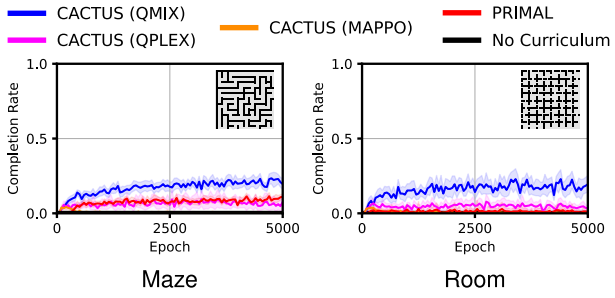

Figure 8: Average training progress of CACTUS variants, PRIMAL, and a naive MARL baseline without any curriculum in structured maps. The performance is evaluated on all maze and room maps provided by [39] respectively.

training was conducted on randomly generated maps as explained in Section 6.1, we evaluated the training progress using all maze and room maps of the common MAPF benchmark from [39].

The results are shown in Fig. 8. Compared to the results for unstructured maps in Section 7.2, all approaches perform significantly worse with none of them reaching an average completion rate above 25%. *CACTUS (QMIX)* consistently outperforms all other approaches, which generally fail to complete more than 10% of all agent tasks.

8 DISCUSSION

In this paper, we presented CACTUS as a lightweight MARL approach to 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.

Our results confirm the necessity of adequate curricula, as standard MARL methods without any curriculum would likely fail to learn any meaningful policy in our MAPF setting (Section 4). This is due to the sparse reward and the dynamic constraints of the problem formulation. The shaped reward function of PRIMAL is helpful in improving performance over standard MARL. Due to the many specifically defined penalties, PRIMAL policies are rather conservative, which leads to a performance plateau that cannot be overcome without imitation learning on suitable expert data. However, CACTUS with QMIX or QPLEX clearly outperforms PRIMAL with 95% less training time and learnable parameters without any additional reward shaping or expert data. CACTUS with MAPPO performs poorly, which indicates the importance of adequate credit assignment mechanisms, e.g, value factorization, to address the coordination problem in MAPF.

CACTUS is robust w.r.t. the choice of hyperparameters as any configuration with decision threshold $U \geq 25\%$ and deviation factor $\eta \geq 1$ outperforms PRIMAL as shown in Fig. 6. However, the decision threshold U should be sufficiently high (at least 50%) to ensure an adequate difficulty level for the agents. 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.

Despite the relatively restrictive time and data budget compared to the original PRIMAL, CACTUS generalizes quite well with value factorization 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 any meaningful policy in the MAPF problem. While generalizing better than the alternative approaches, CACTUS still has some limitations regarding high obstacle density, large map sizes and structured maps with separate rooms and narrow corridors, where there is still potential for improvement. Despite the ability of efficient decentralized decision-making, CACTUS is not competitive to centralized MAPF solvers due to its limited scalability.

Nevertheless, CACTUS clearly demonstrates how simple and well-defined curricula can enhance MARL techniques for MAPF without relying on extremely large neural networks, extensively shaped reward functions, or centralized MAPF solvers for imitation learning. Therefore, we hope to provide a suitable foundation to enable faster progress in connecting the areas of MARL and MAPF with significantly lower costs.

ACKNOWLEDGMENTS

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1817189, 1837779, 1935712, 2121028, 2112533, and 2321786 as well as a gift from Amazon Robotics. The research at the Georgia Institute of Technology was supported by NSF under grant number 2112533. 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.

REFERENCES

[1] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. 2019. Solving the Rubik's Cube with Deep Reinforcement Learning and Search. *Nature Machine Intelligence* 1, 8 (2019), 356–363.

[2] Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida, and Koh Hosoda. 1996. Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning. *Machine learning* 23 (1996), 279–303.

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *26th International Conference on Machine Learning*.

[4] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. Multi-Agent Reinforcement Learning: An Overview. *Innovations in Multi-Agent Systems and Applications-1* (2010), 183–221.

[5] Liron Cohen and Sven Koenig. 2016. Bounded Suboptimal Multi-Agent Path Finding Using Highways. In *IJCAI*. 3978–3979.

[6] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. 2021. PRIMAL _2: Pathfinding via Reinforcement and Imitation Multi-Agent Learning-Lifelong. *IEEE Robotics and Automation Letters* 6, 2 (2021), 2666–2673.

[7] Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. 2020. Emergent Complexity and Zero-Shot Transfer via Unsupervised Environment Design. *NeurIPS* 33 (2020).

[8] Yuqing Du, Pieter Abbeel, and Aditya Grover. 2021. It Takes Four to Tango: Multiagent Self Play for Automatic Curriculum Generation. In *ICLR*.

[9] Ariel Felner, Jiaoyang Li, Eli Boyarski, Hang Ma, Liron Cohen, TK Satish Kumar, and Sven Koenig. 2018. Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding. In *ICAPS*, Vol. 28. 83–87.

[10] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse Curriculum Generation for Reinforcement Learning. In *Conference on Robot Learning*. PMLR, 482–495.

[11] Thomas Gabor, Andreas Sedlmeier, Marie Kiermeier, Thomy Phan, et al. 2019. Scenario Co-Evolution for Reinforcement Learning on a Grid World Smart Factory Domain. In *Genetic and Evolutionary Computation Conference*. 898–906.

[12] Taoan Huang, Sven Koenig, and Bistra Dilkina. 2021. Learning to Resolve Conflicts for Multi-Agent Path Finding with Conflict-Based Search. In *AAAI Conference on Artificial Intelligence*, Vol. 35. 11246–11253.

[13] Taoan Huang, Jiaoyang Li, Sven Koenig, and Bistra Dilkina. 2022. Anytime Multi-Agent Path Finding via Machine Learning-Guided Large Neighborhood Search. In *36th AAAI Conference on Artificial Intelligence (AAAI)*. 9368–9376.

[14] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Lever, et al. 2019. Human-Level Performance in 3D Multiplayer Games with Population-based Reinforcement Learning. *Science* 364, 6443 (2019).

[15] Omri Kaduri, Eli Boyarski, and Roni Stern. 2020. Algorithm Selection for Optimal Multi-Agent Pathfinding. In *ICAPS*, Vol. 30. 161–165.

[16] Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J. Stuckey, and Sven Koenig. 2021. Anytime Multi-Agent Path Finding via Large Neighborhood Search. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 4127–4135.

[17] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W Durham, TK Satish Kumar, and Sven Koenig. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *AAAI Conference on Artificial Intelligence*, Vol. 35. 11272–11281.

[18] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[19] Stephen McAleer, Forest Agostinelli, Alexander Shmakov, and Pierre Baldi. 2018. Solving the Rubik's Cube with Approximate Policy Iteration. In *ICLR*.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Rusu, et al. 2015. Human-Level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (2015).

[21] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *The Journal of Machine Learning Research* 21, 1 (2020).

[22] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. 2016. Source Task Creation for Curriculum Learning. In *AAMAS*.

[23] Phu Pham and Aniket Bera. 2023. Crowd-Aware Multi-Agent Pathfinding with Boosted Curriculum Reinforcement Learning. *arXiv preprint arXiv:2309.10275* (2023).

[24] Thomy Phan, Lenz Belzner, Thomas Gabor, Andreas Sedlmeier, Fabian Ritz, and Claudia Linnhoff-Popien. 2021. Resilient Multi-Agent Reinforcement Learning with Adversarial Value Decomposition. *AAAI Conference on Artificial Intelligence* 13 (2021).

[25] Thomy Phan, Taoan Huang, Bistra Dilkina, and Sven Koenig. 2024. Adaptive Anytime Multi-Agent Path Finding Using Bandit-Based Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)* (2024). https://thomyphan.github.io/publication/2024-02-01-aaai-phan

[26] Thomy Phan, Fabian Ritz, Philipp Altmann, Maximilian Zorn, Jonas Nüßlein, Michael Kölle, Thomas Gabor, and Claudia Linnhoff-Popien. 2023. Attention-Based Recurrence for Multi-Agent Reinforcement Learning under Stochastic Partial Observability. In *40th International Conference on Machine Learning*.

[27] Thomy Phan, Fabian Ritz, Lenz Belzner, Philipp Altmann, Thomas Gabor, and Claudia Linnhoff-Popien. 2021. VAST: Value Function Factorization with Variable Agent Sub-Teams. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 24018–24032.

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *35th International Conference on Machine Learning*. PMLR, 4295–4304.

[29] Daniel Ratner and Manfred Warmuth. 1986. Finding a Shortest Solution for the NxN Extension of the 15-Puzzle is Intractable. In *5th AAAI National Conference on Artificial Intelligence*. AAAI Press, 168–172.

[30] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, et al. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters* 4, 3 (2019), 2378–2385.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[32] Guni Sharon, Roni Stern, Ariel Felner, and Nathan Sturtevant. 2012. Conflict-Based Search For Optimal Multi-Agent Path Finding. *AAAI Conference on Artificial Intelligence* 26, 1 (Sep. 2012), 563–569.

[33] Felipe Leno Da Silva and Anna Helena Reali Costa. 2018. Object-Oriented Curriculum Generation for Reinforcement Learning. In *17th International Conference on Autonomous Agents and Multiagent Systems*. 1026–1034.

[34] David Silver. 2005. Cooperative Pathfinding. *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 1, 1 (Sep. 2005), 117–122.

[35] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, et al. 2017. Mastering the Game of Go without Human Knowledge. *Nature* 550, 7676 (2017), 354–359.

[36] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. 2022. Defining and Characterizing Reward Gaming. *Advances in Neural Information Processing Systems* 35 (2022), 9460–9471.

[37] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *36th International Conference on Machine Learning*. PMLR, 5887–5896.

[38] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum Learning: A Survey. *International Journal of Computer Vision* 130, 6 (2022).

[39] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *International Symposium on Combinatorial Search*, Vol. 10. 151–158.

[40] Jianyu Su, Stephen Adams, and Peter Beling. 2021. Value-Decomposition Multi-Agent Actor-Critics. In *AAAI Conference on Artificial Intelligence*, Vol. 35.

[41] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2018. Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. In *ICLR*.

[42] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, et al. 2018. Value-Decomposition Networks for Cooperative Multi-Agent Learning based on Team Reward. In *AAMAS (Extended Abstract)*.

[43] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*. 1057–1063.

[44] Ming Tan. 1993. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *10th International Conference on Machine Learning*. 330–337.

[45] Gerald Tesauro et al. 1995. Temporal Difference Learning and TD-Gammon. *Commun. ACM* 38, 3 (1995), 58–68.

[46] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Mathieu, et al. 2019. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature* (2019), 1–5.

[47] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2020. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *ICLR*.

[48] Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. 2023. SCRIMP: Scalable Communication for Reinforcement-and Imitation-Learning-Based Multi-Agent Pathfinding. In *2023 International Conference on Autonomous Agents and Multiagent Systems*. 2598–2600.

[49] Christopher JCH Watkins and Peter Dayan. 1992. Q-Learning. *Machine Learning* 8, 3-4 (1992), 279–292.

[50] Jizhou Wu, Tianpei Yang, Xiaotian Hao, Jianye Hao, Yan Zheng, Weixun Wang, and Matthew E Taylor. 2023. PORTAL: Automatic Curricula Generation for Multiagent Reinforcement Learning. In *AAMAS (Extended Abstract)*. 2460–2462.

[51] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems* 35 (2022).

[52] Cheng Zhao, Liansheng Zhuang, Yihong Huang, and Haonan Liu. 2023. Curriculum Learning Based Multi-Agent Path Finding for Complex Environments. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.