

Generative Curricula for Multi-Agent Path Finding via Unsupervised and Reinforcement Learning

Thomy Phan

THOMY.PHAN@USC.EDU

*University of Southern California
Thomas Lord Department of Computer Science
Los Angeles, CA 90089, USA*

Timy Phan

TIMY.PHAN@LMU.DE

*LMU Munich
Computer Vision & Learning Group, School of Arts
Munich, 80799, Germany*

Sven Koenig

SVEN.KOENIG@UCI.EDU

*University of California, Irvine
Department of Computer Science
Irvine, CA 92697, USA*

Abstract

Multi-Agent Path Finding (MAPF) is the challenging problem of finding collision-free paths for multiple agents, which has a wide range of applications, such as automated warehouses, smart manufacturing, and traffic management. Recently, machine learning-based approaches have become popular in addressing MAPF problems in a decentralized and potentially generalizing way. Most *learning-based MAPF* approaches use reinforcement and imitation learning to train agent policies for decentralized execution under partial observability. However, current state-of-the-art approaches suffer from a prevalent bias to micro-aspects of particular MAPF problems, such as congestions in corridors and potential delays caused by single agents, leading to tight specializations through extensive engineering via oversized models, reward shaping, path finding algorithms, and communication. These specializations are generally detrimental to the sample efficiency, i.e., the learning progress given a certain amount of experience, and generalization to previously unseen scenarios. In contrast, *curriculum learning* offers an elegant and much simpler way of training agent policies in a step-by-step manner to master all aspects implicitly without extensive engineering. In this paper, we propose a generative curriculum approach to learning-based MAPF using *Variational Autoencoder Utilized Learning of Terrains (VAULT)*. We introduce a two-stage framework to **(I)** train the VAULT via unsupervised learning to obtain a latent space representation of maps and **(II)** use the VAULT to generate curricula in order to improve sample efficiency and generalization of learning-based MAPF methods. For the second stage, we propose a bi-level curriculum scheme by combining our VAULT curriculum with a low-level curriculum method to improve sample efficiency further. Our framework is designed in a modular and general way, where each proposed component serves its purpose in a black-box manner without considering specific micro-aspects of the underlying problem. We empirically evaluate our approach in maps of the public MAPF benchmark set as well as novel artificial maps generated with the VAULT. Our results demonstrate the effectiveness of the VAULT as a map generator and our VAULT curriculum in improving sample efficiency and generalization of learning-based MAPF methods compared to alternative approaches. We also demonstrate how data pruning can further

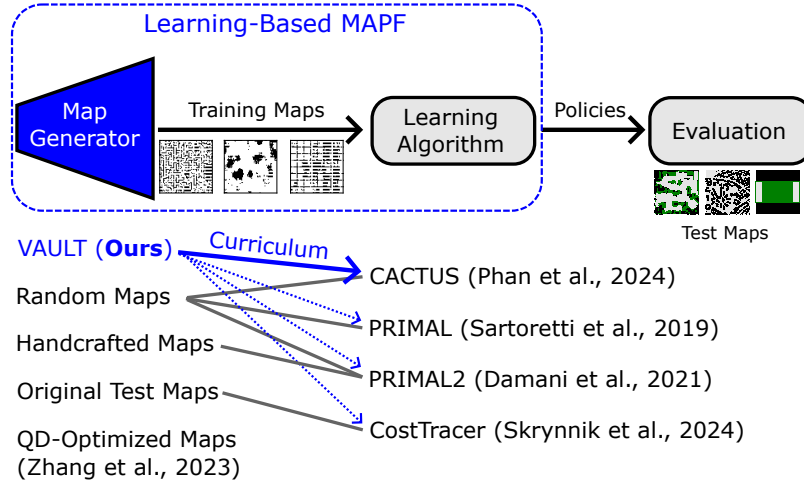


Figure 1: Overview of *learning-based MAPF*, which comprises map generation and learning algorithms to train decentralized agent policies, with some prior work listed here. Our main contributions are a *map generator*, namely the *VAULT*, and a *curriculum scheme* using the *VAULT* for *reinforcement learning (RL)* algorithms, e.g., CACTUS, PRIMAL, etc.

reduce the dependence on available maps without affecting the generalization potential of our approach.

1. Introduction

Multi-Agent Path Finding (MAPF) is the challenging problem of finding collision-free paths for multiple agents, which has a wide range of applications, such as automated warehouses (Li et al., 2021; Zhang et al., 2023b), smart manufacturing (Zhang et al., 2023a), and traffic management (Li et al., 2023). Finding optimal conflict-free paths is NP-hard, which limits the scalability of state-of-the-art MAPF solvers with optimality guarantees (Ratner & Warmuth, 1986; Yu & LaValle, 2013).

Traditional MAPF solvers are centralized and use heuristic search to plan paths and resolve potential collisions for all agents (Felner et al., 2018; Li et al., 2019b; Sharon et al., 2015). Despite recent advances in solving MAPF fast and suboptimally, which can efficiently scale up to thousands of agents (Li et al., 2022; Okumura, 2023, 2024), these solvers cannot generalize to novel maps or dynamic changes without replanning and may require manual adjustments of heuristics (Phan et al., 2024b; Sartoretti et al., 2019). The centralized design of the solvers poses a major obstacle to dynamic real-time applications, as the costs of sufficient communication bandwidth and node availability measures are prohibitive for large-scale systems, where all agents need to communicate with a central computing node, especially for replanning (Oliehoek & Amato, 2016; Tanenbaum & Van Steen, 2007).

Recently, machine learning-based approaches have become popular in addressing MAPF problems in a decentralized and potentially generalizing way, as sketched in Figure 1. Most *learning-based MAPF* approaches use reinforcement and imitation learning to train agent policies for decentralized execution under partial observability, e.g., due to noisy and limited sensors (Phan et al., 2023; Sartoretti et al., 2019). However, current state-of-the-art

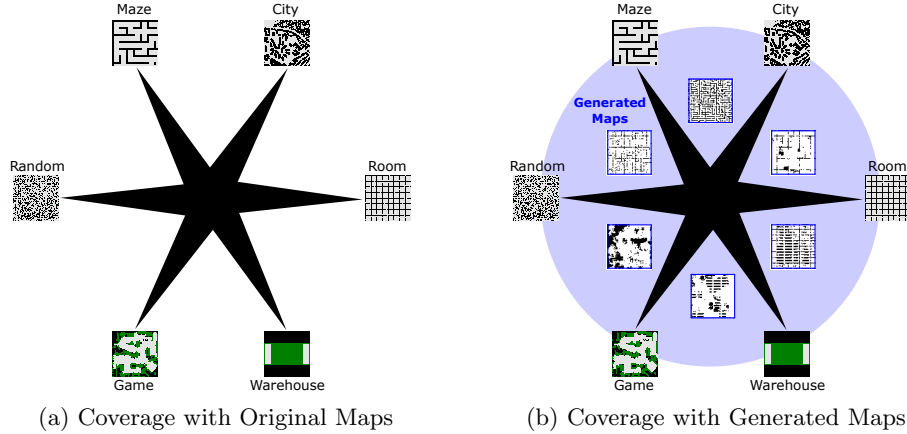


Figure 2: Illustration of the potential generalization coverage when training with the original test maps, e.g., (Stern et al., 2019) **(a)**, or generated maps **(b)**. The former can overfit on edge cases (black), while the latter can interpolate between those cases (blue), thus generalizing better. Note that this illustration only sketches the intuition behind our generative approach and does not necessarily reflect the actual map distribution or alignment.

approaches suffer from a prevalent bias to micro-aspects of particular MAPF problems, such as congestions in corridors and potential delays caused by single agents, leading to tight specializations through extensive engineering using oversized models (Sartoretti et al., 2019), reward shaping (Damani et al., 2021), path finding algorithms (Skrynnik et al., 2024a), and communication (Skrynnik et al., 2024b; Wang et al., 2023a). These specializations are generally detrimental to the *sample efficiency*, i.e., the learning progress given a certain amount of experience, and *generalization* to previously unseen scenarios.

In contrast, *curriculum learning* offers an elegant and much simpler way of training agent policies in a step-by-step manner to master all aspects implicitly without extensive engineering. This was demonstrated recently by a novel *auto-curriculum approach* that achieved notably higher sample efficiency while requiring less than 5% of the compute, data, and parameters than prior learning-based MAPF methods (Phan et al., 2024b). To promote fast and cost-efficient progress in this field, we advocate such directions to reduce tight specialization in micro-aspects and improve sample efficiency and generalization instead.

So far, most learning-based MAPF approaches have focused on unstructured maps with random obstacles (Sartoretti et al., 2019; Phan et al., 2024b; Wang et al., 2023a), hand-crafted maps (Damani et al., 2021), or the original test maps (Skrynnik et al., 2024b; Andreychuk et al., 2025), according to Figure 1. This induces bias and can lead to overfitting on edge cases, as illustrated in Figure 2a.

In this paper, we propose a generative curriculum approach to learning-based MAPF using *Variational Autoencoder Utilized Learning of Terrains (VAULT)*. To train and use the VAULT, we describe a modular and general framework where each proposed component serves its purpose in a black-box manner without considering specific micro-aspects of the underlying problem. Our contributions are as follows:

- We introduce a two-stage framework to **(I)** train the VAULT via unsupervised learning to obtain a smooth latent space representation of maps to interpolate between edge cases (Figure 2b) and **(II)** use the VAULT to generate curricula in order to improve sample efficiency and generalization of learning-based MAPF methods.
- For the second stage, we propose a bi-level curriculum scheme by combining our VAULT curriculum with the auto-curriculum method mentioned above to improve sample efficiency further (Phan et al., 2024a).
- We empirically evaluate our approach in maps of the public MAPF benchmark set as well as novel artificial maps generated with the VAULT. Our results demonstrate the effectiveness of the VAULT as a map generator and our VAULT curriculum in improving sample efficiency and generalization of learning-based MAPF methods compared to alternative approaches. We also demonstrate how data pruning can further reduce the dependence on available maps without affecting the generalization potential of our approach. Code is available at <https://github.com/thomyphan/gen-curricula-mapf>.

Scope of this Work We focus on *unsupervised learning* (to train the VAULT) and *reinforcement learning* (to train agent policies) for decentralized MAPF. Any additional mechanism, such as integrated imitation learning, path finding algorithms, or communication, is excluded to avoid distraction from our goal. However, we will discuss them in related work to distinguish our approach from prior methods and highlight their inability to fully leverage the generalization potential of machine learning due to extensive engineering.

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 the vertex set $\mathcal{V}$ contains all locations v and the edge set $\mathcal{E}$ contains all transitions or movements $\{u, v\}$ between adjacent locations $u, v \in \mathcal{V}$. An *instance* I consists of a map G and a set of *agents* $\mathcal{D} = \{1, \dots, N\}$ with each agent $i \in \mathcal{D}$ having a *start location* $v_{start,i} \in \mathcal{V}$ and a *goal location* $v_{goal,i} \in \mathcal{V}$. At every time step t , each agent i can move along the edges in $\mathcal{E}$ or wait at its current location $v_{t,i} \in \mathcal{V}$ (Stern et al., 2019). MAPF aims to find a collision-free plan for all agents. A *plan* $P = \{p_1, \dots, p_N\}$ consists of individual paths $p_i = \langle p_{i,1}, \dots, p_{i,l(p_i)} \rangle$ per agent i , where $\{p_{i,t}, p_{i,t+1}\} \in \mathcal{E}$, $p_{i,1} = v_{start,i}$, $p_{i,l(p_i)} = v_{goal,i}$, and $l(p_i)$ is the *length* or *travel time* of p_i .

In this paper, 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 (Stern et al., 2019). A plan P is a *solution*, i.e., *feasible*, when it does not have any vertex or edge conflicts, therefore being *collision-free*. The goal is to find a solution that minimizes the *flowtime* $\sum_{p \in P} l(p)$ or *makespan* $\max_{p \in P} l(p)$. The *completion rate* $\rho_t = \frac{|\{i \in \mathcal{D} | v_{t,i} = v_{goal,i}\}|}{N} \in [0, 1]$ is the fraction of agents at time step t which have reached their respective goals. An instance I is completely solved when $\rho_H = 1$ at some time step H .

2.2 MAPF as a Stochastic Game

To apply *reinforcement learning (RL)* to MAPF in a principled way, we need to formulate the MAPF problem, defined in Section 2.1, as a *stochastic game* $\mathcal{SG} = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \Omega \rangle$, where $\mathcal{D} = \{1, \dots, N\}$ is the set of agents, $\mathcal{S}$ is a set of states s_t , $\mathcal{A}(s_t) = \mathcal{A}_1(s_t) \times \dots \times \mathcal{A}_N(s_t)$ is the set of joint actions $a_t = \langle a_{t,1}, \dots, a_{t,N} \rangle$ executable at state $s_t \in \mathcal{S}$, $\mathcal{T}(s_{t+1}|s_t, a_t)$ is the transition probability, $\mathcal{R}(s_t, a_t) = \langle r_{t,1}, \dots, r_{t,N} \rangle \in \mathbb{R}^N$ is the joint reward with $r_{t,i}$ being the reward of agent $i \in \mathcal{D}$, $\mathcal{O}$ is a set of local observations $o_{t,i}$ for each agent i , and $\Omega(s_{t+1}) = o_{t+1} = \langle o_{t+1,1}, \dots, o_{t+1,N} \rangle \in \mathcal{O}^N$ is the subsequent joint observation (Emery-Montemerlo et al., 2004; Hansen et al., 2004).

Regarding MAPF problems, 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 $s_t = \langle v_{t,1}, \dots, v_{t,N} \rangle \in \mathcal{S} \subset \mathcal{V}^N$, 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 $\mathcal{A}_i(s_t)$ of each agent i is defined by the degree of its current location $v_{t,i} \in \mathcal{V}$ plus a wait action. The state transitions w.r.t. $\mathcal{T}$ are deterministic, where a valid move action will change the current location $v_{t,i}$ of the corresponding agent i to an adjacent location $v_{t+1,i}$ with $\{v_{t,i}, v_{t+1,i}\} \in \mathcal{E}$. Any attempt to move over a non-existent edge or cause a collision, i.e., a vertex or edge conflict, is automatically treated as a wait action. The individual reward $r_{t,i}$ is defined by +1 if agent i reaches its goal $v_{goal,i}$, -1 if agent i is not at its goal location $v_{t,i} \neq v_{goal,i}$, and zero if it is staying at its goal location $v_{goal,i}$ (Phan et al., 2024b). Each agent i can partially observe the state s_t through $o_{t,i}$, i.e., a local neighborhood around its location $v_{t,i}$, modeling noisy and limited sensors for decentralized decision-making (Oliehoek & Amato, 2016; Phan et al., 2023).

Due to the partial observability, each agent i needs to maintain an action-observation *history* $\tau_{t,i} = \langle o_{0,i}, a_{0,i}, \dots, a_{t-1,i}, o_{t,i} \rangle$ (Oliehoek & Amato, 2016). $\pi = \langle \pi_1, \dots, \pi_N \rangle$ 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 . In the following, we refer to the joint policy as *agent policies* to express the decentralization of all agent decisions. Each 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 $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}(s_t)$, where $R_{t,i} = \sum_{b=0}^{H-1} \gamma^b r_{t+b,i}$ is the *return* of agent i , $H > 0$ is the *horizon*, and $\gamma \in [0, 1]$ is the *discount factor*.

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 $v_{goal,i}$ was reached, and horizon H otherwise. Therefore, to minimize the MAPF flowtime, according to Section 2.1 we need to find *optimal agent policies* $\pi^* = \langle \pi_1^*, \dots, \pi_N^* \rangle$, which maximize the expected sum of individual returns Q_{tot} for all instances I :

$$\pi^* = \operatorname{argmax}_\pi \mathbb{E}_{\pi, I}[Q_{tot}(s_0, a_0)] = \operatorname{argmax}_\pi \mathbb{E}_{\pi, I}[\sum_{i \in \mathcal{D}} Q_i^\pi(s_0, a_0)] = \operatorname{argmin}_\pi \mathbb{E}_{\pi, I}[\sum_{i \in \mathcal{D}} l(p_i)] \quad (1)$$

In addition, we need to ensure the maximization of the completion rate ρ_t to incentivize feasible solutions for ideally all instances I and avoid local optima, e.g., via exploration.

2.3 Multi-Agent Reinforcement Learning (Multi-Agent RL)

2.3.1 POLICY-BASED REINFORCEMENT LEARNING

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 $J = \mathbb{E}_{\hat{\pi},I}[R_{0,i}]$. *Policy gradient methods* use gradients of the following form (Sutton et al., 2000):

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

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 $V_i^{\hat{\pi}}(s_t) = \mathbb{E}_{\hat{\pi},I}[R_{t,i} | s_t]$ 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 $\mathbb{E}_{\hat{\pi}_i,I}[Q_i^{\hat{\pi}}]$. $Q_i^{\hat{\pi}}$ can be approximated with a *critic* $\hat{Q}_{i,\omega}$ and parameters ω using value-based RL (Watkins & Dayan, 1992; Mnih et al., 2015).

Alternatively, $\hat{\pi}_{i,\theta}$ can be trained via *proximal policy optimization (PPO)* by minimizing the following surrogate loss (Schulman et al., 2017):

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

where $\xi_{t,i}(\theta) = \frac{\hat{\pi}_{i,\theta}(a_{t,i} | \tau_{t,i})}{\hat{\pi}_{i,\theta}^{old}(a_{t,i} | \tau_{t,i})}$ is the policy update 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.

The effectiveness of the policies strongly depends on the data quality and, therefore, on the *exploration* capabilities of the employed RL algorithm (Osband et al., 2019; Schmidhuber, 1991; Schulman et al., 2017). *Sparse reward* or *hard exploration* domains, such as MAPF, where agents only receive positive rewards when reaching their goals (Section 2.2), pose a significant challenge to policy-gradient RL algorithms, which can converge to poor policies due to the lack of informative feedback for the advantage calculation (Eysenbach et al., 2019; Eysenbach & Levine, 2022; Plappert et al., 2018).

In multi-agent scenarios, *independent (policy gradient) RL* is not guaranteed to converge in general due to the *non-stationarity* caused by simultaneously learning agents (Hernandez-Leal et al., 2017; Laurent et al., 2011).

2.3.2 CENTRALIZED TRAINING DECENTRALIZED EXECUTION (CTDE)

For many problems, RL training takes place in a laboratory or in a simulated environment where global information is available (Foerster et al., 2018; Rashid et al., 2020). Therefore, state-of-the-art multi-agent RL algorithms approximate *centralized value functions* $\hat{Q}_i$, which condition on global states s_t and joint actions a_t , and use them as critics in Equation 2 or 3 (Lowe et al., 2017; Foerster et al., 2018; Yu et al., 2022). 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 conventional MAPF plans P , these policies can *generalize* over a variety of scenarios and thus ideally do not need any retraining or replanning for novel maps G' or instances I' (Sartoretti et al., 2019).

$\hat{Q}_i$ can be approximated separately for each agent i using global information, which is often done in actor-critic algorithms like MAPPO or MADDPG (Lowe et al., 2017; Yu et al., 2022). However, this approach lacks a *multi-agent credit assignment mechanism* for agent teams, where all agents optimize a common objective, such as Q_{tot} in Equation 1.

Therefore, a common value function $\hat{Q}(\tau_t, a_t) \approx Q_{tot}(s_t, a_t)$ can be learned, which is factorized into $\langle \hat{Q}_1, \dots, \hat{Q}_N \rangle$ as *local utility functions* by using a *factorization operator* Ψ (Rashid et al., 2020; Phan et al., 2021, 2023):

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

In practice, Ψ is realized with deep neural networks, such that $\langle \hat{Q}_1, \dots, \hat{Q}_N \rangle$ can be learned end-to-end by minimizing the mean squared *temporal difference (TD)* loss (Rashid et al., 2020; Sunehag et al., 2018). A factorization operator Ψ is *decentralizable* when satisfying the *IGM (Individual-Global-Max)* such that (Son et al., 2019):

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

Satisfying the IGM ensures coordinated behavior at any time step t given an adequate value function approximation $\hat{Q}_i$ per agent i (whose quality depends on the exploration). Alternatively, $\hat{Q}_i$ can serve as *individual critics* to learn coordinated policies $\hat{\pi}_i$ via actor-critic RL (Peng et al., 2021; Phan et al., 2021; Su et al., 2021).

There exists a variety of factorization operators Ψ , which satisfy Equation 5 using monotonicity constraints like QMIX (Rashid et al., 2020) or nonlinear transformation like QPLEX or QTRAN (Son et al., 2019; Wang et al., 2020).

2.4 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 (Bengio et al., 2009; Soviany et al., 2022). The difficulty can depend on various aspects like the complexity of data samples, the objective function, or the machine learning model itself (Narvekar et al., 2016; Florensa et al., 2017).

Curriculum learning has been applied to RL to solve hard exploration problems with sparse rewards and constraints (Narvekar et al., 2020). The methods commonly use self-play (Tesauro et al., 1995), task graphs with traversal mechanisms (Silva & Costa, 2018), or automatic task generation (Dennis et al., 2020; Gabor et al., 2019; Wang et al., 2019).

A key challenge of curriculum learning is to determine a suitable sequence of tasks that are neither too easy nor too difficult for the learner to ensure steady and robust progress (Florensa et al., 2017; Phan et al., 2024b; Silva & Costa, 2018; Wang et al., 2019).

In this paper, we focus on *automatic task generation* via artificial maps G' and instances I' for learning-based MAPF, which is controlled by a flexibly usable generative model and a black-box optimization algorithm.

2.5 Autoencoder (AE)

2.5.1 PRELIMINARIES

An *autoencoder (AE)* is a neural network f trained to approximate the identity function w.r.t. a given dataset $\mathcal{X} = \{x_1, \dots, x_K\}$ with K unlabeled samples of dimension $d \geq 1$

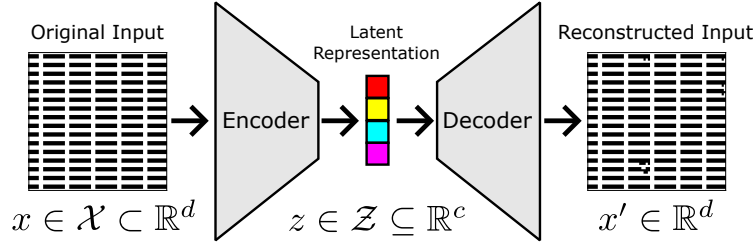


Figure 3: Schematic illustration of an autoencoder. The original input $x \in \mathcal{X} \subset \mathbb{R}^d$, e.g., an image of a finite dataset $\mathcal{X}$, is passed through the encoder to produce a *latent representation* or *vector* $z \in \mathcal{Z} \subseteq \mathbb{R}^c$ with $c < d$. The latent representation z is then passed through the decoder to generate a reconstruction $x' \in \mathbb{R}^d$ of the original input x . Note that x' is not necessarily included in the original dataset $\mathcal{X}$.

such that $f(x_k) \approx x_k$ (Bank et al., 2023; Kramer, 1992; Rumelhart et al., 1986). As illustrated in Figure 3, an AE consists of a learnable *encoder* $enc : \mathbb{R}^d \rightarrow \mathbb{R}^c$ and *decoder* $dec : \mathbb{R}^c \rightarrow \mathbb{R}^d$ with $d > c$ ¹. The encoder compresses data samples $x \in \mathcal{X} \subset \mathbb{R}^d$ into a *latent representation* or *vector* $z \in \mathcal{Z} \subseteq \mathbb{R}^c$ with $\mathcal{Z}$ denoting the *latent space* of the AE. The decoder $dec : \mathbb{R}^c \rightarrow \mathbb{R}^d$, attempts to reconstruct x via z by generating an artificial sample $x' \in \mathbb{R}^d$. Note that x' is not necessarily included in the original dataset $\mathcal{X}$. Both encoder and decoder are neural networks with potentially multiple layers to learn arbitrary abstract representations of x . AEs are trained via *unsupervised learning* by minimizing the following loss function using gradient descent (Baldi, 2012):

$$\mathcal{L}_{enc,dec}^{AE} = \mathbb{E}_{x \in \mathcal{X}}[\text{diff}(x, f(x))] \quad (6)$$

where $f(x) = dec(enc(x))$ represents the AE, and *diff* is the *reconstruction loss* function, e.g., the mean squared error for numeric data or the cross-entropy loss for categorical data.

AEs are commonly used for compression of complex data and non-linear clustering (Bank et al., 2023). In fact, AEs can be viewed as a generalization of the *principal component analysis (PCA)*, being able to learn non-linear manifolds (Japkowicz et al., 2000; Kramer, 1991; Plaut, 2018).

2.5.2 VARIATIONAL AUTOENCODER (VAE)

Variational autoencoders (VAE) are an extension of AEs based on *variational Bayes inference* (Kingma, 2013). Instead of mapping a sample x to a latent vector z , the VAE encoder enc maps x to a conditional latent distribution $\mathbb{P}(z|x)$. In practice, $\mathbb{P}(z|x)$ is approximated with a multivariate Gaussian distribution $q_\phi(z|x) = \mathcal{N}(\mu(x), \sigma^2(x))$, where $\mu(x)$ is the centroid and $\sigma^2(x)$ is the covariance of the latent distribution associated with x (Baldi, 2012; Kingma, 2013). A latent vector $z \sim q_\phi(\cdot|x)$ is sampled from the latent distribution and passed through the decoder dec for reconstruction. Through the distributional representation $q_\phi(z|x)$, VAEs can learn a smooth latent space representation. This allows them to

1. If $c \geq d$, the autoencoder only needs to set the weights of d units per layer to one, while setting the weights of the remaining $c - d$ units to zero. Thus, the *bottleneck* constraint $c < d$ enforces the learning of non-trivial latent representations (Bank et al., 2023; Kramer, 1992; Rumelhart et al., 1986).

interpolate between known data samples and morph them together, which is useful for, e.g., image generation (Kingma, 2013). Thus, the decoder in Figure 3 represents a *generative model*, which can create artificial samples x' from any latent vector sample $z' \in \mathbb{R}^c$ (Dai & Wipf, 2019; Van Den Oord et al., 2017).

In addition to the reconstruction loss $\mathcal{L}_{enc,dec}^{AE}$ of Equation 6, a VAE minimizes the *Kullback–Leibler (KL) divergence* D_{KL} between $q_\phi(z|x)$ and the normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ as a regularizing term (Kingma, 2013):

$$\mathcal{L}_{enc,dec}^{VAE} = \mathcal{L}_{enc,dec}^{AE} + \lambda \mathbb{E}_{x \in \mathcal{X}} [D_{KL}(q_\phi(z|x), \mathcal{N}(\mathbf{0}, \mathbf{I}))] \quad (7)$$

where $\lambda \geq 0$ regulates the KL divergence term. The higher λ , the more randomness is injected into the VAE, whereas smaller λ values lead to more deterministic VAE behavior and less smooth representations. The VAE reduces to a standard AE, when $\lambda = 0$.

Besides data generation, VAEs are also used for efficient processing in complex neural network architectures, such as modern diffusion models (Esser et al., 2021, 2024; Rombach et al., 2022) and planning-based decision-making (Asai et al., 2022; Ha & Schmidhuber, 2018; Hafner et al., 2019, 2020, 2021, 2023).

3. Related Work

3.1 Map Generation for Reinforcement Learning

Map generation is often used to create curricula for RL using coevolution or regret-based learning (Dennis et al., 2020; Jiang et al., 2021; Parker-Holder et al., 2022; Samvelyan et al., 2023; Wang et al., 2019). Most methods operate directly on the vertex space $\mathcal{V}$, e.g., grid cells, and only scale to small map sizes, such as 15×15 grids. All learning-based generators are tightly coupled to the learned agent policies due to being co-trained along with them (Bolland et al., 2022; Dennis et al., 2020; Samvelyan et al., 2023), which complicates reuse for other purposes, such as new RL training tasks or unbiased testing of other agent policies.

Our work uses *unsupervised learning* to train VAEs for map generation in a dedicated stage without bias toward any agent policy. Therefore, we can flexibly use our map generator for different training and testing purposes. Instead of optimizing over the vertex space $\mathcal{V}$, we search the *low-dimensional latent space* of the VAE via optimization algorithms to generate larger maps, such as 64×64 grids, for learning-based MAPF (Section 4.1.1).

3.2 Map Optimization for MAPF

Map optimization for MAPF is an emerging field where layouts are modified to aid traditional MAPF solvers in maximizing their completion rate (Gao & Prorok, 2023b, 2023a; Zhang et al., 2023b; Qian et al., 2024). Most approaches operate on the vertex space $\mathcal{V}$ and enforce domain-specific constraints, e.g., the obstacle density and the alignment of obstacle and goal locations (Gao & Prorok, 2023b; Zhang et al., 2023b).

(Gao & Prorok, 2023b, 2023a) use RL and constraint optimization to generate maps for MAPF, which only scale to small maps with a handful of obstacles. (Zhang et al., 2023b; Qian et al., 2024) uses *quality diversity (QD)* optimization based on *MAP-Elites* (Mouret & Clune, 2015). The method optimizes a diverse population of maps under domain-specific constraints using evolutionary operators and a surrogate fitness function for efficient

evaluation to improve the completion rate of traditional MAPF solvers. *Map connectivity* and *betweenness centrality* have been used to regulate the difficulty of QD-optimized maps w.r.t. their empirical hardness for traditional MAPF solvers (Ewing et al., 2022; Ren et al., 2024; Qian et al., 2024). (Zhang et al., 2023a) proposes a map generator based on *neural cellular automata (NCA)* and CMA-ES optimization (Fontaine & Nikolaidis, 2023; Fontaine et al., 2020) to replicate known structures for arbitrary map sizes.

Our work focuses on map generation for *learning-based MAPF*, as depicted in Figure 1. Besides improving the completion rate of RL algorithms, we want to improve their *sample efficiency* and *generalization capabilities*. Therefore, we do not enforce domain-specific constraints to *avoid bias*. The notion of difficulty for RL algorithms depends on the *amount of exploration* needed to progress (Ecoffet et al., 2019), in contrast to traditional MAPF solvers, which seem more sensitive to map connectivity and betweenness centrality (Ewing et al., 2022; Ren et al., 2024).

3.3 Machine Learning for Traditional MAPF Solvers

Machine learning is increasingly integrated into traditional MAPF solvers to guide search algorithms or select appropriate MAPF solvers for certain instances (Huang et al., 2021, 2022; Kaduri et al., 2020; Ren et al., 2021; Yan & Wu, 2024). Most techniques are offline learning approaches based on supervised or imitation learning, where scores or recommendation labels are provided by an oracle, e.g., a traditional MAPF solver or a human expert (Huang et al., 2022; Kaduri et al., 2020; Zhang et al., 2022). Multi-armed bandit algorithms have been used recently for online learning in anytime MAPF to further improve the scalability and effectiveness (Phan et al., 2024, 2025). A broad overview of the integration of machine learning techniques in MAPF is provided in (Alkazzi & Okumura, 2024). Despite the increasing popularity of machine learning techniques, the enhanced MAPF solvers remain centralized, thus being limited regarding generalization, dynamic changes, and prohibitive cost regarding communication and availability measures of central computing nodes (Oliehoek & Amato, 2016; Sartoretti et al., 2019; Tanenbaum & Van Steen, 2007).

Our work focuses on machine learning techniques that enable *decentralized and generalizing MAPF*, i.e., agent policies that can be executed separately under partial observability, ideally in previously unseen environments. In particular, we use *unsupervised learning* to train map generators and *reinforcement learning* to train agent policies by using the map generators to *create curricula*. According to our scope defined in the Introduction and Figure 1, we do not regard combinations with traditional centralized MAPF solvers yet.

3.4 Learning-Based MAPF for Decentralized Planning

3.4.1 REINFORCEMENT LEARNING-BASED MAPF

Many learning-based MAPF approaches use RL to train agent policies for decentralized planning under partial observability. Since MAPF is a hard exploration problem, imitation learning is often employed by using a traditional MAPF solver as a “teacher”, which generates action recommendations for small-scale scenarios (Sartoretti et al., 2019; Damani et al., 2021; Li et al., 2020; Skrynnik et al., 2024a; Wang et al., 2023a). To further enhance the RL training, rewards are extensively shaped by assigning handcrafted penalties for very

specific situations, such as collision attempts, out-of-boundary moves, and blocking, which is heuristically determined by a separate A* search (Sartoretti et al., 2019). While this may be promising for particular map categories, e.g., unstructured maps, it can fundamentally change our original objective of Equation 1 and lead to unintended side effects or *reward gaming* when used for novel maps with different structures (Devlin & Kudenko, 2011; Foerster et al., 2018; Skalse et al., 2022).

Pathfinding via Reinforcement and Imitation Multi-Agent Learning (PRIMAL) is the first learning-based MAPF approach, combining RL, imitation learning, and reward shaping (Sartoretti et al., 2019). PRIMAL employs a considerably complex neural network architecture composed of *convolutional neural networks (CNN)*, *recurrent neural networks (RNN)*, *multi-input streams* for an agent’s observation and its coordinates for training and execution. PRIMAL was succeeded by PRIMAL2, adding convention learning, i.e., specialization in handcrafted corridor scenarios (Damani et al., 2021), and SCRIMP, adding communication and a more advanced neural network architecture with residual networks and attention mechanisms (Wang et al., 2023a). Large neural networks can memorize training data due to overparameterization, which eases specialization in micro-aspects of particular MAPF problems (Baldi & Sadowski, 2013; Srivastava et al., 2014; Wager et al., 2013). However, tight specializations are detrimental to generalization, which is known as *overfitting* (Srivastava et al., 2014).

Alternative approaches focus on further micro-aspects, such as the locality of agent interaction and periodic neighborhood sensing, using path finding algorithms (Skrynnik et al., 2024a), Monte-Carlo algorithms (Skrynnik et al., 2024a), and large foundation models (Andreychuk et al., 2025). In most cases, a plain RL algorithm like PPO is used to train agent policies. We particularly highlight *CostTracer*, which is a PPO variant guided by modified observations and rewards via best-first-search, and used for decentralized online planning, e.g., Monte-Carlo tree search (Skrynnik et al., 2024a). Despite depending on an online planning algorithm for execution, CostTracer uses a simpler learning architecture than previous methods, like PRIMAL, PRIMAL2, and SCRIMP.

Most of the approaches mentioned above use independent RL, without any credit assignment mechanism, such as value factorization, according to Equations 4 and 5 (Rashid et al., 2020; Sunehag et al., 2018). Thus, there is no guarantee of convergence due to the non-stationarity, caused by simultaneously learning agents (Hernandez-Leal et al., 2017; Laurent et al., 2011), and effective coordination w.r.t. the IGM condition of Equation 5 (Son et al., 2019; Wang et al., 2020) – regardless of the engineering effort.

3.4.2 CURRICULUM LEARNING-BASED MAPF

Besides the lack of modern multi-agent RL techniques, the extensive engineering of methods described above also indicates a lack of exploration, which can cause convergence to poor policies, as mentioned in Section 2.3.1. To learn effective policies, an agent must explore its environment sufficiently to understand its dynamics (Sutton & Barto, 2018). The lack of exploration in prior learning-based MAPF approaches can have various reasons, such as bias through imitation learning, poorly designed reward functions, a strong dependence on additional replanning algorithms, and overfitting due to oversized neural networks and handcrafted scenarios (Silver et al., 2017; Skalse et al., 2022; Srivastava et al., 2014).

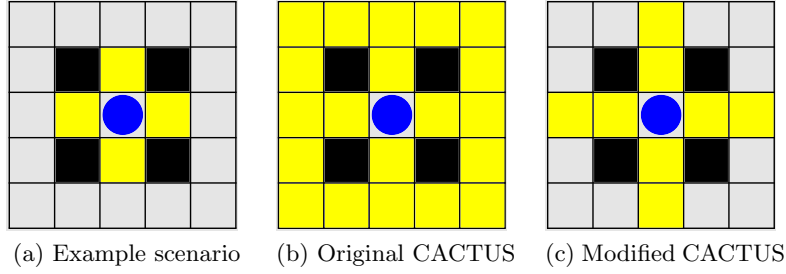


Figure 4: Curriculum scheme of CACTUS (Phan et al., 2024b). **(a)** Example scenario with an agent as a blue circle, black obstacles, and an initial *goal allocation area*. The goal is randomly allocated anywhere in the yellow region. **(b)** Original CACTUS variant increasing the goal allocation area with the Chebyshev distance. **(c)** Modified CACTUS variant, as used in this paper, increasing the goal allocation area with the geodesic distance.

Curriculum learning is promising to improve exploration while enabling methods that are both simple and effective. Some works proposed manually designed curricula to enhance PRIMAL, where agents focus on particular skills, e.g., navigation, before moving on to the next task (Zhao et al., 2023; Pham & Bera, 2023). However, these approaches also rely on oversized models and reward shaping, limiting computational efficiency and generalization potential (Zhao et al., 2023; Pham & Bera, 2023).

Confidence-based Auto-Curriculum for Team Update Stability (CACTUS) is the first auto-curriculum approach to learning-based MAPF (Phan et al., 2024b). As illustrated in Figure 4, CACTUS follows a simple reverse curriculum scheme of randomly placing goal locations within close vicinity of all agents and gradually increasing their goal allocation areas, according to the RL progress, which is assessed with a confidence-based measure. The size of the goal allocation areas is determined by a distance function, e.g., Euclidean, Chebyshev, or geodesic distance. In contrast to other learning-based MAPF approaches, CACTUS uses value factorization as a credit-assignment mechanism for decentralized coordination and convergence in a general way without domain-specific engineering (Phan et al., 2021; Rashid et al., 2020; Wang et al., 2020). The pseudocode and hyperparameters are provided in Appendix A.2.4. CACTUS demonstrated superior sample efficiency to PRIMAL with less than 5% of the PRIMAL effort regarding computational resources, training data, and neural network size. (Phan et al., 2024a) provides further extensions and theoretical insights about the effectiveness and efficiency of CACTUS.

3.4.3 SUMMARY AND LIMITATIONS OF PRIOR WORK

In the last half-decade, the prevalent focus on micro-aspects of particular MAPF problems and the extensive engineering of methods have become the norm in learning-based MAPF (Alkazzi & Okumura, 2024), leading to a lack of sample efficiency and generalization, and slow progress in our field with only a handful of publications in top-tier venues per year. This contrasts the fast-evolving multi-agent RL community (Gronauer & Diepold, 2022; Hernandez-Leal et al., 2019), which uses much simpler and general methods for complex and large-scale tasks, e.g., such as video games (Rashid et al., 2020; Foerster et al., 2018;

Phan et al., 2021, 2023; Yu et al., 2022). This trend bears the risk of “reinventing the wheel” and missing out on the latest advances that could help to improve and generalize our methods (Phan et al., 2021; Rashid et al., 2020; Son et al., 2019; Wang et al., 2020).

We provide an overview of state-of-the-art learning-based MAPF approaches and their additional components and mechanisms beyond RL and basic neural networks, such as feedforward networks, CNNs, and RNNs, in Table 1. The additions are grouped according to the level of bias they induce regarding domain-specific aspects:

- (I) The machine learning techniques make abstract assumptions, e.g., decomposability of tasks for curriculum learning, or expert data for imitation learning, but they can be transferred to other domains due to not being MAPF or navigation-specific.
- (II) The neural network architectures can have inductive biases that make certain assumptions about the data, e.g., CNNs for matrix or tensor data, RNNs for sequence data, and attention mechanisms for high-dimensional inputs of varying sizes. The generality of these architectures depends on the problem, e.g., matrices could represent images, spatial structures in a grid world, or adjacency matrices of graphs.
- (III) The additions are designed explicitly for MAPF or navigation-related aspects and cannot be trivially transferred to other domains, e.g., convention learning for corridors may be less relevant for airspace coordination, where avoiding movable obstacles is more important (thus requiring new handcrafted scenarios).

Table 1 highlights the complexity and bias induced by prior work like PRIMAL, PRIMAL2, and SCRIMP, especially regarding the additions in aspect group (III), which limits transferability to other domains and generalization beyond the scenarios of the training distribution. For example, methods specializing in corridors could do well in 2D multi-robot warehouses but not necessarily in 3D airspace coordination with movable obstacles of different sizes and variable speeds, e.g., birds and helicopters (Ho et al., 2022b, 2022a). Reward shaping also limits generalization since obstacle and collision avoidance might need different penalties for different map categories to avoid unintended side effects and reward gaming (Skalse et al., 2022).

As mentioned in Section 3.4.2, most of these additions are presumably (unintended) compensation measures for outdated multi-agent RL technology and insufficient exploration, as curriculum approaches like CACTUS do not require extensive engineering effort to learn effective agent policies (Phan et al., 2024b). This allows for much simpler and more general techniques, which can be transferred to other domains more easily. In addition to the technical merits, this also has positive societal benefits due to reduced energy consumption and easier usability for researchers and students without the need for vast computing resources.

So far, most works have focused on policy learning algorithms and used unstructured maps (Phan et al., 2024b; Sartoretti et al., 2019; Wang et al., 2023a), handcrafted scenarios (Damani et al., 2021), or the actual test maps (Skrynnik et al., 2024a, 2024b) for training, as shown in Figure 1, limiting their generalization potential.

To this end, we propose a *generative curriculum approach* using the VAULT to create artificial maps for sample-efficient and generalizing RL training. As specified in Table 1, our VAULT approach adds *unsupervised learning* to the curriculum learning to train a VAE for *map generation* that is unbiased toward the agent policies and, therefore, *flexibly usable*

Table 1: Overview of state-of-the-art learning-based MAPF approaches and their additional components and mechanisms beyond RL and basic neural networks, such as feedforward networks, CNNs, and RNNs. The additions are grouped according to the level of bias they induce w.r.t. domain-specific aspects, where **(I)** represents the least and **(III)** the most biased additions. Additions that are not directly related to any machine learning technique (except for additional data or inputs) are highlighted in italics.

	PRIMAL	PRIMAL2	SCRIMP	CostTracer	CACTUS	VAULT
(I) Machine Learning Techniques						
Curriculum Learning	✗	✗	✗	✗	✓	✓
Unsupervised Learning	✗	✗	✗	✗	✗	✓
Imitation Learning	✓	✓	✓	✗	✗	✗
(II) Neural Network Additions						
Multi-Input Stream	✓	✓	✓	✗	✗	✗
Residual Networks	✗	✗	✓	✓	✗	✗
Attention Mechanisms	✗	✗	✓	✗	✗	✗
(III) Domain-Specific Additions						
Reward Shaping	✓	✓	✓	✓	✗	✗
<i>Handcrafted Conventions</i>	✗	✓	✗	✗	✗	✗
<i>Path Finding Algorithms</i>	✓	✓	✓	✓	✗	✗
<i>Communication Channel</i>	✗	✗	✓	✗	✗	✗

for different training and testing purposes. We employ an *optimization algorithm* to search the latent space of the VAULT in order to improve sample efficiency and generalization of the agent policies without further updates or adjustments to the VAULT. In the following, we will describe all components of our curriculum approach in more detail.

4. Generative Curricula for Learning-Based MAPF

We now introduce our two-stage framework to generate curricula for learning-based MAPF via *Variational Autoencoder Utilized Learning of Terrains (VAULT)*, as illustrated in Figure 5. At Stage I, we train the VAULT as an artificial map generator using unsupervised learning. At Stage II, we use the fully trained VAULT to generate curricula based on maps G' or instances I' for *RL training*² of agent policies $\hat{\pi}$. Our goal is to improve sample efficiency and generalization of learning-based MAPF methods.

2. To avoid confusion about the term “*training*” regarding the VAULT and the actual agent policies $\hat{\pi}$, we refer to the latter as “*RL training*”, according to our focus explained in the Introduction.

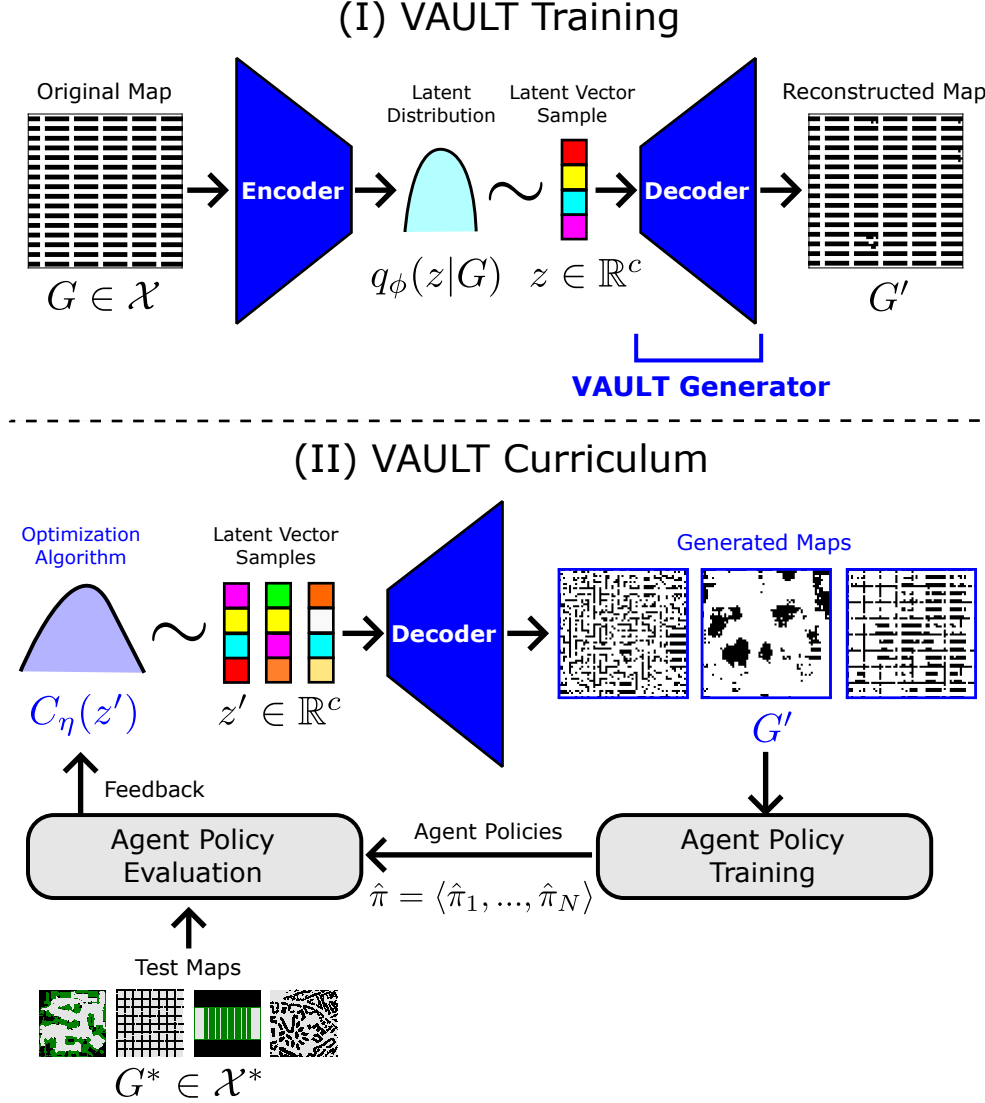


Figure 5: Overview of our two-stage framework. All blue components represent our contributions integrated into the established framework of learning-based MAPF, as shown in Figure 1, which is represented by the gray components in Stage II. **(I) VAULT Training:** A *variational autoencoder (VAE)* is trained on a set of maps $G \in \mathcal{X}$. The VAE decoder represents the *VAULT generator* or *VAULT* for short. **(II) VAULT Curriculum:** An optimization algorithm searches the latent space of the VAULT with a parametrized distribution $C_\eta(z')$ to generate maps for the RL training of agent policies $\hat{\pi}$. The policies $\hat{\pi}$ are evaluated with test maps $G^* \in \mathcal{X}^*$, listed in Table 2, for feedback to update the parameters η in order to refine the map generation. More details on Stage II are illustrated in Figure 11. Note that the VAULT is only trained *once at Stage I* and remains *unchanged* throughout Stage II, thus can be flexibly used for different training and testing purposes.

While Stage I only needs to be run once to train the VAULT (and potentially pruned versions of it), Stage II can be run arbitrarily often by reusing the fully trained VAULT for generative curricula without further updates to it. This flexibility enables further research with a common generative model without the need to train new ones from scratch.

4.1 Stage I: VAULT Training

We now describe our approach to training the VAULT using all maps G of a dataset $\mathcal{X}$, as depicted in Figure 5 (top). We provide intermediate results of this stage to visualize and motivate the introduced concepts for the subsequent sections.

4.1.1 DATA PREPARATION

For simplicity and computational efficiency, we encode all maps G as binary matrices, which can represent obstacle locations in a 2D grid world or the adjacency matrix of a graph with unit edge weights, according to Section 2.1. More general representations, e.g., for arbitrary edge weights or whole instances I with agent locations, will require more advanced learning models, such as transformers and graph attention networks (Vaswani et al., 2017; Veličković et al., 2018), which is out of scope of this work.

To train the VAULT, we create a map dataset $\mathcal{X}$ from the 2D grid maps of the public MAPF benchmark set (Stern et al., 2019), including the *map categories* $m \in \{\text{Random, Game, City, Warehouse, Room, Maze}\}$, as listed in Table 2. Empty maps are not included in $\mathcal{X}$ because they are trivial to reconstruct.

Table 2: The map categories m of the public MAPF benchmark set (Stern et al., 2019), considered in this paper. We list the test maps $G^* \in \mathcal{X}^*$ and their sizes, as used in our experiments in Section 6, and indicate if the map categories are used for training the VAULT and the CoreVAULT as its pruned version, which is explained later in Section 4.1.3.

	Category m	VAULT?	CoreVAULT?	Test Map G^*	Size of G^*
	Empty	✗	✗	<i>empty-48-48</i>	48×48
	Random	✓	✓	<i>random-64-64-10</i>	64×64
	City	✓	(✓)	<i>Paris_1_256</i>	256×256
	Game	✓	(✓)	<i>den520d</i>	256×257
	Warehouse	✓	(✓)	<i>warehouse-10-20-10-2-2</i>	170×84
	Room	✓	✓	<i>room-64-64-16</i>	64×64
	Maze	✓	(✓)	<i>maze-128-128-2</i>	128×128

For each map, we extract random 64×64 subgrids which are randomly rotated and flipped to augment and diversify our dataset (Shorten & Khoshgoftaar, 2019; Wang et al., 2017). Focusing on fixed-size (sub)maps simplifies the VAULT curriculum at Stage II so that we can optimize over a fixed-dimensional latent space. It also improves the diversity of our dataset because we can focus on different regions of the same map. Therefore, RL

training on such (sub)maps can reduce bias toward the original benchmark maps to some extent. Our dataset $\mathcal{X}$ is balanced such that it contains an equal amount of 2,000 (sub)maps per category m considered for training, i.e., $|\mathcal{X}| = 12,000$ (sub)maps in total.

4.1.2 TRAINING AND EVALUATION

The encoder *enc* of our VAE consists of several convolutional layers, which successively reduce the size of the original input G with $d = 64 \times 64 = 4,096$ dimensions. The final convolution of *enc* is processed by two fully connected output heads of dimension $c = 128 < d$ to obtain the mean vector $\mu(G)$ and covariance $\sigma^2(G)$ of the approximate latent space distribution $q_\phi(z|G)$, as shown in Figure 5 (top). A latent vector sample $z \sim q_\phi(\cdot|G)$ is used to reconstruct G via the decoder *dec*, which processes the c -dimensional vector z with a fully connected layer and several deconvolutional layers to restore the original map size of $d = 64 \times 64 = 4,096$. The decoder output is processed by a sigmoid function to normalize the output values between 0 and 1. More details are in Appendix A.1.1.

To train the VAE, we minimize the loss function of Equation 7, where the reconstruction loss function diff_{VAULT} is defined by the *log loss* or *binary cross-entropy loss*:

$$\text{diff}_{VAULT}(G, G') = \frac{1}{d} \sum_{b=1}^d [x_b \log x'_b + (1 - x_b) \log(1 - x'_b)] \quad (8)$$

where x_b and x'_b are the b th entry of the flattened binary matrices G and G' , respectively.

The complete VAE loss function, according to Equation 7, is calculated with our dataset $\mathcal{X}$ from Section 4.1.1 and minimized iteratively via gradient descent. The VAULT training is formulated in Algorithm 1, where $\mathcal{X}$ is the map dataset, *enc* is the encoder network, *dec* is the decoder network, representing the actual VAULT as a generative model, and λ is the VAE regularization or smoothing factor. The training stops when the VAE loss of Equation 7 converges or some time limit is exceeded.

Algorithm 1 VAULT: Variational Autoencoder Utilized Learning of Terrains (Stage I)

```

1: procedure TrainingTheVAULT( $\mathcal{X}$ , enc, dec,  $\lambda$ )
2:   Initialize the learning parameters of enc and dec
3:   while VAE not converged and time limit not exceeded do
4:      $\mathcal{L}_{enc,dec}^{VAE} \leftarrow 0$ 
5:     for map  $G$  in  $\mathcal{X}$  do
6:        $q_\phi(\cdot|G) \leftarrow \text{enc}(G)$  ▷ Encode map  $G$  into a latent space distribution
7:        $z \sim q_\phi(\cdot|G)$  ▷ Sample a latent vector
8:        $G' \leftarrow \text{dec}(z)$  ▷ Reconstruct map  $G$  from the latent vector sample
9:        $\mathcal{L}_G^{VAE} \leftarrow \text{diff}_{VAULT}(G, G')$  ▷ Reconstruction loss of Equation 8
10:       $\mathcal{L}_G^{VAE} \leftarrow \mathcal{L}_G^{VAE} + \lambda D_{KL}(q_\phi(z|G), \mathcal{N}(\mathbf{0}, \mathbf{I}))$  ▷ KL regularization
11:       $\mathcal{L}_{enc,dec}^{VAE} \leftarrow \mathcal{L}_{enc,dec}^{VAE} + \frac{1}{|\mathcal{X}|} \mathcal{L}_G^{VAE}$ 
12:   Calculate gradients of enc and dec via the VAE loss  $\mathcal{L}_{enc,dec}^{VAE}$ 
13:   Apply gradient descent to the parameters of enc and dec
14:   return dec ▷ The actual VAULT, our generative model for learning-based MAPF

```

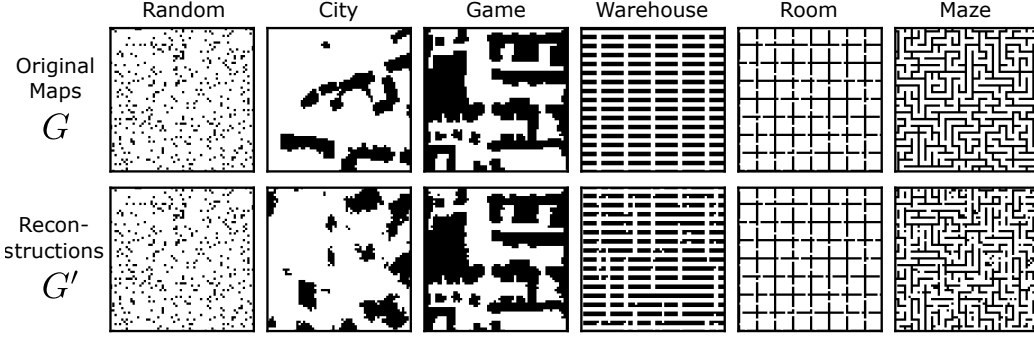


Figure 6: Map reconstruction examples of random 64×64 subgrids of the original benchmark maps (Table 2) using the fully trained VAE of the VAULT.

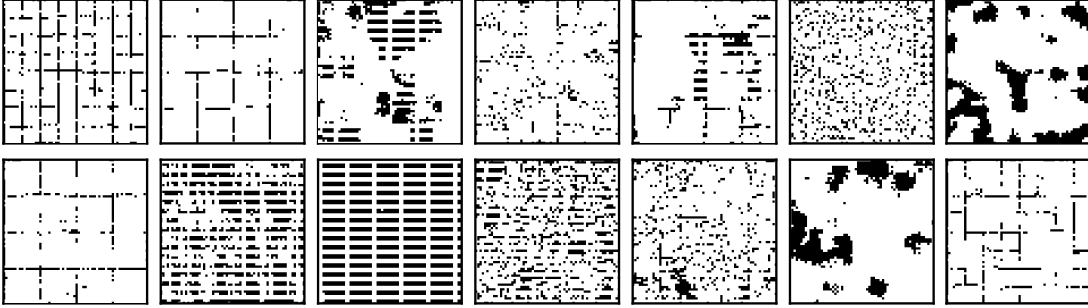


Figure 7: Examples of artificial maps G' generated with the VAULT using random sampling of its latent space (Figure 5).

Note that the training procedure is completely *unsupervised*. Even though we know the corresponding map categories m of all maps $G_m \in \mathcal{X}$, according to Table 2, the VAE itself is unaware of them and operates solely on the binary matrices without any labels.

Figure 6 shows some map reconstruction examples. Random and Room maps can be reconstructed accurately in most cases. While our VAE captures most of the coarse features of all map categories, it can be inaccurate regarding particular cells or regions, causing shifts in obstacles, e.g., in Warehouse or Maze maps. However, our VAE is unable to reconstruct any City map accurately.

By sampling random vectors $z' \in \mathbb{R}^c$ of dimension $c = 128$, we can use the VAULT, i.e., the decoder of our VAE, to generate artificial maps (Baldi, 2012; Kingma, 2013). Some examples are shown in Figure 7. The VAULT can replicate certain structures, like (incomplete) rooms, warehouse shelves, or random obstacles. Interestingly, many generated maps are morphed with features of different map categories m in a way that is non-existent in the original MAPF benchmark set (Stern et al., 2019). While this does not seem helpful for reconstruction purposes, it is *actually intriguing* for learning-based MAPF. The morphing introduces *emergent novel maps* G' that are not contained in the test map set $\mathcal{X}^*$, thus can improve generalization in learning-based MAPF without explicitly specializing in micro-aspects, handcrafted scenarios, or even the original test maps $G^* \in \mathcal{X}^*$, in contrast to (Sartoretti et al., 2019; Damani et al., 2021; Skrynnik et al., 2024b; Wang et al., 2023a).

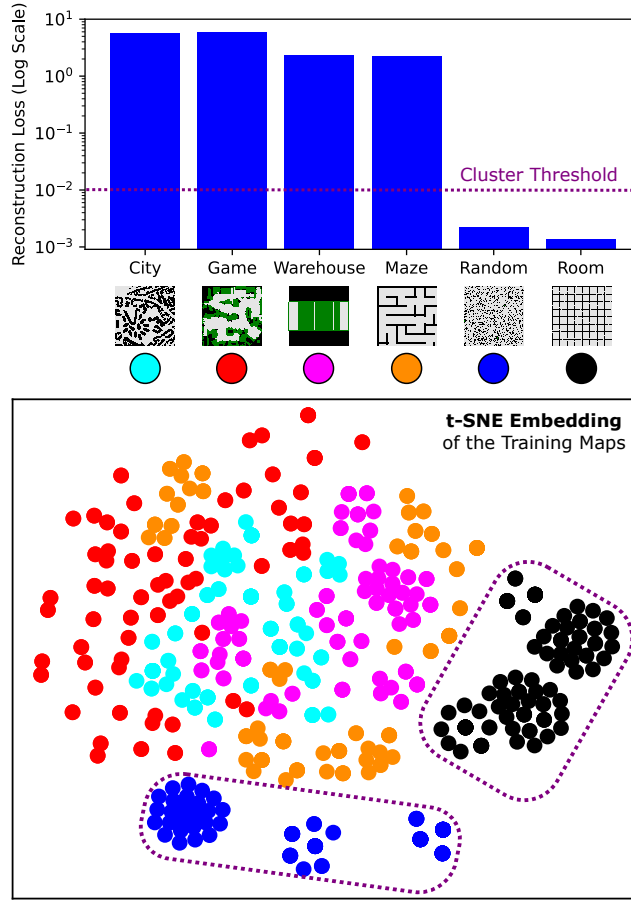


Figure 8: Evaluation of the VAULT model. **Top:** The average reconstruction losses per map category m used for training the VAULT (Table 2) in descending order. Note the logarithmic scale on the y-axis. **Bottom:** Two-dimensional t-SNE embedding of the learned latent representation of all training maps, depicted as colored points. The colors represent the corresponding map category shown at the top. The blue and black point clusters can be aggregated into separate clusters, as indicated by the dotted purple boxes and losses.

Figure 8 (top) shows the average reconstruction losses per map category m used for training the VAULT. According to the loss values, our VAE can easily reconstruct **Random** and **Room** maps with very low loss values. Figure 8 (bottom) shows a two-dimensional t-SNE embedding of the learned latent representation for each training map $G \in \mathcal{X}$, as a non-linear 2D projection of 128-dimensional z vectors, visualizing how the training data is aligned in the latent space. The VAE neatly clusters **Random** and **Room** maps (blue and black points, respectively), while all other map categories are mixed together in a larger and less cohesive structure. This alignment suggests that some map categories may be redundant and thus can be pruned for better generalization and efficiency (Sorscher et al., 2022; Toneva et al., 2019; Yang et al., 2023), which we will explore in the next section.

4.1.3 COREVAULT TRAINING VIA DATA PRUNING

In the machine learning community, a common viewpoint is that “*training bigger models on bigger datasets with larger computational resources is the sole key*” to progress in AI research (Sachdeva & McAuley, 2023). In contrast to this viewpoint, *data pruning* or *core set construction* techniques have been proposed to extract subsets $\hat{\mathcal{X}} \subseteq \mathcal{X}$ of high-quality data from the original dataset (Bachem et al., 2017; Sorscher et al., 2022; Toneva et al., 2019; Yang et al., 2023). Training on such a pruned dataset $\hat{\mathcal{X}}$ should not lead to reduced performance, e.g., prediction accuracy, but avoid tight coupling with irrelevant data, thus improving *generalization* and *training efficiency* by reducing the computational demand, e.g., for loss calculations (Sachdeva & McAuley, 2023; Sorscher et al., 2022).

Based on our VAULT evaluation in Figure 6, we propose a simple *category-based method* of constructing $\hat{\mathcal{X}} \subseteq \mathcal{X}$. We define $\hat{\mathcal{X}} = \bigcup_b \mathcal{G}_b$ via *clusters* $\mathcal{G}_b$ obtained through the

VAE $f(\cdot) = \text{dec}(\text{enc}(\cdot))$ ³ of the original VAULT in Section 4.1.2. The clusters are defined by a user-defined *loss threshold* δ_{rec} and the average reconstruction loss $\mathcal{L}_m^{AE} = \frac{1}{|\mathcal{X}|} \sum_{G \in \mathcal{X}} \text{diff}_{\text{VAULT}}(G, f(G))$ of each map category m . All maps G_m of a category m with $\mathcal{L}_m^{AE} < \delta_{\text{rec}}$ are assigned to a separate cluster $\mathcal{G}_b$. Otherwise, they are assigned to a *default cluster* $\mathcal{G}_{\text{default}}$. The pruned dataset $\hat{\mathcal{X}}$ is constructed by joining all clusters $\mathcal{G}_b$ and a subset $\mathcal{G}'_{\text{default}} \subseteq \mathcal{G}_{\text{default}}$, representing a single map category of the default cluster $\mathcal{G}_{\text{default}}$ as the *default representative*. Thus, we can discard all remaining maps in $\mathcal{G}_{\text{default}} \setminus \mathcal{G}'_{\text{default}}$.

Our data pruning method is formulated in Algorithm 2, where f represents our original VAE trained earlier in Section 4.1.2, $\mathcal{X}$ is our original dataset of Section 4.1.1, and δ_{rec} is a user-defined loss threshold. The resulting dataset $\hat{\mathcal{X}}$ can be used to train the *CoreVAULT*.

Algorithm 2 Data Pruning for the CoreVAULT (Stage I)

```

1: procedure DataPruning( $f, \mathcal{X}, \delta_{\text{rec}}$ )
2:    $\hat{\mathcal{X}} \leftarrow \emptyset$  and  $b \leftarrow 1$ 
3:    $\mathcal{M}_{\text{default}} \leftarrow \emptyset$  ▷ Map categories representing the default cluster  $\mathcal{G}_{\text{default}}$ 
4:   for map category  $m$  in {City, ..., Maze} do ▷ Maps for VAULT training (Table 2)
5:      $\mathcal{G}_b \leftarrow \{G \in \mathcal{X} | G \text{ belongs to category } m\}$  ▷ Cluster candidate
6:      $\mathcal{L}_m^{AE} = \frac{1}{|\mathcal{G}_b|} \sum_{G \in \mathcal{G}_b} \text{diff}_{\text{VAULT}}(G, f(G))$  ▷ Average reconstruction loss, Equation 8
7:     if  $\mathcal{L}_m^{AE} < \delta_{\text{rec}}$  then
8:        $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \cup \mathcal{G}_b$  and  $b \leftarrow b + 1$  ▷ Add cluster candidate to the pruned dataset
9:     else
10:       $\mathcal{M}_{\text{default}} \leftarrow \mathcal{M}_{\text{default}} \cup \{m\}$ 
11:   Define a default representative  $\mathcal{G}'_{\text{default}}$  via a random category  $m \in \mathcal{M}_{\text{default}}$ 
12:    $\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \cup \mathcal{G}'_{\text{default}}$ 
13:   return  $\hat{\mathcal{X}}$  ▷ The pruned dataset for training the CoreVAULT
    
```

Applying Algorithm 2 to our original dataset $\mathcal{X}$ with $\delta_{\text{rec}} = 10^{-2}$ leads to the **Random** and **Room** clusters, as shown in Figure 8. All remaining points represent the default cluster $\mathcal{G}_{\text{default}}$. Thus, $\hat{\mathcal{X}}$ consists of three map categories, namely **Random**, **Room**, and any other

3. In this case, we treat our VAE as a deterministic AE (Section 2.5.1) by using $z = \mu(G)$ for reconstruction.

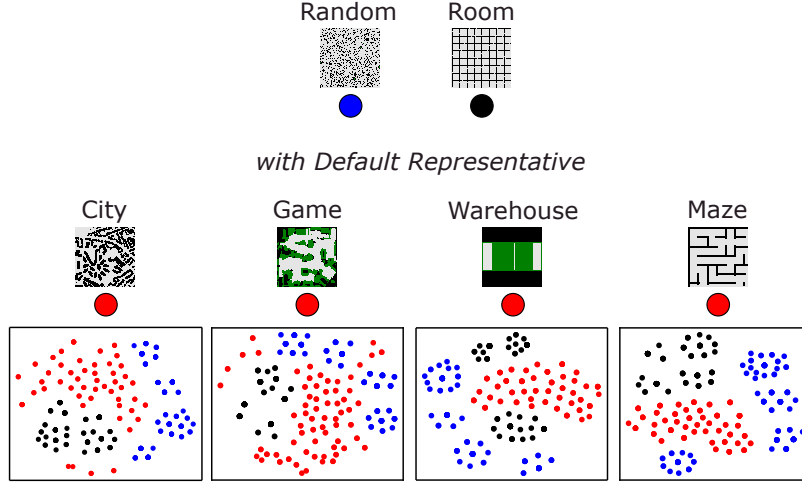


Figure 9: t-SNE embeddings of the learned latent representations of four CoreVAULT models trained with different default representatives, i.e., **Game**, **City**, **Warehouse**, or **Maze** maps. While **Random** and **Room** maps are always separated from each other, the default representatives tend to “connect” both categories by intercepting or surrounding their clusters.

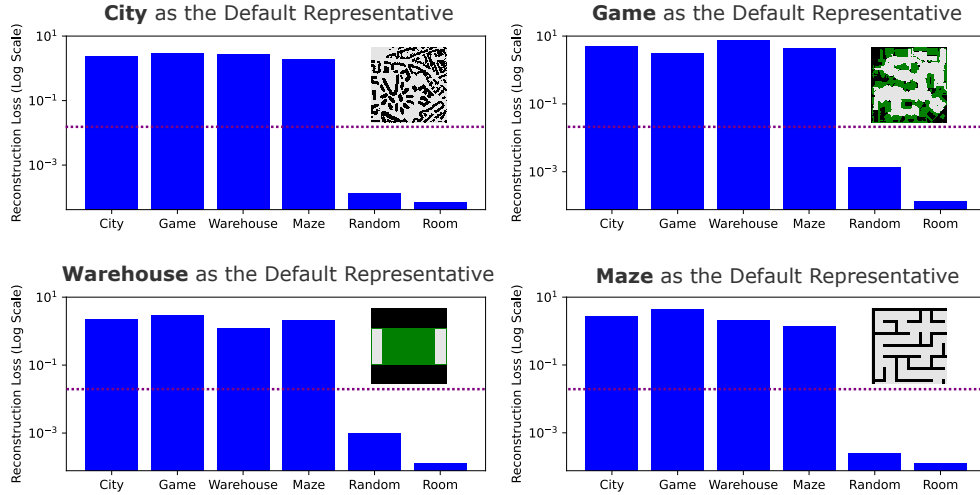


Figure 10: Average reconstruction losses of different CoreVAULT models with a default representative from $\mathcal{G}_{default}$, according to Algorithm 2 and Figure 8. Note the logarithmic scale on the y-axis. The purple dotted lines indicate the loss threshold $\delta_{rec} = 0.02$, i.e., the average reconstruction loss of our original VAULT, to determine the clusters in Figure 8.

category as the default representative. With $\hat{\mathcal{X}}$ we can further reduce bias in our generative model, i.e., the CoreVAULT, toward the test maps $G^* \in \mathcal{X}^*$ and improve generalization.

Including a default representative $\mathcal{G}'_{default} \subseteq \mathcal{G}_{default}$ in $\hat{\mathcal{X}}$ can help to avoid bias toward **Random** and **Room** maps which can be easily replicated by the VAULT. While **Random** and **Room** are always separated from each other in the latent space, according to Figures 8 and 9, the default representatives tend to “connect” both categories by intercepting or surrounding

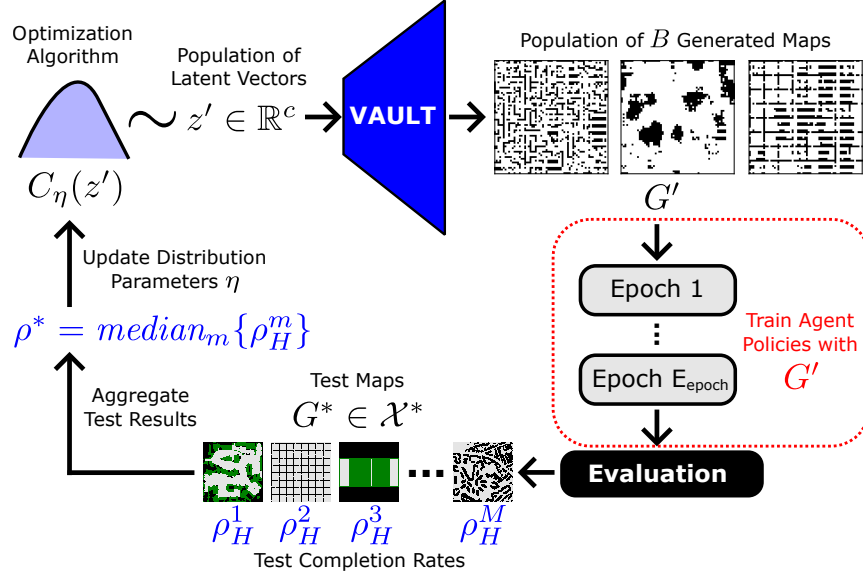


Figure 11: Overview of our basic curriculum scheme using the VAULT (or CoreVAULT). An optimization algorithm is used to search the latent space. Before each *training cycle* of $E_{epoch} \geq 1$ epochs, i.e., agent policy updates, a population of B artificial maps G' with $1 \leq B \leq E_{epoch}$ is generated, using a parametrized distribution $C_\eta(z')$. The agent policies $\hat{\pi}$ are trained on these maps using some RL algorithm (Phan et al., 2024b; Sartoretti et al., 2019; Skrynnik et al., 2024b). After E_{epoch} epochs of RL training, the agent policies $\hat{\pi}$ are evaluated in test maps G_m^* of different categories m , listed in Table 2, to determine the completion rate ρ_H^m per test map. The *median completion rate* ρ^* of all ρ_H^m is used to update the parameters η to refine the map generation for future training cycles.

their clusters, thus enabling a smooth transition between them. As shown in Figure 6, the maps of $\mathcal{G}_{default}$ are often morphed with other maps in generated versions, thus offering the potential to improve generalization to unseen maps that are substantially different from **Random** and **Room** maps. Our experimental results in Sections 6.3 and 6.4 suggest that omitting the default representative can affect the sample efficiency of RL training.

In principle, the pruned dataset $\hat{\mathcal{X}}$ can be refined further via fixed-point iteration by alternately running Algorithms 1 and 2 until $\hat{\mathcal{X}}$ does not change. Since the reconstruction losses of all CoreVAULT models in Figure 10 suggest that the fixed-point is already reached with our clustering in Figure 8, we defer any further investigation to future work.

Training a CoreVAULT model, in addition to our original VAULT, adds roughly 50% (200% if we train all four variants of Figures 9 and 10) of extra computational effort to our Stage I. However, this is only a one-time effort since the CoreVAULT can be reused in the subsequent Stage II without further updates or adjustments, as explained above.

4.2 Stage II: VAULT Curriculum

We can now use the fully trained VAULT (or CoreVAULT) for learning-based MAPF to generate curricula via artificial maps G' or instances I' . We will first propose a basic

curriculum scheme solely based on map generation for any learning-based MAPF method using latent space optimization, and then introduce a bi-level curriculum scheme to generate whole instances by combining our VAULT curriculum with CACTUS (Phan et al., 2024b).

4.2.1 BASIC CURRICULUM SCHEME

Figure 11 provides a general overview of our basic curriculum scheme. An optimization algorithm is used to search the latent space of the VAULT. RL training of agent policies $\hat{\pi}$ is conducted in *epochs* of $E_{episode} \geq 1$ *episodes*, i.e., instance trials of a maximum of H time steps before the instance is reset. After each epoch e , the agent policies $\hat{\pi}$ are updated using some RL algorithm (Phan et al., 2024b; Sartoretti et al., 2019; Skrynnik et al., 2024b).

Algorithm 3 Basic Curriculum Scheme with the VAULT or CoreVAULT (Stage II)

```

1: procedure VAULTCurriculum( $VAULT, \Theta, \Phi, \mathcal{X}^*$ )
2:   Initialize the parameters  $\eta, \theta, \omega$  of  $C_\eta, \hat{\pi}, \hat{Q}$ , respectively
3:    $\rho^+ \leftarrow 0$  ▷ Highest median completion rate so far
4:   while time budget not exceeded do
5:      $\mathcal{B}_{population} \leftarrow \emptyset$  and  $\mathcal{G}_{train} \leftarrow \emptyset$  ▷ Population of latent vectors and maps
6:     for counter  $b$  in  $1, \dots, B$  do
7:        $z' \sim C_\eta(\cdot)$ 
8:        $\mathcal{B}_{population} \leftarrow \mathcal{B}_{population} \cup \{z'\}$  and  $\mathcal{G}_{train} \leftarrow \mathcal{G}_{train} \cup \{VAULT(z')\}$ 
9:     for epoch  $e$  in  $1, \dots, E_{epoch}$  do
10:      Randomly select a map  $G'$  from  $\mathcal{G}_{train}$ 
11:      Run  $E_{episode}$  episodes with random instances on  $G'$ 
12:      Use RL algorithm  $\Theta$  to update  $\theta, \omega$  for agent policies  $\hat{\pi}$  and value function  $\hat{Q}$ 
13:      for test map category  $m$  in  $\{\text{Empty}, \dots, \text{Maze}\}$  do ▷ See Table 2
14:        Run  $E_{episode}$  episodes with predefined test instances on  $G_m^* \in \mathcal{X}^*$ 
15:        Record the completion rate  $\rho_H^m$  for the map category  $m$  represented by  $G_m^*$ 
16:       $\rho^* \leftarrow \text{median}_m\{\rho_H^m\}$ 
17:      if  $\rho^* > \rho^+$  then
18:        Use optimization algorithm  $\Phi$  to update  $\eta$  for  $C_\eta$  using  $\mathcal{B}_{population}$  and  $\rho^*$ 
19:       $\rho^+ \leftarrow \rho^*$ 
20:   return  $\hat{\pi}$  ▷ Fully trained agent policies for decentralized MAPF

```

Before each *training cycle* of $E_{epoch} \geq 1$ epochs, a population of B artificial maps G' with $1 \leq B \leq E_{epoch}$ is generated, using a parametrized distribution $C_\eta(z')$. The agent policies $\hat{\pi}$ are trained on these maps via $E_{episode}$ episodes. After E_{epoch} epochs of RL training, the agent policies $\hat{\pi}$ are evaluated in test maps $G_m^* \in \mathcal{X}^*$ of different categories m , listed in Table 2, to determine the completion rate ρ_H^m , i.e., the fraction of agents that reached their goal locations, per test map G_m^* . Unlike MAPF-specific objective measures, such as the travel time, flowtime, or makespan, whose magnitudes depend on the maps and instances, the completion rate $\rho_H^m \in [0, 1]$ is a *normalized measure* that can be aggregated straightforwardly over all map categories. We use the *median completion rate* ρ^* of all ρ_H^m to update the parameters η of C_η to refine the map generation for future training cycles.

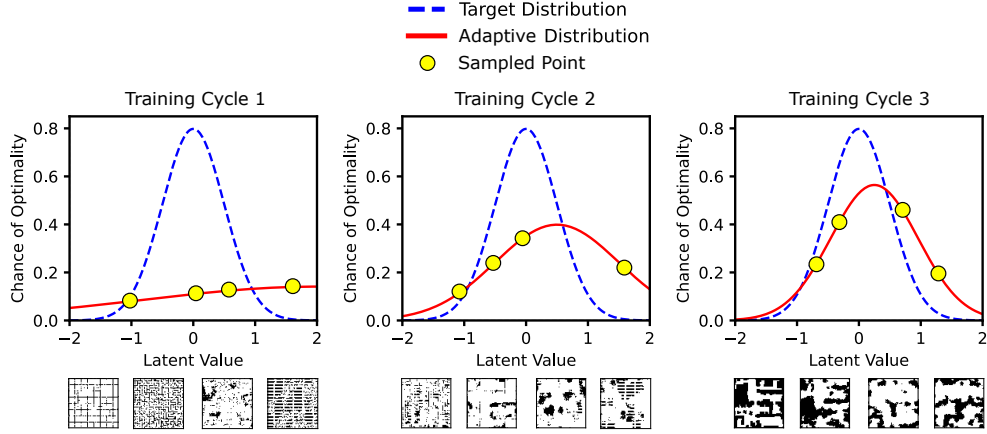


Figure 12: Illustration of the *Cross-Entropy (CE) method* for a one-dimensional Gaussian distribution, inspired by (Belzner, 2016). The dashed blue line represents the unknown target distribution of the optimal latent value. The solid red line represents an adaptive Gaussian distribution that is updated iteratively via maximum likelihood estimation based on previously generated yellow samples for map generation (examples at the bottom).

Alternative test measures for updating η would be the *average* and the *worst-case completion rate*. The former induces bias toward easy maps, whose high completion rates can skew the overall generalization assessment, while the latter induces bias towards difficult maps, where progress is slow and thus leads to a stagnating curriculum. The median offers a tradeoff between both extremes and is, therefore, our preferred test measure.

The complete formulation of our basic curriculum scheme is provided in Algorithm 3, where the *VAULT* is a generative model, e.g., represented by a VAE decoder, that can create maps G' from latent vectors z' , Θ is an RL algorithm to train agent policies $\hat{\pi}$, Φ is an optimization algorithm to search the latent space of the VAULT via a distribution C_η , and $\mathcal{X}^*$ is the test map set to periodically evaluate the agent policies $\hat{\pi}$ (Lines 13–15 in Algorithm 3) and update the distribution parameters η (Line 18 in Algorithm 3).

In the following, we briefly discuss two simple optimization algorithms for Φ and C_η :

Random Sampling The easiest way of implementing C_η is *random uniform sampling* of latent vectors z' with $\eta = \emptyset$. In this case, we generate random maps G' via the VAULT (or CoreVAULT) similar to the ones shown in Figure 6 without any adaptation, according to Line 18 of Algorithm 3. Therefore, the RL training and generalization progress completely depend on the underlying RL algorithm Θ and the quality of the generated maps G' .

Cross-Entropy (CE) Method The *Cross-Entropy (CE) method* offers an adaptive and black-box approach to find promising regions in the latent space to improve sample efficiency and generalization (Rubinstein, 1997). These regions are approximated with a parametrized distribution C_η , which is initialized as a flat distribution of high variance. The parameters η are continuously updated via maximum likelihood estimation, according to the median completion rate ρ^* , to gradually move the probability mass of C_η toward promising regions in the latent space. We consider a simplified CE variant by modeling C_η as a spherical Gaussian $\mathcal{N}(\mu, \Sigma)$ with a mean vector μ and a covariance $\Sigma = \sigma_{max}^2 \mathbf{1}$, representing a diagonal matrix

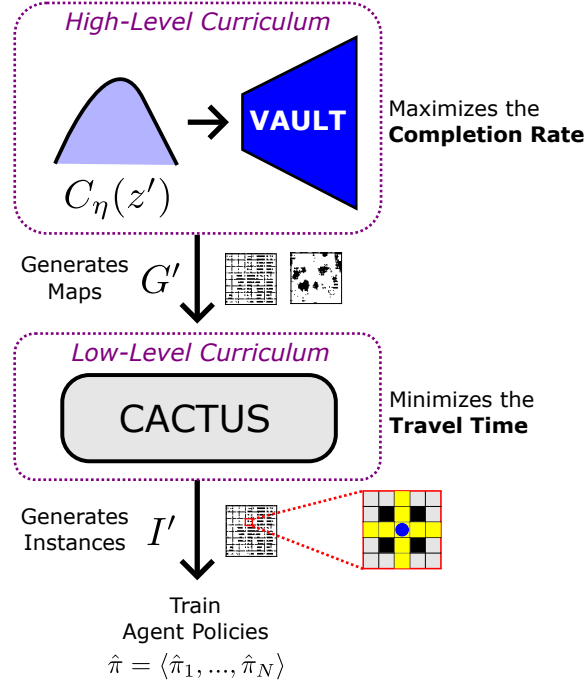


Figure 13: Illustration of our bi-level curriculum scheme combining our VAULT curriculum (Section 4.2.1) and CACTUS (Phan et al., 2024b) to generate whole instances I' for sample-efficient and generalizing RL training of agent policies $\hat{\pi}$. The formulation of CACTUS is provided in Appendix A.2.4 and Algorithm 4.

with each diagonal entry having the maximum variance σ_{max}^2 of C_η , w.r.t., all $c = 128$ dimensions of the latent space. Thus, we can efficiently sample and update $\mathcal{N}(\mu, \Sigma)$ by treating each dimension as a separate one-dimensional Gaussian with variance σ_{max}^2 , as illustrated in Figure 12. More sophisticated alternatives, such as ES, CMA-ES, CMA-MAE, that come with various additional hyperparameters, e.g., adaptive mutation and learning rates (Beyer & Schwefel, 2002; Fontaine & Nikolaidis, 2023; Fontaine et al., 2020; Hansen, 2016; Schwefel, 1993) are left for future work.

4.2.2 BI-LEVEL CURRICULUM SCHEME

Algorithm 3 represents a *map-based approach* to generate curricula for learning-based MAPF via the VAULT (or CoreVAULT) and can be used with any available RL algorithm (Phan et al., 2024b; Sartoretti et al., 2019; Skrynnik et al., 2024b). However, we can improve sample efficiency further by enabling *instance-based curricula* using a suitable RL algorithm.

CACTUS offers a simple but elegant way of generating instances I' based on given maps G' , as illustrated in Figure 4, by placing each agent i randomly on the map for its start location $v_{start,i}$ (Phan et al., 2024b). The corresponding goal locations $v_{goal,i}$ are randomly placed around $v_{start,i}$ inside an *allocation area*, which increases gradually based on the RL training progress of all agents. CACTUS solves the exploration problem in MAPF at the

instance level, offering high sample efficiency. However, its generalization potential depends on the training maps G' , which is a limiting factor, as discussed in Section 3.4.3.

Given our VAULT curriculum and CACTUS, we can now formulate a *bi-level curriculum scheme* based on Algorithm 3 to generate *whole training instances* I' for sample-efficient and generalizing RL training, as shown in Figure 13. Our scheme benefits from the high sample efficiency of CACTUS and the generalization potential of our VAULT curriculum:

- The **high-level curriculum** is represented by our VAULT curriculum using an optimization algorithm Φ (Algorithm 3). It generates artificial maps G' for RL training to support the generalization across all test (and previously unseen) maps by optimizing the median completion rate ρ^* , according to Algorithm 3 and Figure 11.
- The **low-level curriculum** is represented by CACTUS as the RL method Θ (Algorithm 3). CACTUS uses the generated maps G' of the VAULT curriculum to create instances I' by allocating start and goal locations to train effective agent policies $\hat{\pi}$ that optimize the travel time $l(p_i)$ per agent i via $\hat{Q} \approx Q_{tot}$, according to Equation 1.

4.3 Conceptual Discussion

Our approach represents a modular framework for learning-based MAPF, where each proposed component operates in a black-box manner with limited dependencies between each other and without considering specific micro-aspects of the underlying problem.

The key component of our framework is the VAULT, which serves as a map generator to improve sample efficiency and generalization of learning-based MAPF methods. The VAULT is trained via unsupervised learning on available maps and does not depend on any labels regarding map categories, RL algorithms, or previously trained agent policies, in contrast to (Bolland et al., 2022; Dennis et al., 2020). Thus, we can flexibly use our trained VAULT model for various purposes, e.g., RL training of different agent policies or tasks, and unbiased testing. The dependence on available maps can be further reduced by training the CoreVAULT on a pruned dataset $\hat{\mathcal{X}} \subseteq \mathcal{X}$. However, we cannot reduce our dataset arbitrarily, as our experimental results in Sections 6.3 and 6.4 suggest that some maps, e.g., Room maps, are crucial for sample efficiency and generalization.

Our VAULT curriculum in Algorithm 3 can be used with any available optimization and RL algorithm without specific assumptions about micro-aspects. For simplicity, we use random sampling and the CE method for optimization, which fully operate in a black-box manner. In contrast to prior work, which operates on the vertex space $\mathcal{V}$ under domain-specific constraints, our optimization algorithms do not require explicit structural information of the generated maps but only the dimensionality c of the VAULT latent space. They are efficient because random sampling does not require any update, and our CE variant adapts its Gaussian for each dimension independently via incremental updates to the mean and variance. For further improvement, we can easily employ more advanced optimization algorithms without affecting the VAULT (or CoreVAULT).

Our bi-level curriculum scheme represents a neat combination of our VAULT curriculum and CACTUS, which are both general methods, according to Section 3.4 and Table 1, and do not consider specific micro-aspects. Both methods complement each other, where our VAULT curriculum optimizes the generation of maps by maximizing the completion rate,

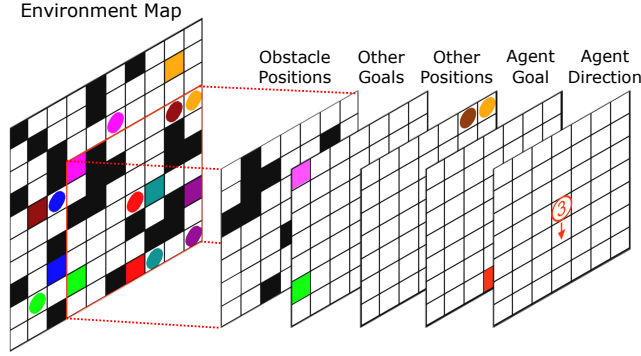


Figure 14: Example of an individual observation $o_{t,i}$ of the red agent i in a 2D grid world (Phan et al., 2024b). Agents are represented as colored circles, their goals as squares of the same color, and obstacles as black squares. The observation is encoded by five channels: The locations of obstacles, other agents’ goals, nearby agents, the location of the goal $v_{goal,i}$, and the geodesic distance and direction of agent i to its goal $v_{goal,i}$.

while CACTUS creates instances based on the generated maps by minimizing the agent travel times. Thus, our bi-level curriculum scheme can optimize the MAPF objective of Equation 1 in a sample-efficient and generalizing way, as we will demonstrate in Section 6.

While each component could be explicitly engineered toward even better performance, e.g., by integrating domain knowledge, shaping objective and reward functions, or choosing a different distribution for optimization, we explicitly focus on a *simple setup* to avoid distraction from our main concepts and goal. The rest is up to future work.

5. Experimental Setup

In our experiments, we focus on 2D grid worlds, where each vertex $v \in \mathcal{V}$ represents a cell with a maximum degree of four. Each agent i observes the environment through a 7×7 field of view encoded as a multi-channel image $o_{t,i}$, as illustrated in Figure 14, which is independent of the actual map size (Phan et al., 2024b; Sartoretti et al., 2019).

5.1 Maps

Training Maps We employ different map generators, i.e., the *VAULT* and ablations without Random or Room maps, the *CoreVAULT*, *MAP-Elites*, *PRIMAL maps*, and the *Original Maps* $G \in \mathcal{X}$. Unless stated otherwise, the VAULT generator is always used together with the CE method, as described in Section 4.2.1. MAP-Elites is inspired by (Zhang et al., 2023b) and optimizes maps using QD and the median completion rate ρ^* (Lehman & Stanley, 2011; Mouret & Clune, 2015). The QD-optimized maps are stored in an archive, according to their similarity to the test maps $G^* \in \mathcal{X}^*$ and their obstacle density relative to the most similar test map. PRIMAL maps are generated with random obstacles, according to a density ratio sampled from a triangular distribution between 0 and 0.5 with a peak at 0.33 (Sartoretti et al., 2019). The Original Maps serve as a proxy method for

the NCA generator (Zhang et al., 2023a), which replicates known structures to larger map sizes without altering them. More details are provided in Appendix A.2.2.

Test Maps We evaluate our VAULT curriculum on 25 predefined scenarios for all test maps $\mathcal{G}^* \in \mathcal{X}^*$ from the public MAPF benchmark set (Stern et al., 2019), as listed in Table 2, as well as artificial maps randomly generated with the *VAULT-B*, representing another separate VAULT model, which is trained on all training maps $G \in \mathcal{X}$ from Section 4.1.1 using Algorithm 1. We use the VAULT-B to provide 25 pre-generated test maps and scenarios, denoted as **Generated**, independent of the first VAULT model, which is used for RL training. According to Table 2, the test map sizes can differ from the 64×64 format of the VAULT and MAP-Elites maps. However, this is not an issue for our agent policies $\hat{\pi}$ which make decisions based on a 7×7 field of view, as illustrated in Figure 14, regardless of the actual map size (Sartoretti et al., 2019; Phan et al., 2024b). For conciseness, our main paper focuses on the results of the map categories **Random**, **Game**, **City**, **Warehouse**, and **Generated**, but we provide additional results regarding the other test maps in Appendix B.

5.2 Algorithms

Our base RL algorithm is CACTUS due to its simplicity and high sample efficiency compared to other methods (Phan et al., 2024b). The original CACTUS formulation used the Chebyshev distance, as depicted in Figure 4b, for measuring the allocation radius R_{alloc} . This can be detrimental to structured maps, as some locations inside the goal allocation area may require long detours and, consequently, more exploration. To mitigate this problem, we use the geodesic distance, as depicted in Figure 4c, for ensure reachability of all locations inside the goal allocation area, as recommended in (Phan et al., 2024a).

We also provide the pure RL variants of PRIMAL, PRIMAL2, and CostTracer, i.e., without imitation learning, integrated path finding, and communication channels, according to Table 1, to compare with CACTUS. To distinguish between PRIMAL and PRIMAL2, we include convention learning by pre-training PRIMAL2 on handcrafted corridor scenarios, as suggested in (Damani et al., 2021). For a fair and computationally efficient comparison, we trained all RL methods using the same number of training episodes, the same map generators, and the same neural network architectures, as suggested in (Phan et al., 2024b) to avoid any overparameterization bias. More details are provided in Appendix A.2.3.

We also compare our VAULT curriculum with traditional centralized MAPF solvers, namely CBSH, MAPF-LNS, and LaCAM*. CBSH is an optimal solver with completeness guarantees, meaning that it will find an existing solution with a sufficient time budget (Felner et al., 2018). Due to its optimality, CBSH serves as an absolute lower bound baseline regarding the (normalized) flowtime. CBSH uses CG, DG, and WDG as high-level admissible heuristics (Felner et al., 2018; Li et al., 2019a), as well as symmetry breaking techniques (Li et al., 2019b). MAPF-LNS is an anytime algorithm without any optimality or completeness guarantees (Li et al., 2021). LaCAM* is an anytime algorithm that guarantees completeness and eventual optimality, i.e., given a sufficient time budget (Okumura, 2023). In our work, LaCAM* optimizes the makespan, which we also use to ensure that all test instances are solvable within $H = 256$ steps.

5.3 Computing Infrastructure

All experiments were run on a high-performance computing cluster with CentOS Linux, Intel Xeon 2640v4 CPUs, and 64 GB RAM.

6. Experimental Results

In this section, we run RL experiments with our VAULT curriculum, i.e., Stage II of our approach, as explained in Section 4 and illustrated in Figure 5. Therefore, we only update the agent policies $\hat{\pi}$ and value functions $\hat{Q}$ without training the VAEs of the VAULT (and CoreVAULT) further. All RL training runs are conducted with $N = 32$ agents.

For each experiment, all learning-based MAPF approaches are run 20 times over 4,000 epochs to report the average progress and the 95% confidence interval. A training cycle consists of $E_{epoch} = 50$ epochs with $E_{episode} = 10$ episodes each. Thus, each learning-based MAPF approach is trained with $40,000 = 4,000 \cdot E_{episode}$ episodes in total. PRIMAL2 is additionally pre-trained with 20,000 randomly generated corridor observations for convention learning before the first RL training epoch. An episode is terminated after $H = 256$ time steps or when a completion rate of $\rho_t = 1$ is achieved at some time step $t < H$. We use the following measures to assess the *effectiveness* of all approaches:

- **Completion Rate:** The average rate of agents reaching their goals. We use it to assess the high-level curriculum and the suitability of the map generators for RL training.
- **Average Travel Time** or normalized flowtime: The average path length of all agents. We use it to assess the low-level curriculum of CACTUS based on the map generator. It serves as a proxy measure of the flowtime for better comparability since not all approaches are able to fully complete all instances, especially of challenging maps.

We report the RL training progress of each approach to assess its *sample efficiency*, i.e., the improvement speed w.r.t. the number of epochs. After each training cycle, we extract the current agent policies and evaluate them in the predefined instances of the test maps $\mathcal{G}^* \in \mathcal{X}^*$ listed in Table 2 to measure the completion rate and average travel time, according to Lines 13–15 in Algorithm 3. To assess *generalization*, we evaluate the fully trained agent policies with different numbers of agents $N = \{25, 50, 100, 200, 400, 800\}$ in the test maps. Furthermore, we evaluate the generalization of the agent policies to artificial maps randomly generated with the VAULT-B.

Since the completion rate and the average travel time are highly correlated in our results, we primarily focus on the completion rate in the paper to assess our VAULT curriculum, according to Figures 11 and 13, and only compare our achieved travel times with traditional MAPF solvers in Section 6.5. Additional comparisons regarding the average travel time and other test maps are provided in Appendix B.

6.1 Comparison of Different Reinforcement Learning Methods

Setting In this preliminary experiment, we compare CACTUS, CostTracer, PRIMAL, and PRIMAL2, using the VAULT and the Original Maps for RL training to assess their sample efficiency and suitability as our base RL algorithm for further experiments with our

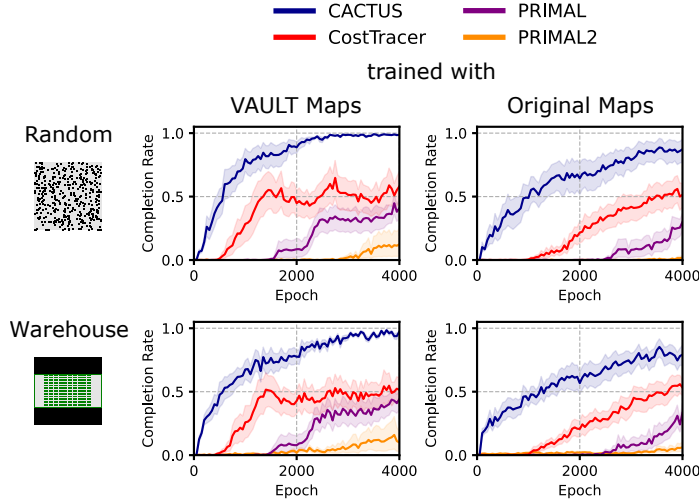


Figure 15: Completion rate progress of CACTUS, CostTracer, PRIMAL, and PRIMAL2 w.r.t. the training epochs in **Random** and **Warehouse** maps. The training maps are randomly generated with the VAULT (left) or randomly chosen from the Original Maps, according to Table 2. Before the first RL training epoch 0, PRIMAL2 is pre-trained on 20,000 handcrafted corridor observations. Shaded areas show the 95% confidence interval.

basic VAULT curriculum scheme (Section 4.2.1). For a fair comparison of the training map quality, we use random sampling for both map generators instead of the CE method.

Results The results for the **Random** and **Warehouse** maps are shown in Figure 15. All methods progress faster when trained with the VAULT. CACTUS progresses fastest in all cases with CostTracer being the second fastest method. PRIMAL makes slight progress and PRIMAL2 only improves visibly when trained with the VAULT maps.

Discussion The VAULT can generate training maps that improve sample efficiency of any RL algorithm compared to the Original Maps. The results also confirm that extensive engineering and specialization in micro-aspects are not beneficial for sample efficiency and generalization, as the simplest methods, according to Table 1, namely CACTUS and CostTracer, perform best in all settings. As discussed in Sections 3.4.2 and 4.2.2, CACTUS progresses fastest due to effectively addressing the credit assignment and exploration problem. CostTracer explores completely randomly without the support of Monte-Carlo tree search, while PRIMAL and PRIMAL2 rely on shaped rewards, which induce bias toward specific micro-aspects that are not transferable to arbitrary maps in general. PRIMAL2 performs worst, confirming that specializing in handcrafted scenarios is detrimental to RL training when the neural networks are not large enough to memorize them. We continue using CACTUS as our base RL algorithm in the following due to its high sample efficiency.

6.2 Curriculum Learning with the VAULT

Setting We now assess our VAULT curriculum with CACTUS. We evaluate the sample efficiency and generalization capabilities of our bi-level curriculum scheme (Section 4.2.2)

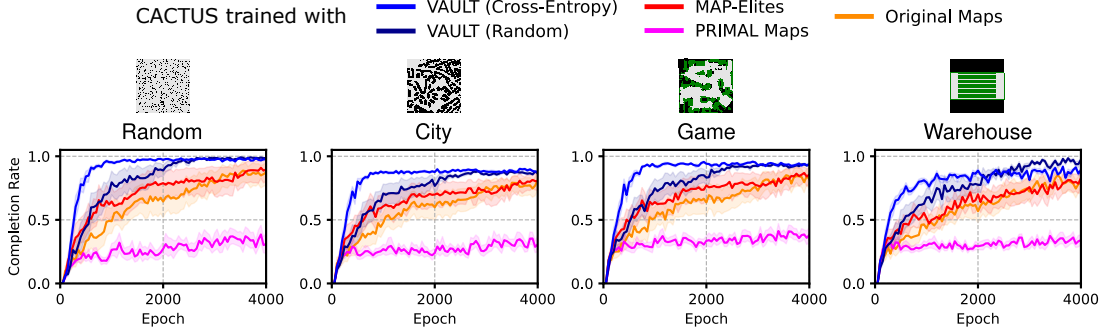


Figure 16: Completion rate progress of CACTUS trained with the VAULT, MAP-Elites, PRIMAL maps, or the Original Maps over **Random**, **City**, **Game**, and **Warehouse** test maps w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

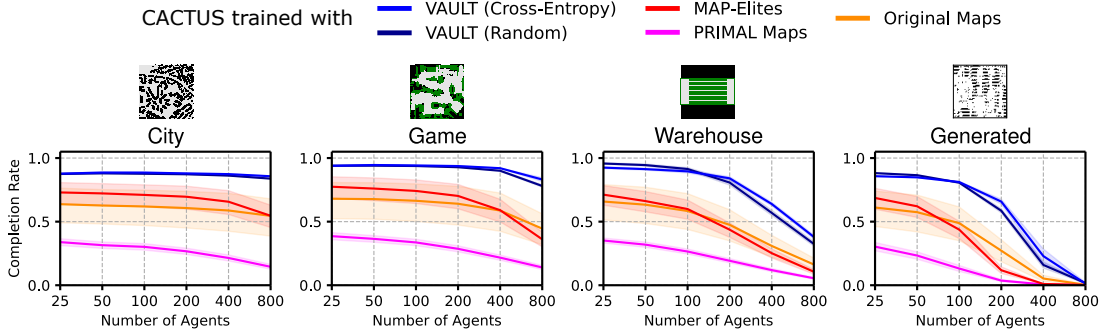


Figure 17: Test completion rate of CACTUS trained with the VAULT, MAP-Elites, PRIMAL maps, or the Original Maps over **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) test maps w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

by employing the CE method. The quality of the generated maps is assessed by employing random sampling in the latent space of the VAULT. Both approaches are compared with alternative map generators, such as MAP-Elites and PRIMAL Maps. The Original Maps serve as an alternative provider for training maps. For simplicity, we denote the resulting agent policies according to their respective map generators, e.g., VAULT policies.

Results The training results are shown in Figure 16. Our VAULT policies progress fastest when using the CE method. Our VAULT policies using random sampling progress second fastest. MAP-Elites policies progress slightly faster than Original Map policies. PRIMAL Map policies progress the worst, never achieving a completion rate of over 50%.

The test results for **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) maps, are shown in Figure 17. Our VAULT policies perform best, while MAP-Elites policies slightly outperform Original Map policies in settings with fewer agents, e.g., $N \leq 50$. PRIMAL Map policies generalize worst in all test maps. In the **Generated** maps, the completion rate of all policies deteriorates fastest with an increasing number of agents.

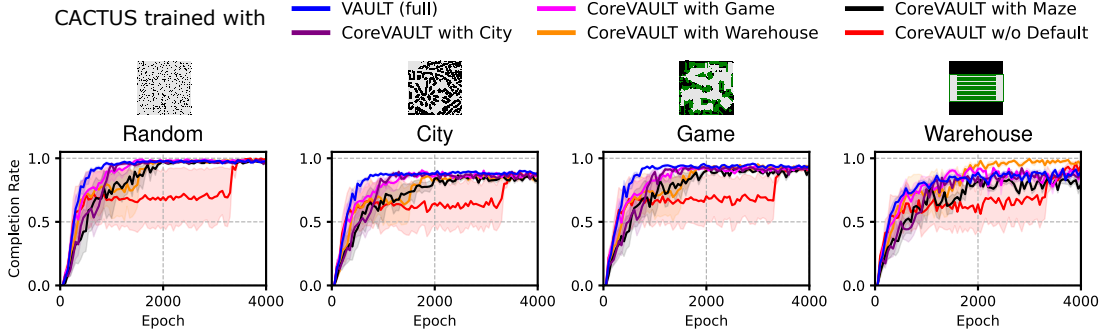


Figure 18: Completion rate progress of CACTUS trained with the VAULT and different variants of the CoreVAULT over **Random**, **City**, **Game**, and **Warehouse** test maps w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

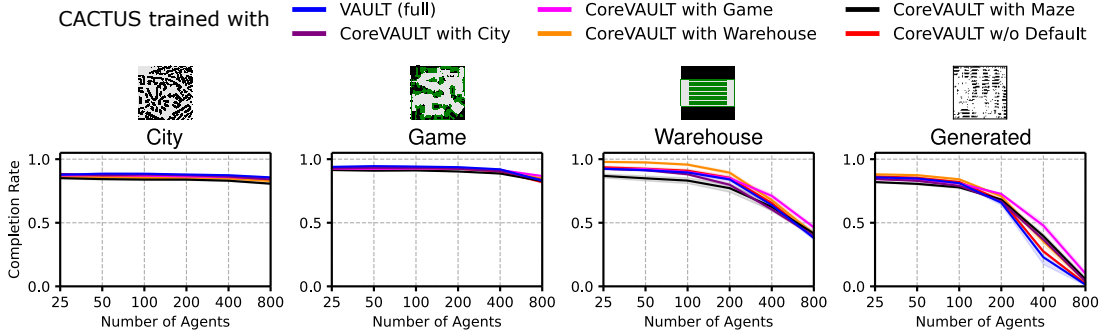


Figure 19: Test completion rate of CACTUS trained with the VAULT and different variants of the CoreVAULT over **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) test maps w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

Discussion The results confirm the effectiveness of our bi-level curriculum scheme, proposed in Section 4.2.2. The combination of the VAULT, the CE method, and CACTUS leads to significant improvements in sample efficiency and generalization, compared to all non-VAULT alternatives. The random sampling variant of our VAULT curriculum confirms the usefulness of the generated VAULT maps for RL training since it leads to better sample efficiency and generalization than alternative map generators, such as MAP-Elites and PRIMAL maps. Training with the Original Maps can be viewed as a specialization on “edge cases” inducing bias toward specific maps (Figure 2a). In contrast, the VAULT manages to “smooth” these edge cases via latent representations, as illustrated in Figure 8, resulting in morphed maps (Figure 6). This latent smoothing eases RL training substantially and improves generalization due to the reduced “edge case” bias. The **Generated** maps in Figure 17 lead to a steeper performance decrease than **Game**, **City**, and **Warehouse** maps, indicating that the VAULT-B can generate sufficiently challenging maps for testing.

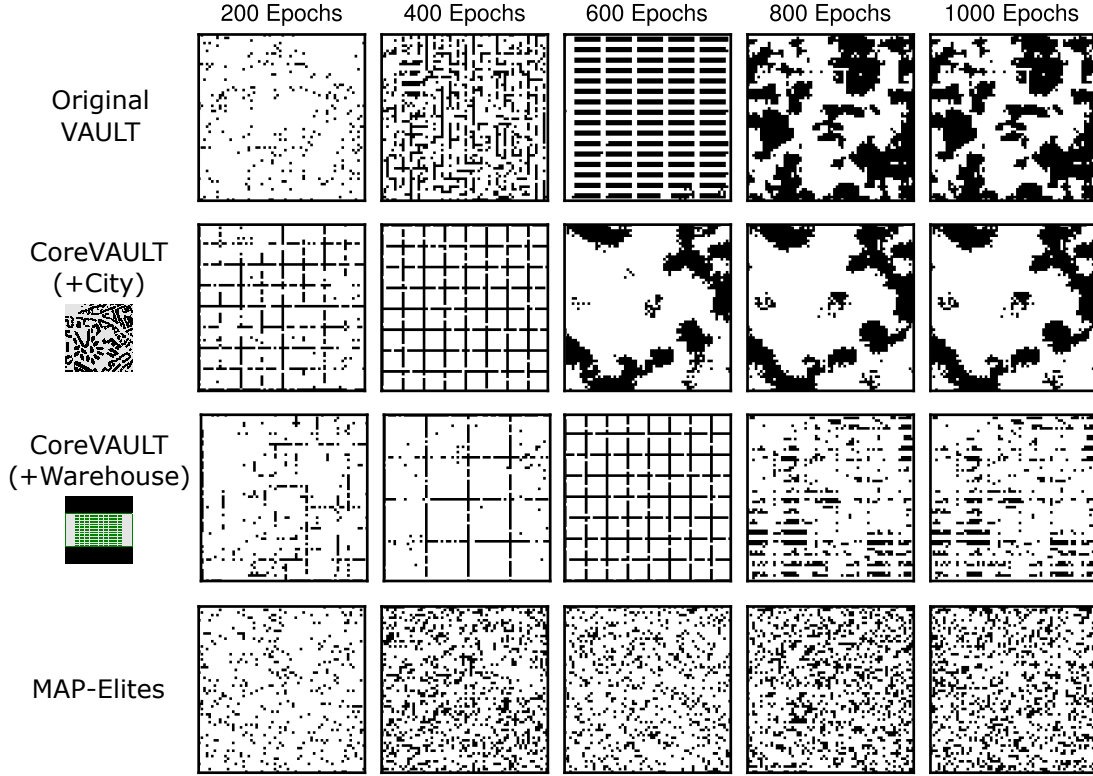


Figure 20: The evolution of maps generated with the original VAULT, the CoreVAULT with **City** and **Warehouse** as default representatives, and MAP-Elites over the first 1,000 epochs of RL training.

6.3 Curriculum Learning with the CoreVAULT

Setting Next, we evaluate the effect of data pruning by assessing our VAULT curriculum with different CoreVAULT models, specified in Section 4.1.3 and Figures 9 and 10. We also provide a CoreVAULT model trained without any default representative.

Results The training results of the CoreVAULT policies are shown in Figure 18. Training with the original VAULT progresses fastest, while all CoreVAULT policies with a default representative progress second fastest without significant differences between each other. Omitting the default representative affects the sample efficiency notably, thus requiring more epochs to keep up with the other CoreVAULT policies eventually.

The test results for **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) maps, are shown in Figure 19. After 40,000 episodes of RL training, all policies achieve similar performance, regardless of which VAULT or CoreVAULT model was used.

Figure 20 visualizes the evolution of maps generated with the original VAULT, the CoreVAULT with **City** and **Warehouse** as default representatives, and MAP-Elites over the first 1,000 epochs of RL training. The original VAULT starts generating very randomly distributed and morphed maps with structures of different map categories before converging to a map that resembles an artificial **Game** map. The CoreVAULT models start generating

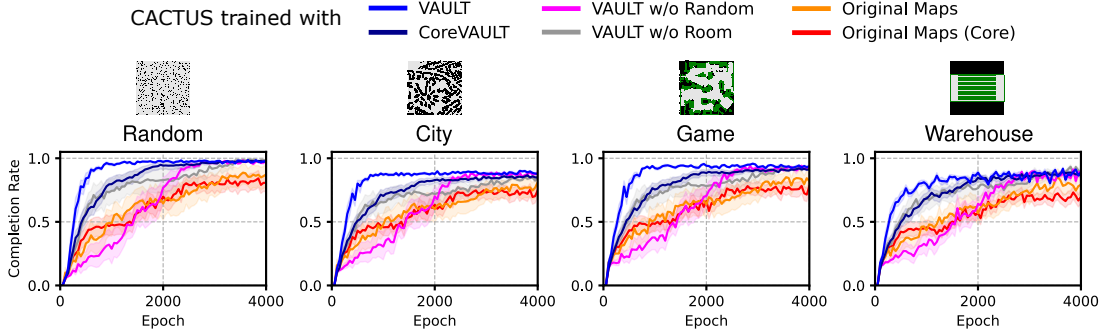


Figure 21: Completion rate progress of CACTUS trained with the VAULT and its ablations (without **Random** and **Room** maps, respectively), the CoreVAULT, or the Original Maps (pruned or not) over **Random**, **City**, **Game**, and **Warehouse** test maps w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

elements of **Random** and **Room** maps before converging to maps that resemble their respective default representatives partially morphed with **Random** elements. MAP-Elites creates seemingly unstructured maps with varying obstacles, known from (Zhang et al., 2023b).

Discussion The results confirm the generalization potential of using data pruning for training the CoreVAULT. Despite being trained on only half of the training maps at most, our CoreVAULT policies can generalize at least as well as the original VAULT policies. Our experiments also confirm the necessity of including default representatives in the CoreVAULT to improve sample efficiency. According to the evolution of maps, as shown in Figure 20, the CoreVAULT models first generate elements of **Random** and **Room** maps, indicating their importance in the early stages of RL training. Later on, the maps adopt elements from the default representatives which smooth the latent spaces as “connectors” (Figure 9) and ease optimization via the CE method. This is further supported by the reduced sample efficiency when omitting the default representative. In this case, the CE method requires more time to explore the latent space before eventually converging to a promising region, i.e., some unstructured **Random** map. Due to the similar sample efficiency and generalization, we report the aggregated results of all CoreVAULT policies (with a default representative) in the following experiments.

6.4 Curriculum Learning with VAULT Ablations

Setting Given the positive results of the CoreVAULT, we assess the importance of the **Random** and **Room** maps, which are not part of the default cluster $\mathcal{G}_{default}$ but represent separate clusters in the latent space of the original VAULT, according to Section 4.1.3 and Figure 8. Therefore, we assess the VAULT ablations excluding **Random** and **Room** maps from the dataset $\mathcal{X}$ with the original VAULT and CoreVAULT policies. As additional baselines, we use the Original Maps and a pruned variant only consisting of the pruned dataset $\hat{\mathcal{X}}$ like the corresponding CoreVAULT models, which we denote as *Original Maps (Core)*.

Results The training results of the VAULT and its ablations without **Random** and **Room** maps, respectively, and the CoreVAULT are shown in Figure 21. Training with the original

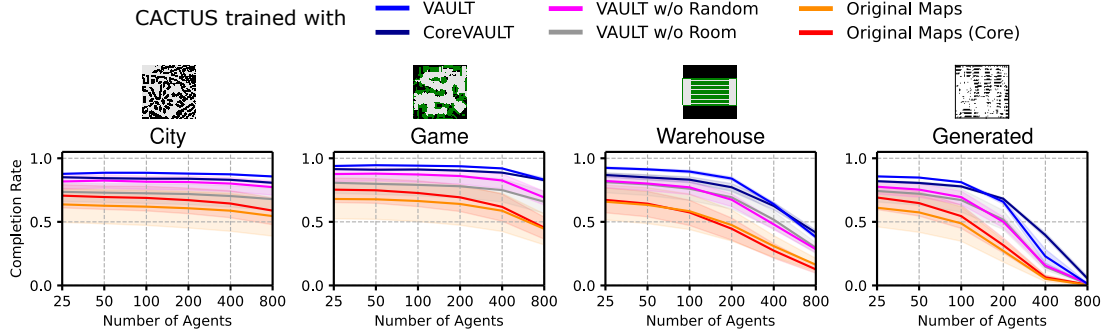


Figure 22: Test completion rate of CACTUS trained with the VAULT and its ablations (without **Random** and **Room** maps, respectively), CoreVAULT, or the Original Maps (pruned or not) over **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) test maps w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

VAULT progresses fastest, while our CoreVAULT policies progress second fastest. Omitting **Random** maps affects the sample efficiency notably, thus requiring more epochs to keep up with the original VAULT and CoreVAULT policies eventually. Omitting **Room** maps leads to a slightly worse sample efficiency than the CoreVAULT policies. The Original Map (Core) policies progress similarly to the Original Map policies without any degradation.

The test results of the VAULT and its ablations without **Random** and **Room** maps, respectively, and the CoreVAULT for **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) maps are shown in Figure 19. After 40,000 episodes of RL training, the agent policies trained with the VAULT ablations underperform the original VAULT and CoreVAULT policies with an increasing number of agents, while generally outperforming the Original Map policies. Omitting **Room** maps leads to higher sensitivity toward the larger maps, namely **City** and **Game**.

Discussion The results confirm the importance of **Random** and **Room** maps for training the VAULT and CoreVAULT. Unlike the maps of the default cluster $\mathcal{G}_{default}$, **Random** and **Room** maps cannot be omitted from the (pruned) dataset $\hat{\mathcal{X}}$ without affecting sample efficiency and generalization of the agent policies, despite the ablated VAULT being trained on more maps than the CoreVAULT models. **Room** maps seem to be especially important regarding the early stages of RL training, as indicated in Figure 20. In contrast to the default representatives evaluated in Section 6.3, the CE method cannot compensate for the absence of these categories, as shown in the test results. Training on the pruned dataset $\hat{\mathcal{X}}$ also appears to be effective for the Original Maps, as the agent policies trained with *Original Maps (Core)* progress similarly and generalize slightly better, except in the **Warehouse** map.

6.5 Comparison with Traditional MAPF Solvers

Setting Finally, we compare our VAULT and CoreVAULT policies with the traditional MAPF solvers CBSH, MAPF-LNS, and LaCAM*. All traditional MAPF solvers are run on all 25 test instances per map with a time budget of 60 seconds.

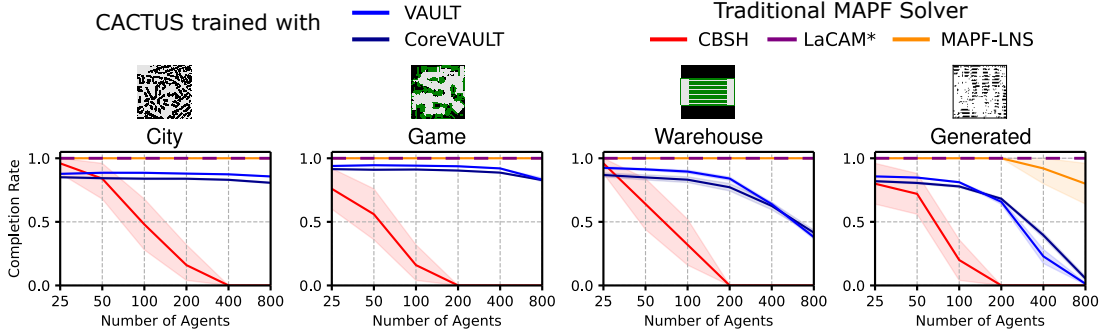


Figure 23: Test completion rate of CACTUS trained with the VAULT or CoreVAULT and the traditional MAPF solvers CBSH, MAPF-LNS, and LaCAM* over **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) test maps w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

Results The test results for **City**, **Game**, **Warehouse**, and **Generated** (i.e., created by the VAULT-B) maps are shown in Figure 23. LaCAM* consistently achieves a completion rate of 100% in all test maps. MAPF-LNS also achieves a completion rate of 100% except in **Generated** maps when the number of agents exceeds $N = 200$. Our VAULT and CoreVAULT policies consistently achieve completion rates of about 90% in **City** and **Game** maps but their performance degrades in the smaller **Warehouse** and **Generated** maps with an increasing number of agents. CBSH scales the worst in all test maps.

In Table 3, we provide the average travel times and iteration/trial counts for the test maps with the largest number of agents, where our VAULT or CoreVAULT policies were able to solve at least 80% of the test instances with a completion rate of 100%. The iteration/trial counts in Table 3 indicate the computational effort required per instance and serve as an estimate of the potential replanning cost that an environmental change would incur, e.g., the required communication rounds for distributing new plans. CBSH is not included due to its insufficient completion rates. In the larger test maps, namely **City** and **Game**, our VAULT or CoreVAULT policies are more effective than MAPF-LNS and LaCAM* by achieving a lower average travel time in most cases. However, MAPF-LNS and LaCAM* outperform our policies in the smaller test maps, i.e., **Warehouse** and **Generated**. MAPF-LNS consistently achieves shorter travel times than LaCAM* but requires at least thousands of iterations. LaCAM* requires hundreds of iterations until convergence, while our VAULT and CoreVAULT policies only require a single trial per test instance without global iterative refinement or branching like heuristic search (Sartoretti et al., 2019).

Discussion The results indicate that our decentralized VAULT and CoreVAULT policies can compete with traditional MAPF solvers in large maps with many agents, achieving shorter travel times within a single trial per test instance being only trained with a one-time effort of 40,000 training episodes (and the training of the VAULT). A few instances could not be solved completely because some scattered goal-reaching agents occasionally block other agents, thus preventing a perfect completion rate of 1. In contrast, traditional MAPF solvers must be run from scratch for every instance. According to the iteration/trial counts, MAPF-LNS is the most expensive anytime solver for dynamically changing environments. Even

Table 3: Average travel time and iteration/trial count of our VAULT and CoreVAULT policies, MAPF-LNS, and LaCAM* with the 95% confidence interval for the test maps of Figure 23. The best two (i.e., the smallest) values are highlighted in boldface. The iteration/trial counts serve as an estimate of the potential replanning cost that an environmental change would incur, e.g., the required communication rounds for distributing new plans.

	VAULT Policies	CoreVAULT Policies	MAPF-LNS	LaCAM*
Average Travel Time				
City ($N = 800$)	109.6 ± 1.0	105.1 ± 1.4	192.2 ± 5.9	230.4 ± 7.8
Game ($N = 400$)	95.2 ± 1.3	95.9 ± 1.5	173.8 ± 8.4	205.7 ± 11.1
Warehouse ($N = 200$)	117.2 ± 1.6	121.1 ± 4.4	88.6 ± 4.1	110.2 ± 7.6
Generated ($N = 100$)	128.9 ± 4.2	127.9 ± 2.0	43.6 ± 4.8	55.8 ± 18.5
Average Iteration/Trial Count				
City ($N = 800$)	1	1	$3,786 \pm 569$	519.0 ± 44.6
Game ($N = 400$)	1	1	$3,887 \pm 585$	400.8 ± 31.0
Warehouse ($N = 200$)	1	1	$21,328 \pm 4,168$	212.1 ± 22.1
Generated ($N = 100$)	1	1	$64,414 \pm 24,578$	106.5 ± 30.8

though LaCAM* needs considerably fewer iterations, its centralized design is prohibitive for large-scale systems, where sufficient communication bandwidth and node availability measures are needed, especially for replanning (Oliehoek & Amato, 2016; Tanenbaum & Van Steen, 2007). The performance drop of MAPF-LNS in the **Generated** map indicates that the VAULT-B is capable of generating challenging maps even for traditional state-of-the-art MAPF solvers. According to Appendix B, MAPF-LNS tends to fail in maps that resemble **Room** maps, which are known to be challenging (Li et al., 2022; Ren et al., 2024) while easy to generate, according to Section 4.1.2 and Figures 6, 7, and 20.

7. Conclusion and Future Work

We presented a generative curriculum approach to learning-based MAPF using the VAULT. Therefore, we introduced a two-stage framework to (I) train the VAULT and (II) use it to generate curricula in order to improve sample efficiency and generalization of learning-based MAPF methods. We proposed a bi-level curriculum scheme by combining the VAULT and CACTUS to improve sample efficiency further.

Our framework is designed in a modular and general way, where each proposed component serves its purpose in a black-box manner without considering specific micro-aspects of the underlying problem. For example, the VAULT is trained via unsupervised learning without any explicit labels at Stage I, data pruning for training the CoreVAULT is solely based on the reconstruction loss, and the optimization algorithm at Stage II searches latent spaces via Gaussian distributions without further information about the underlying maps or the generative model. Therefore, we can keep our methods simple without extensive engineering, which eases usability and reproducibility, and supports more cost-efficient and sustainable research in the future.

Our results demonstrate the potential of the VAULT as a powerful map generator for learning-based MAPF despite not being incentivized to learn useful representations via ex-

plicit labels or MAPF objectives. In contrast to RL training on the original maps, the VAULT can smooth the map space with its latent representations that enable flexible morphing of map features, which is beneficial for generalization beyond known maps. Optimizing over the latent space of the VAULT is easier than over the vertex space due to its low dimensionality and smoothness, thus improving sample efficiency and generalization over prior map generation approaches like MAP-Elites.

Our VAULT policies generalize better than other approaches regarding previously unseen maps, such as benchmark maps, novel artificial maps generated with another VAULT model, as well as different numbers of agents and map sizes. Compared to training regimes with other map generators, our VAULT policies consistently achieve higher completion rates and average travel times in the test instances.

The generalization capabilities can be further improved by adequately pruning the map dataset to train the CoreVAULT. Despite being trained on fewer maps than the original VAULT, the CoreVAULT models can still generate effective curricula to train agent policies that generalize similarly or even better than policies trained with the original VAULT. Our insight on data pruning can be directly transferred to the original maps, whose pruned counterpart does not affect the generalization of the resulting agent policies either, thus being potentially useful for future research on alternative map embeddings.

Using a suitable optimization algorithm in the VAULT curriculum can substantially improve the sample efficiency of RL training and even compensate for insufficiently trained VAULT models to some extent, e.g., when excluding the default cluster for the CoreVAULT.

Based on the promising results of our approach, there are many potential directions that we want to pursue in the future, as summarized below.

Generalization of the VAULT Training In this work, we focused on fixed-sized 2D grid worlds. While our VAE can be easily extended to variable-sized maps by removing the fully connected layers in the encoder and decoder, the VAULT curriculum will require a more advanced optimization algorithm to search latent spaces of variable dimensionality. Our VAE can be extended to encode whole instances by employing more advanced neural networks, such as transformers (Kaduri et al., 2020; Ren et al., 2021). Furthermore, we can extend the VAULT to general graphs by replacing the convolutional and deconvolutional layers with graph neural networks (Kipf & Welling, 2016; Li et al., 2020; Scarselli et al., 2008; Velićković et al., 2018).

Extensions to Training the VAULT To further reduce bias from the original MAPF benchmark maps listed in Table 2, we can retrain the VAULT on data generated by itself. This can be implemented as a fixed-point iteration by tracking the reconstruction loss of samples from the previous VAE model and the reconstructions of the current VAE model. Based on the insights on the empirical hardness of maps for traditional MAPF, we can train VAEs that additionally condition on a difficulty variable, such as the map connectivity, i.e., the second smallest eigenvalue of the normalized Laplacian matrix of a map (Ren et al., 2024). This would add one additional dimension to our (latent) search space and can regularize our curriculum by increasing or decreasing difficulty when the median completion rate exceeds or falls below a certain threshold. Another interesting direction is the generation of *guidance maps* by creating real-valued matrices instead of binary ones, which can direct

the search for traditional MAPF solvers (Chen et al., 2024; Cohen & Koenig, 2016; Zhang et al., 2024).

Alternative Generative Models In recent years, diffusion models have become popular due to their ability to generate higher-quality data than VAEs (Dhariwal & Nichol, 2021; Xiao et al., 2022). Common diffusion models operate on the latent space of a pre-trained VAE (Rombach et al., 2022), which can be directly applied to our VAULT generator. However, diffusion models employ complex architectures which are computationally demanding for training and inference (Rombach et al., 2022; Wang et al., 2023b) and, thus, would slow down our VAULT curriculum significantly. Diffusion models typically require millions of data samples to capture the underlying data distribution (Wang et al., 2023b), which could be useful for non-binary maps, e.g., with special locations for dispatching tasks, storage, or safety-critical operations. Other alternatives are generative adversarial networks (Goodfellow et al., 2014) and inverse retrieval techniques (Zheng et al., 2025), which could improve the data quality or interpretability of the map optimization procedure, respectively.

Alternatives to Data Pruning Data pruning for training the VAULT can be conducted in a more fine-grained way by discarding redundant maps within map categories. For example, the (sub)maps extracted from the warehouse maps always exhibited strong similarities, indicating high redundancy within this specific map category. *Data distillation* offers an alternative data reduction approach, where the dataset is compressed into a smaller set of artificial data samples (Sachdeva & McAuley, 2023).

Extensions to the VAULT Curriculum In addition to our simple optimization algorithms proposed in Section 4.2.1, we can employ more advanced methods, such as CMA-ES, CMA-MAE, etc. (Beyer & Schwefel, 2002; Fontaine & Nikolaidis, 2023; Fontaine et al., 2020; Hansen, 2016; Schwefel, 1993), to improve sample efficiency further. To improve generalization, robustness, and testing, we can employ adversarial search techniques to challenge our agent policies more (Gabor et al., 2019; Phan et al., 2020, 2021; Wang et al., 2019). Adversarial search for robustness represents an orthogonal approach to the empirical hardness regularization suggested above (Ren et al., 2024).

Acknowledgments

The research at the University of Southern California and University of California, Irvine 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 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.

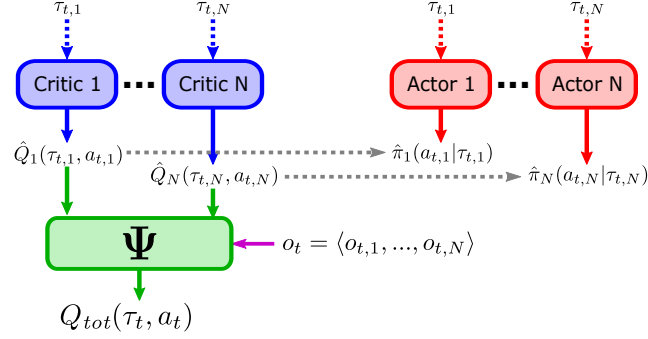


Figure 24: Common actor-critic scheme as used in CACTUS (Phan et al., 2024b) and other works on cooperative multi-agent RL (Peng et al., 2021; Phan et al., 2021; Su et al., 2021).

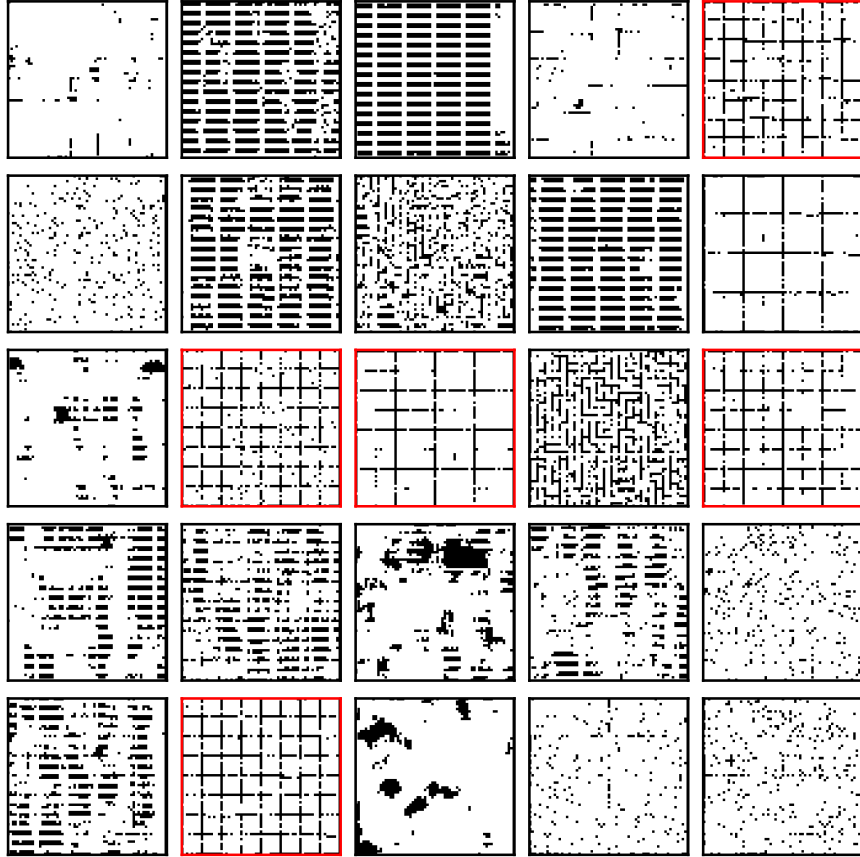


Figure 25: The 25 test maps generated with the VAULT-B, as described in Section 5.1. Maps, where MAPF-LNS could not find any solution (Figure 23) are highlighted in red.

Appendix A. Experimental Details

A.1. Neural Network Architectures

A.1.1. VARIATIONAL AUTOENCODERS FOR THE VAULT AND COREVAULT

We use deep convolutional and deconvolutional networks to implement *enc* and *dec*.

The input layer of the encoder *enc* processes a map G as a binary 64×64 matrix. The first hidden layer convolves 32 filters of kernel size 3×3 with a convolutional layer of 32 filters with stride 1 with the input matrix and applies a ReLU activation. The result is processed by five subsequent convolutional layers convolving 32 filters of kernel size 3×3 with alternatingly stride 1 and 2 with the previous hidden result followed by a ReLU activation. Afterward, the result is flattened and processed by two separate fully connected linear layers of $c = 128$ units, representing the mean $\mu(G)$ and the variance $\sigma^2(G)$ of the input map G . $\mu(G)$ and $\sigma^2(G)$ represent the parameters ϕ of the latent distribution $q_\phi(z|x)$, i.e., a Gaussian, and are used to sample a c -dimensional latent vector $z \sim q_\phi(\cdot|G)$.

The decoder processes the c -dimensional latent vector sample z with a fully connected linear layer of $c = 128$ units, followed by a sequence of deconvolutional layers that process the latent vector z analogously to the encoder layers (albeit in reverse order) to restore the original map size of 64×64 . The output of the decoder is processed by a sigmoid activation.

A.1.2. POLICY AND VALUE NETWORKS

We use *multilayer perceptrons (MLP)* to implement $\hat{\pi}_i$ and $\hat{Q}_i$ for each agent i and the factorization operator Ψ of QMIX, which are depicted in Figure 24.

Since all regarded maps are 2D grid worlds, the observations are encoded as multi-channel image, as illustrated in Figure 14. We flatten the multi-channel images before feeding them into the MLPs. $\hat{\pi}_i$ 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 have two hidden layers of 128 units with ELU activation and one or $|\mathcal{A}_i|$ linear output units, respectively. For PRIMAL, PRIMAL2, and CostTracer, we use the same architecture for a fair and computationally efficient comparison w.r.t. data, compute, and model size. The policy and value network architectures are identical to the MLPs used in (Phan et al., 2024b) with less than 5% of the model size originally used for PRIMAL and PRIMAL2 (Damani et al., 2021; Sartoretti et al., 2019).

A.2. Training Details and Hyperparameters

A.2.1. TRAINING THE VAULT

All VAEs used in this work are trained for 12,000 iterations via Algorithm 1, using ADAM with a learning rate of 0.001, and a regularization factor of $\lambda = 0.001$. All hyperparameters for training the VAULT and CoreVAULT, and the CE method for our VAULT curriculum, are listed in Table 4.

A.2.2. MAP GENERATION

All map generators used to train CACTUS for the comparison in Section 6.2 are described in this Section. To ensure connectivity of all generated maps, we iteratively connect random

Table 4: Hyperparameters (HP) and their respective final values used for training the VAULT and CoreVAULT, and the CE method for our VAULT curriculum. We also list the numbers that have been tried during the development of our paper.

HP	Final Value	Numbers/Range	Description
d	$64 \times 64 = 4096$	$\{32 \times 32, 64 \times 64, 128 \times 128\}$	Input dimension and map size. $d = 64 \times 64$ offered the best tradeoff between map diversity and computational demand.
c	128	$\{64, 128, 256\}$	Latent space dimension. $c = 128$ offered the best tradeoff between reconstruction capabilities and computational demand.
λ	0.001	$\{0.1, 0.01, 0.001, 0.0001\}$	Regularization factor for the KL term in the VAE loss of Equation 7.
α_{VAE}	0.001	$\{0.001\}$	Learning rate. We used the default value of ADAM in <code>torch</code> without further tuning.
δ_{rec}	0.02	$\{0.02\}$	Loss threshold for dataset pruning. We used the average reconstruction loss of the original VAULT after 12,000 iterations of training.
μ_0	0	$\{0\}$	Initial centroid for the Gaussian distribution used in our CE method.
$\sigma_{max,0}^2$	10^3	$\{10^2, 10^3, 10^4\}$	Initial variance for the CE method. The higher the value the higher the potential diversity of generated maps, which may require more time to find promising regions in the latent space.
B	25	$\{10, 25, 50\}$	Population size used for the CE method. B cannot be greater than E_{epoch} (Table 5).

pairs of disconnected regions via deletion of obstacles along random connecting paths until no isolated region is left. The randomized connections avoid artificially straight corridors between the regions thus reducing domain-specific bias, e.g., toward warehouses.

VAULT Maps are randomly sampled from the latent space of the VAULT or CoreVAULT. We use a spherical Gaussian distribution $\mathcal{N}(\mu_0, \Sigma_0)$ with a 128-dimensional zero vector as the centroid μ_0 and a diagonal matrix $\Sigma_0 = 1000 \cdot \mathbf{1} \in \mathbb{R}^{128 \times 128}$ as the covariance, which serves as an initial distribution for the CE method in our VAULT curriculum. For random sampling, we use $\mathcal{N}(\mu_0, \Sigma_0)$ without further changes. Examples of random training maps are shown in Figure 7 and examples of random test maps generated with the VAULT-B are shown in Figure 25. Test maps, where MAPF-LNS could not find any solution, according to Figure 23, are highlighted in red.

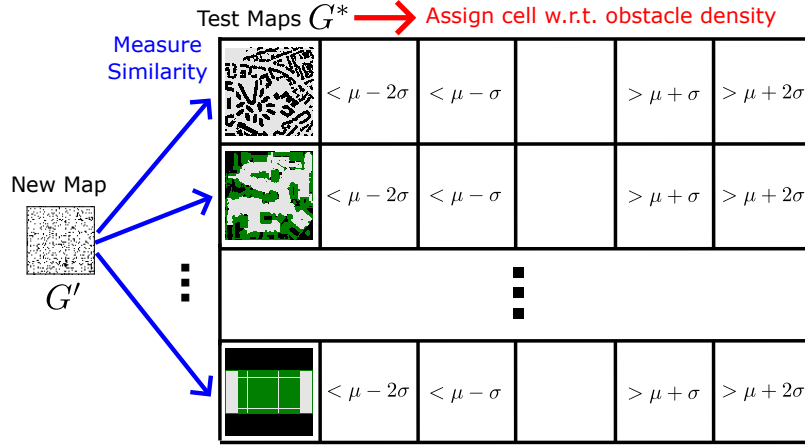


Figure 26: Schematic illustration of our MAP-Elites archive structure. A new map G' is first matched with a test map $G^* \in \mathcal{X}^*$, listed in Table 2, to select a row. G' is then assigned to a column w.r.t. its obstacle density relative to the obstacle density distribution of the best-matching test map, e.g., if G' has an obstacle density of 0.5 and the test map has an average obstacle density of $\mu = 0.3$ with a standard deviation of $\sigma = 0.1$ per cell, then G' is assigned to the 4th column because $\mu + \sigma < 0.5 \leq \mu + 2\sigma$.

MAP-Elites MAP-Elites optimizes a population of $B = 25$ maps using evolutionary operations, as proposed in (Zhang et al., 2023b). We employ an archive, which assigns a map G to an archive entry according to its similarity to the test maps $G^* \in \mathcal{X}^*$, listed in Table 2, and its obstacle density relative to the best-matching test map. The similarity is assessed by the minimum mean squared error between G and the test maps G^* . The obstacle density of G is classified, according to the density distribution of the best-matching test map using a deviation interval σ , 2σ , and ∞ in both directions (Figure 26). At every epoch, a map G is sampled from the archive and modified by randomly changing k matrix entries, where k is determined by a geometric distribution $\mathbb{P}(k) = (1 - p)^{k-1}p$ with $p = \frac{1}{2}$. Similar to our VAULT curriculum, formulated in Algorithm 3, MAP-Elites is updated after each training cycle of $E_{epoch} = 50$ epochs using the median completion rate ρ^* .

PRIMAL Maps PRIMAL maps are generated with random obstacles according to a density ratio sampled from a triangular distribution between 0 and 0.5 with a peak at 0.33 (Sartoretti et al., 2019). As proposed in (Sartoretti et al., 2019), the map size is randomly selected from a set $\{10 \times 10, 40 \times 40, 70 \times 70\}$ with map size 10×10 being chosen with a chance of 50% while the other map sizes are chosen with a chance of 25%.

Original Maps The Original Maps are sampled uniformly from all training maps $G \in \mathcal{X}$ used for training the VAULT. The creation of the map dataset is described in Section 4.1.1.

Test Instances For test maps of size 64×64 or smaller, as listed in Table 2, we use the predefined random scenarios of the public MAPF benchmark set (Stern et al., 2019). For the larger maps, we ran LaCAM* on the predefined random scenarios and moved the goals closer to the start locations such that the minimum makespan is 150 at most to ensure feasibility for all learning-based MAPF approaches with a horizon of $H = 256$. For the test

maps generated with the VAULT-B, we allocated the start and goal locations randomly and verified the feasibility with LaCAM*.

Table 5: Hyperparameters (HP) and their respective final values used by CACTUS and all other RL algorithms evaluated in the paper. We also list the numbers that have been tried during the development of our paper.

HP	Final Value	Numbers/Range	Description
H	256	{256}	Maximum number of time steps per episode.
$E_{episode}$	10	{5, 10, 20}	Number of episodes per epoch or batch.
E_{epoch}	50	{50}	Number of epochs per training cycle before a new population of B training maps is sampled (Table 4).
Total epochs	4000	{1000, 2000, 4000, 8000}	Number of epochs. We gradually increased it to assess the stability of the learning progress until convergence.
$DIST$	Manhattan	{Chebyshev, Euclidean, Manhattan}	The distance function to determine the goal allocation area of CACTUS, according to Section 3.4.2 and Figure 4.
$\delta_{threshold}$	0.75	{0.75}	Curriculum threshold of CACTUS. Recommended default value.
η_{conf}	2	{2}	Deviation factor for the confidence assessment. Recommended default value.
α_{RL}	0.001	{0.001}	Learning rate. We used the default value of ADAM in <code>torch</code> without further tuning.
Clip norm	1	{1, ∞ }	Gradient clipping parameter. Using a clip norm of 1 leads to better performance than disabling it with ∞ .
$ \tau_{t,i} $	1	{1, 5, 10}	Local history length. It was set to 1 to reduce computation because the other values did not significantly improve performance.

A.2.3. REINFORCEMENT LEARNING (RL) TRAINING

The policy and value networks are updated after every epoch, i.e., $E_{episode} = 10$ episodes of RL training, using ADAM with a learning rate of 0.001.

For each experiment, all learning-based MAPF approaches are run 20 times over 4,000 epochs. A training cycle consists of $E_{epoch} = 50$ epochs with $E_{episode} = 10$ episodes each. Thus, each approach is trained with $40,000 = 4,000 \cdot E_{episode}$ training episodes in total. PRIMAL2 is additionally pre-trained with 20,000 randomly generated corridor observa-

Algorithm 4 CACTUS Algorithm from (Phan et al., 2024b, 2024a)

```

1: procedure CACTUS( $\mathcal{G}, DIST, \delta_{threshold}, \eta_{conf}$ )
2:   Initialize parameters of  $\hat{\pi}_i, \hat{Q}_i$  for each agent  $i \in \mathcal{D}$  and  $\Psi$ 
3:   Set  $R_{alloc} \leftarrow 1$  ▷ Initialize goal allocation radius
4:   for epoch  $e$  in  $1, \dots, 4000$  do ▷ Total epochs according to Table 5
5:     for episode  $b$  in  $1, \dots, E_{episode}$  do
6:       Sample a map  $G'$  from  $\mathcal{G}$  and start locations  $v_{start,i}$  for all agents  $i \in \mathcal{D}$ 
7:       for agent  $i \in \mathcal{D}$  do ▷ Instance generation based on map  $G'$ , Figure 13
8:         Set  $\mathcal{V}_{goal,i} \leftarrow \{v \in \mathcal{V} | DIST(v, v_{start,i}) \leq R_{alloc}\}$  ▷ Allocation area, Figure 4
9:         Randomly select a goal location  $v_{goal,i}$  from  $\mathcal{V}_{goal,i}$ 
10:      for time step  $t$  in  $0, \dots, H - 1$  do
11:         $a_t \sim \hat{\pi}(\cdot | \tau_t)$  ▷ Sample a joint action with the agent policies
12:        Execute joint action  $a_t$  and record experience sample
13:         $\rho_H^b \leftarrow \frac{|\{i \in \mathcal{D} | v_{H,i} = v_{goal,i}\}|}{N}$  ▷ Completion rate of episode  $b$ 
14:      Update  $\Psi$  and  $\hat{\pi}_i, \hat{Q}_i$  for each agent  $i \in \mathcal{D}$  with all experience samples of epoch  $e$ 
15:      Calculate the average  $\mu_\rho$  and standard deviation  $\sigma_\rho$  of all completion rates  $\rho_H^b$ 
16:      if  $\mu_\rho - \eta_{conf}\sigma_\rho \geq \delta_{threshold}$  then ▷ Confidence-based radius update
17:         $R_{alloc} \leftarrow R_{alloc} + 1$ 
18:  return  $\hat{\pi} = \langle \hat{\pi}_1, \dots, \hat{\pi}_N \rangle$ 
    
```

tions for convention learning before the first RL training epoch. The hyperparameters for CACTUS and all other RL algorithms evaluated in the paper are listed in Table 5.

A.2.4. CACTUS ALGORITHM

A formulation of CACTUS from (Phan et al., 2024b) is provided in Algorithm 4. $\mathcal{G}$ is a set of training maps or a map generator, $DIST: \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}$ is a vertex distance function, $\delta_{threshold}$ is the curriculum decision threshold, and η_{conf} is the deviation factor.

Appendix B. Additional Results

We report additional results regarding the test maps listed in Table 2 and performance measures, i.e., the completion rate and average travel time, as defined in Sections 2.1 and 6. The results further support our claims and discussions in Section 6.

B.1. Comparison of Different Reinforcement Learning Methods

The results for the **Empty**, **Game**, **Room**, and **Maze** maps are shown in Figures 27 and 28. All methods progress faster w.r.t. completion rate and average travel time when trained with the VAULT. CACTUS progresses fastest in all cases achieving the highest completion rate and the lowest average travel time. CostTracer progresses second fastest. PRIMAL makes slight progress and PRIMAL2 only improves visibly when trained with the VAULT maps.

Due to the consistently poor performance of all approaches in the **Maze** map, a common RL problem known from (Dennis et al., 2020), we omit it in the following results.

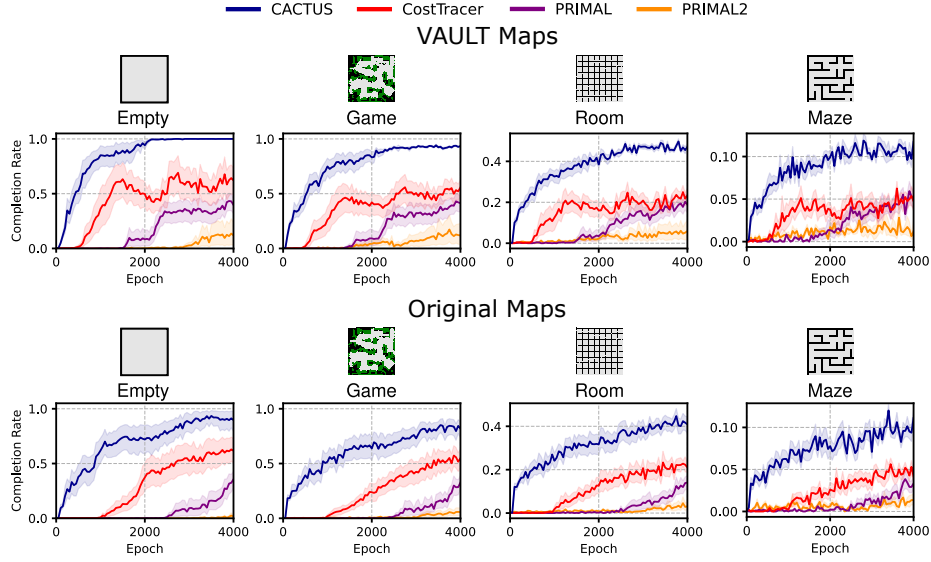


Figure 27: Completion rate progress of CACTUS, CostTracer, PRIMAL, and PRIMAL2 w.r.t. the training epochs. The training maps are randomly generated with the VAULT (**top row**) or randomly chosen from the Original Maps (**bottom row**), according to Table 2. Before the first RL training epoch 0, PRIMAL2 is pre-trained on 20,000 handcrafted corridor observations. Shaded areas show the 95% confidence interval.

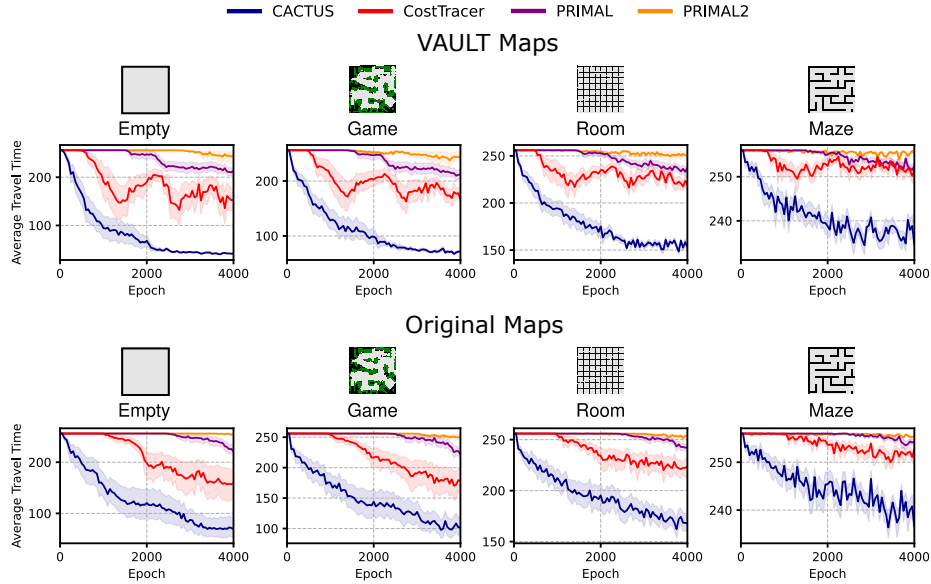


Figure 28: Average travel time progress of CACTUS, CostTracer, PRIMAL, and PRIMAL2 w.r.t. the training epochs. The training maps are randomly generated with the VAULT (**top row**) or randomly chosen from the Original Maps (**bottom row**), according to Table 2. Before the first RL training epoch 0, PRIMAL2 is pre-trained on 20,000 handcrafted corridor observations. Shaded areas show the 95% confidence interval.

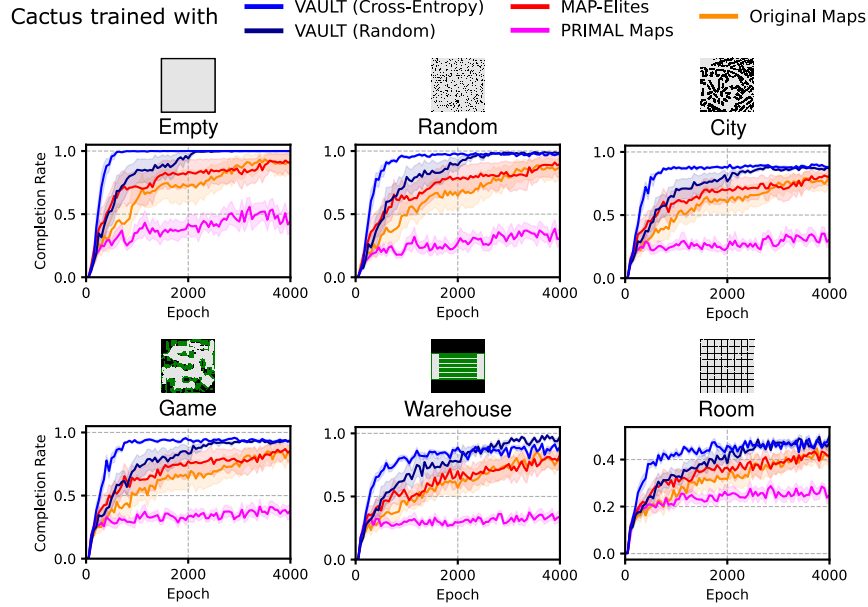


Figure 29: Completion rate progress of CACTUS trained with the VAULT, MAP-Elites, PRIMAL maps, or the Original Maps w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

B.2. Curriculum Learning with the VAULT

The training results w.r.t. completion rate and average travel time are shown in Figures 29 and 30. Our VAULT policies progress fastest when using the CE method. Our VAULT policies using random sampling progress second fastest. MAP-Elites policies progress slightly faster than Original Map policies. PRIMAL Map policies progress the worst, never achieving a completion rate of over 50% except in the **Empty** map.

The test results w.r.t. completion rate and average travel time, are shown in Figures 31 and 32. Our VAULT policies perform best, while MAP-Elites policies slightly outperform Original Map policies in settings with a smaller number of agents, e.g., $N \leq 50$. PRIMAL Map policies generalize worst in all test maps. In the **Generated** maps, the completion rate of all policies deteriorates fastest with an increasing number of agents.

B.3. Curriculum Learning with the CoreVAULT

The training results of the CoreVAULT policies w.r.t. completion rate and average travel time, are shown in Figures 33 and 34. Training with the original VAULT progresses fastest, while all CoreVAULT policies with a default representative progress second fastest without significant differences between each other. Omitting the default representative affects the sample efficiency notably, thus requiring more epochs to keep up with the other CoreVAULT policies eventually.

The test results w.r.t. completion rate and average travel time, are shown in Figures 35 and 36. After 40,000 episodes of RL training, all policies achieve similar performance, regardless of which VAULT or CoreVAULT model was used for RL training.

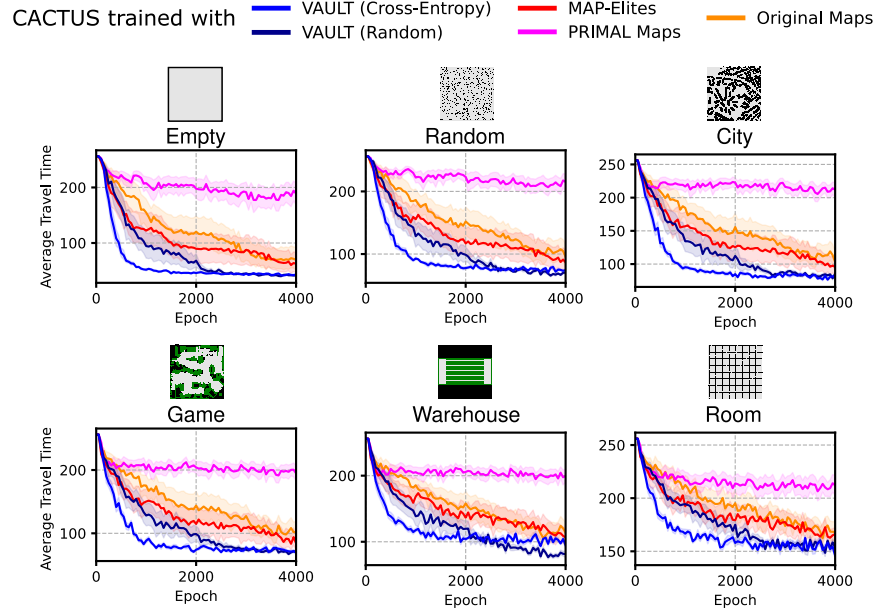


Figure 30: Average travel time progress of CACTUS trained with the VAULT, MAP-Elites, PRIMAL maps, or the Original Maps w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

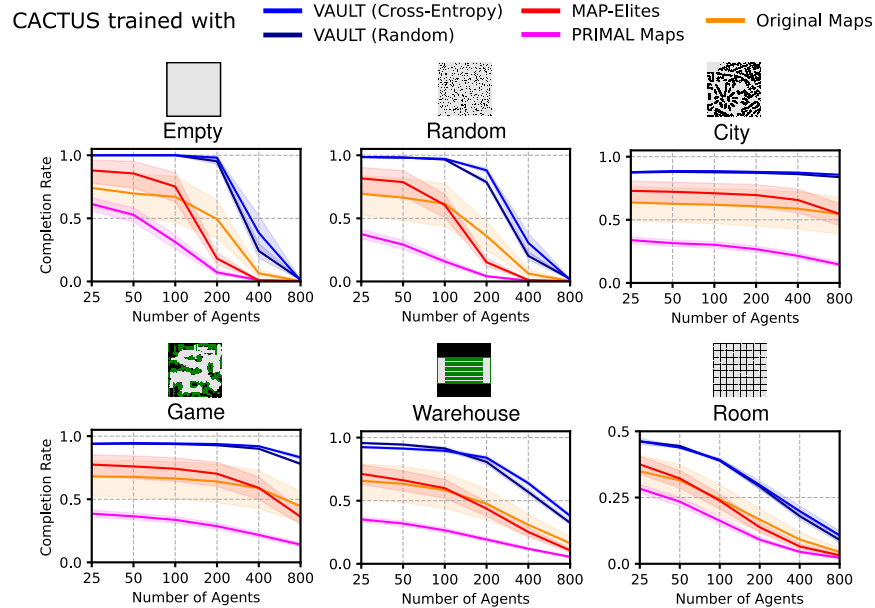


Figure 31: Test completion rate of CACTUS trained with the VAULT, MAP-Elites, PRIMAL maps, or the Original Maps w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

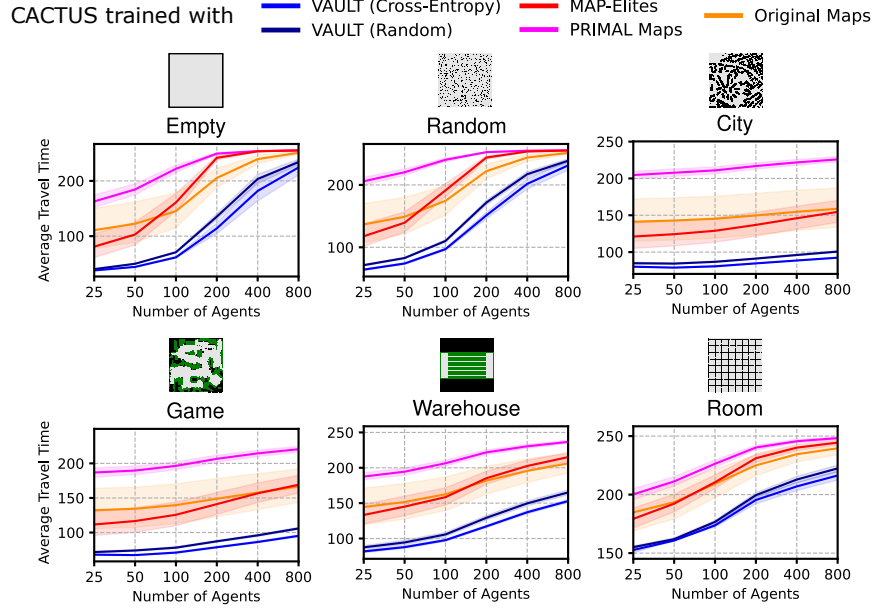


Figure 32: Test average travel time of CACTUS trained with the VAULT, MAP-Elites, PRIMAL maps, or the Original Maps w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

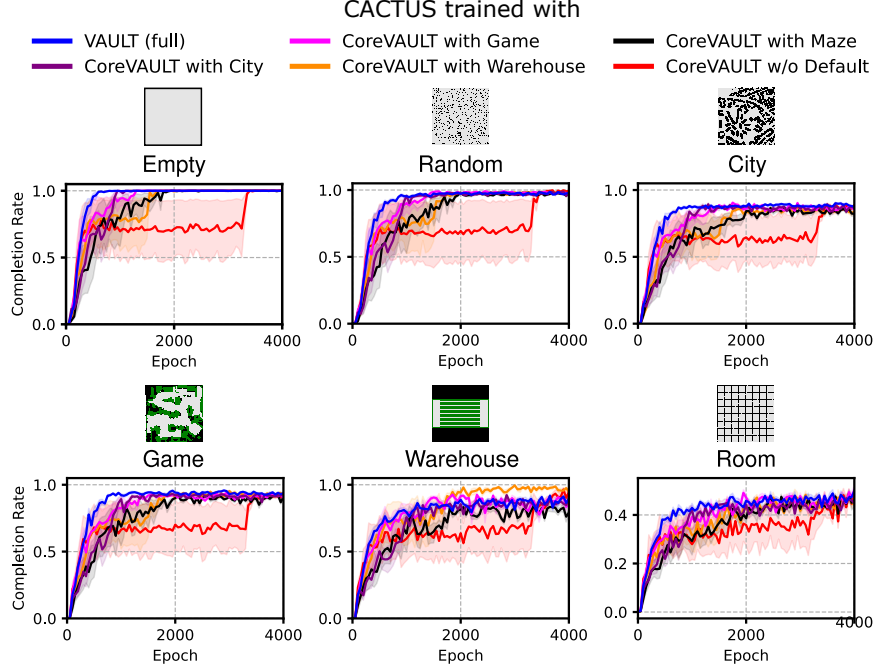


Figure 33: Completion rate progress of CACTUS trained with the VAULT and different variants of the CoreVAULT w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

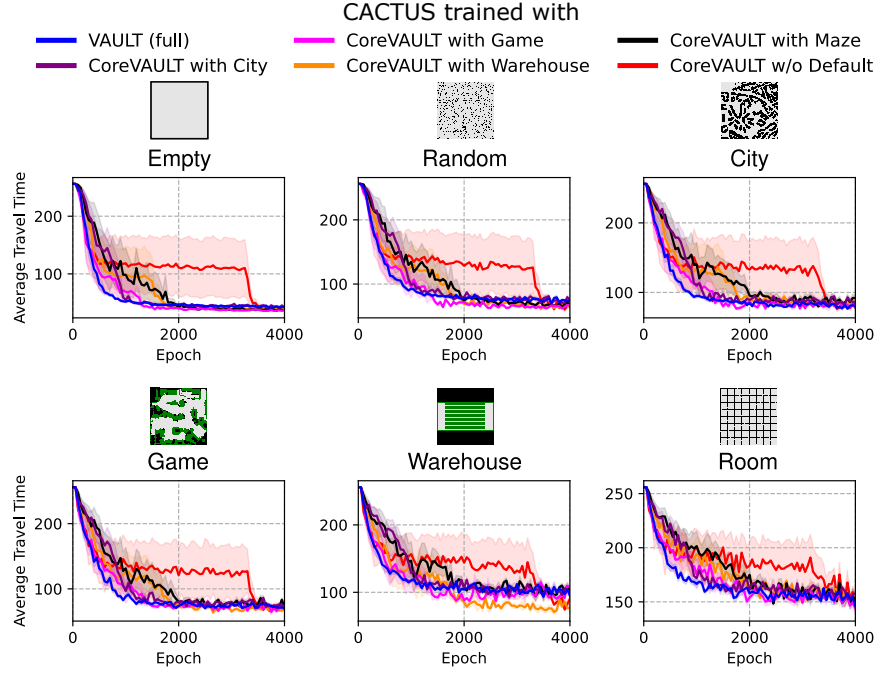


Figure 34: Average travel time progress of CACTUS trained with the VAULT and different variants of the CoreVAULT w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

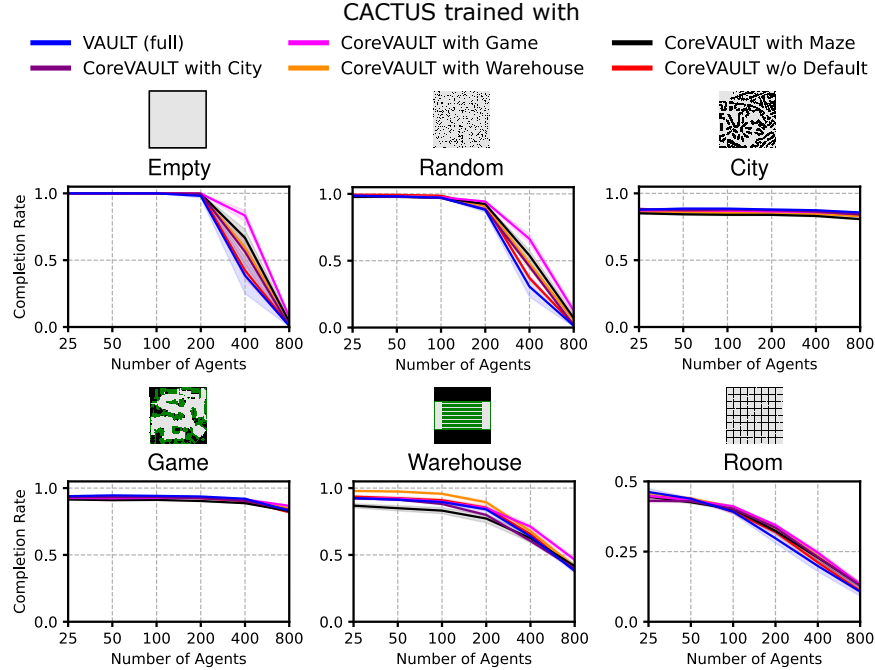


Figure 35: Test completion rate of CACTUS trained with the VAULT and different variants of the CoreVAULT w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

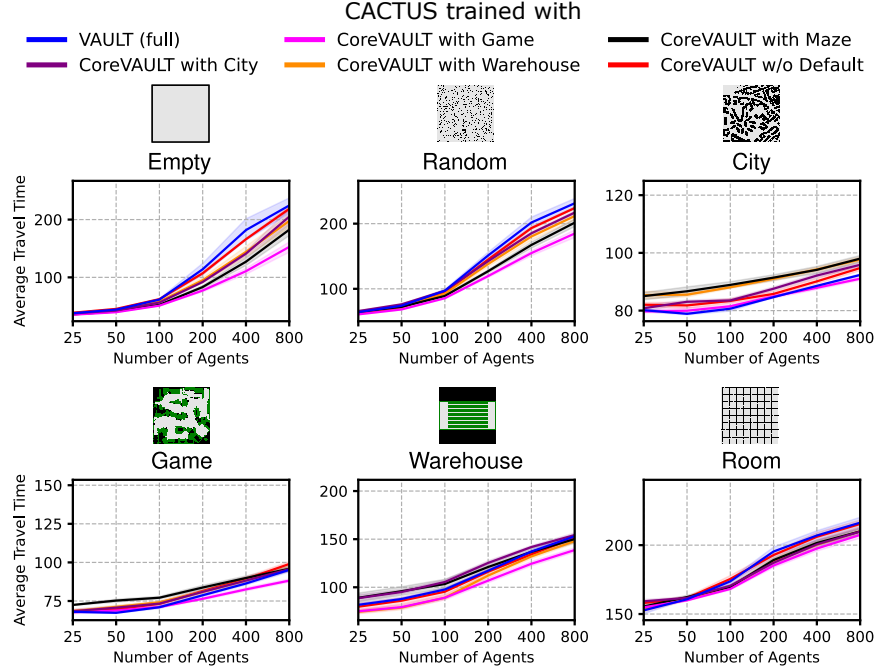


Figure 36: Test average travel time of CACTUS trained with the VAULT and different variants of the CoreVAULT w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

B.4. Curriculum Learning with VAULT Ablations

The training results w.r.t. completion rate and average travel time of the VAULT and its ablations without **Random** and **Room** maps, respectively, and the CoreVAULT, are shown in Figures 37 and 38. Training with the original VAULT progresses fastest, while our CoreVAULT policies progress second fastest. Omitting **Random** or **Room** maps affects the sample efficiency notably, thus requiring more epochs to eventually keep up with the original VAULT and CoreVAULT policies. The Original Map (Core) policies progress similarly to the Original Map policies without any degradation in sample efficiency.

The test results w.r.t. completion rate and average travel time of the VAULT and its ablations without **Random** and **Room** maps, respectively, and the CoreVAULT, are shown in Figures 39 and 40. After 40,000 episodes of RL training, the agent policies trained with the VAULT ablations underperform the original VAULT and CoreVAULT policies with an increasing number of agents, while generally outperforming the Original Map policies. Omitting **Room** maps leads to higher sensitivity toward **Empty**, **City**, and **Game**.

B.5. Comparison with Traditional MAPF Solvers

The test results w.r.t. completion rate are shown in Figure 41. LaCAM* consistently achieves a completion rate of 100% in all test maps. MAPF-LNS also achieves a completion rate of 100% except in **Room** maps when the number of agents exceeds $N = 400$. Our VAULT and CoreVAULT policies consistently achieve completion rates of about 90% in

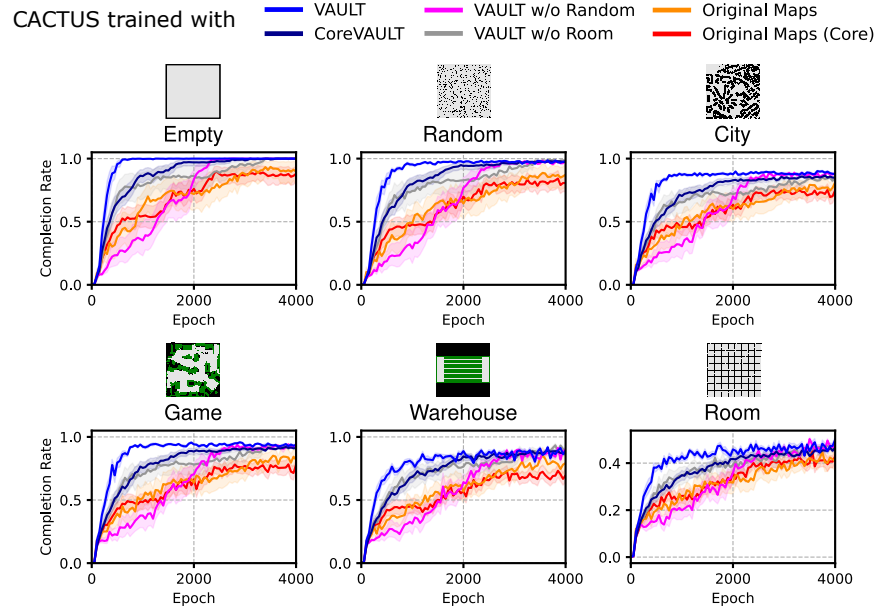


Figure 37: Completion rate progress of CACTUS trained with the VAULT and its ablations (without **Random** and **Room** maps, respectively), the CoreVAULT, or the Original Maps (pruned or not) w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

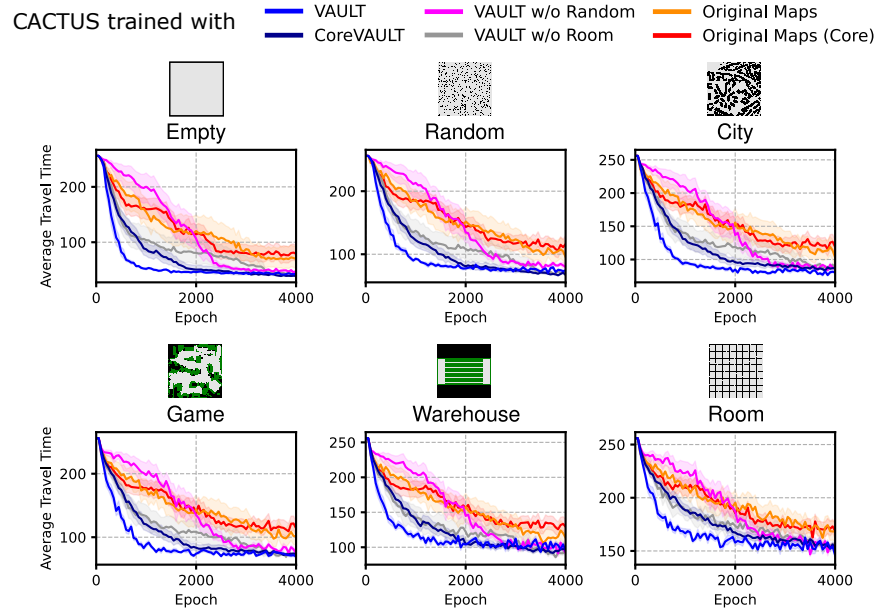


Figure 38: Average travel time progress of CACTUS trained with the VAULT and its ablations (without **Random** and **Room** maps, respectively), the CoreVAULT, or the Original Maps (pruned or not) w.r.t. the training epochs. Shaded areas show the 95% confidence interval.

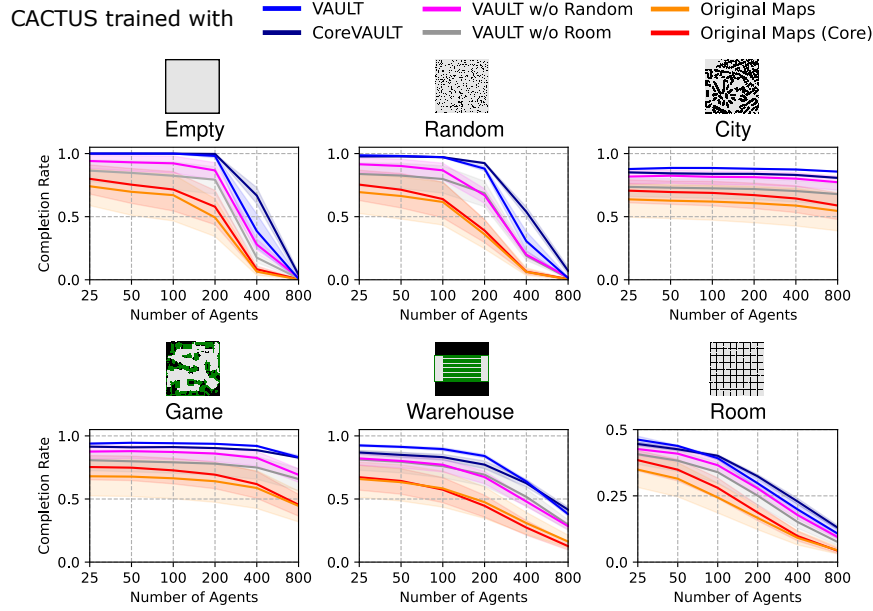


Figure 39: Test completion rate of CACTUS trained with the VAULT and its ablations (without Random and Room maps, respectively), CoreVAULT, or the Original Maps (pruned or not) w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

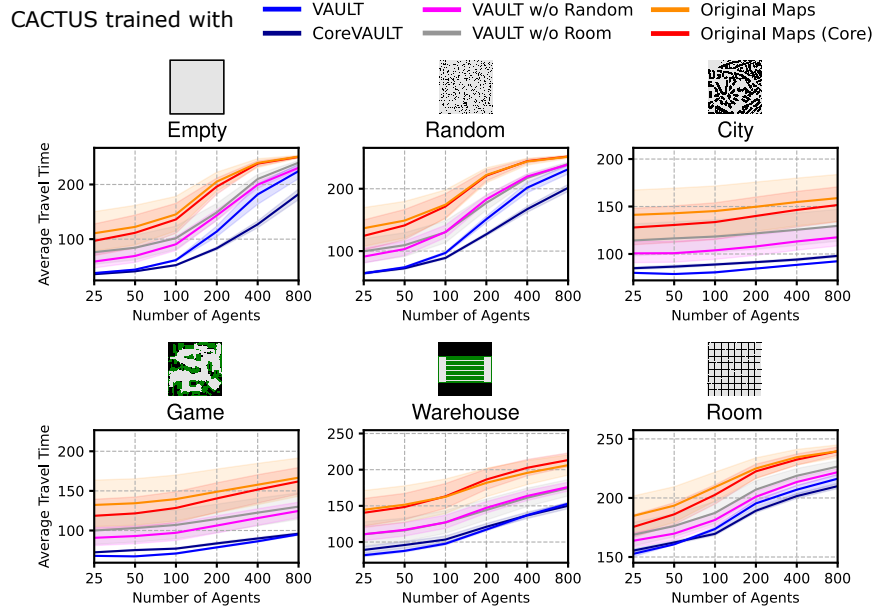


Figure 40: Average travel time of CACTUS trained with the VAULT and its ablations (without Random and Room maps, respectively), CoreVAULT, or the Original Maps (pruned or not) w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

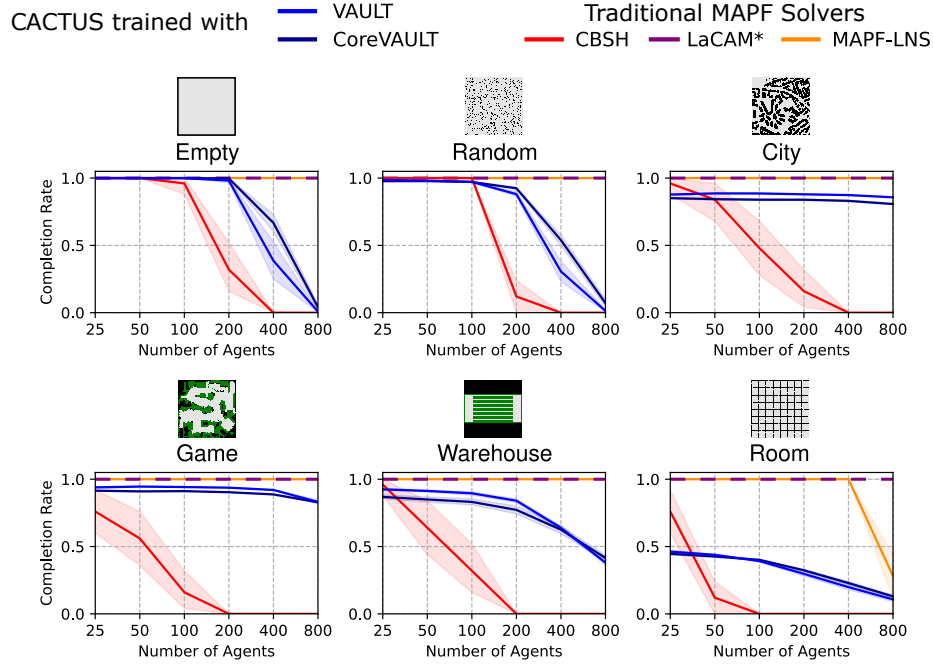


Figure 41: Test completion rate of CACTUS trained with the VAULT or CoreVAULT and the traditional MAPF solvers CBSH, MAPF-LNS, and LaCAM* w.r.t. the number of agents after 40,000 episodes of RL training. Shaded areas show the 95% confidence interval.

City and Game maps but their performance degrades in the smaller maps, namely Empty, Random, Warehouse, and Room, with an increasing number of agents. CBSH scales worst in all test maps.

References

- Alkazzi, J.-M., & Okumura, K. (2024). A Comprehensive Review on Leveraging Machine Learning for Multi-Agent Path Finding. *IEEE Access*, 12, 57390–57409.
- 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.
- Asai, M., Kajino, H., Fukunaga, A., & Muise, C. (2022). Classical Planning in Deep Latent Space. *Journal of Artificial Intelligence Research*, 74, 1599–1686.
- Bachem, O., Lucic, M., & Krause, A. (2017). Practical Coreset Constructions for Machine Learning. In *arXiv preprint arXiv:1703.06476*.
- Baldi, P. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. In Guyon, I., Dror, G., Lemaire, V., Taylor, G., & Silver, D. (Eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, Vol. 27 of *Proceedings of Machine Learning Research*, pp. 37–49, Bellevue, Washington, USA. PMLR.

- Baldi, P., & Sadowski, P. J. (2013). Understanding Dropout. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., & Weinberger, K. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 26. Curran Associates, Inc.
- Bank, D., Koenigstein, N., & Giryas, R. (2023). *Autoencoders*, pp. 353–374. Springer International Publishing, Cham.
- Belzner, L. (2016). *Simulation-Based Autonomous Systems in Discrete and Continuous Domains*. Ph.D. thesis, LMU Munich.
- Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum Learning. In *26th International Conference on Machine Learning*.
- Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution Strategies – A Comprehensive Introduction. *Natural Computing*, 1, 3–52.
- Bolland, A., Boukas, I., Berger, M., & Ernst, D. (2022). Jointly Learning Environments and Control Policies with Projected Stochastic Gradient Ascent. *Journal of Artificial Intelligence Research*, 73, 117–171.
- Chen, Z., Harabor, D., Li, J., & Stuckey, P. (2024). Traffic Flow Optimisation for Life-long Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 20674–20682.
- Cohen, L., & Koenig, S. (2016). Bounded Suboptimal Multi-Agent Path Finding using Highways. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 3978–3979.
- Dai, B., & Wipf, D. (2019). Diagnosing and Enhancing VAE Models. In *International Conference on Learning Representations*.
- Damani, M., Luo, Z., Wenzel, E., & Sartoretti, G. (2021). PRIMAL2: Pathfinding via Reinforcement and Imitation Multi-Agent Learning-Lifelong. *IEEE Robotics and Automation Letters*, 6(2), 2666–2673.
- Dennis, M., Jaques, N., Vinitzky, 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, 13049–13061.
- Devlin, S., & Kudenko, D. (2011). Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems. In *Tenth International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 225–232. ACM.
- Dhariwal, P., & Nichol, A. Q. (2021). Diffusion Models Beat GANs on Image Synthesis. In Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W. (Eds.), *Advances in Neural Information Processing Systems*.
- Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., & Clune, J. (2019). Go-Explore: A New Approach for Hard-Exploration Problems. In *arXiv preprint arXiv:1901.10995*.
- Emery-Montemerlo, R., Gordon, G., Schneider, J., & Thrun, S. (2004). Approximate Solutions for Partially Observable Stochastic Games with Common Payoffs. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pp. 136–143. IEEE.

- Esser, P., Kulal, S., Blattmann, A., Entezari, R., Müller, J., Saini, H., Levi, Y., Lorenz, D., Sauer, A., Boesel, F., et al. (2024). Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *Proceedings of the 41st International Conference on Machine Learning*.
- Esser, P., Rombach, R., & Ommer, B. (2021). Taming Transformers for High-Resolution Image Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883.
- Ewing, E., Ren, J., Kansara, D., Sathiyarayanan, V., & Ayanian, N. (2022). Betweenness Centrality in Multi-Agent Path Finding. In *International Conference on Autonomous Agents and Multiagent Systems*.
- Eysenbach, B., Gupta, A., Ibarz, J., & Levine, S. (2019). Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*.
- Eysenbach, B., & Levine, S. (2022). Maximum Entropy RL (Provably) Solves Some Robust RL Problems. In *International Conference on Learning Representations*.
- 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 *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 28, pp. 83–87.
- 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.
- Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).
- Fontaine, M., & Nikolaidis, S. (2023). Covariance Matrix Adaptation Map-Annealing. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 456–465.
- Fontaine, M. C., Togelius, J., Nikolaidis, S., & Hoover, A. K. (2020). Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 94–102.
- Gabor, T., Sedlmeier, A., Kiermeier, M., Phan, T., et al. (2019). Scenario Co-Evolution for Reinforcement Learning on a Grid World Smart Factory Domain. In *Genetic and Evolutionary Computation Conference*, p. 898–906.
- Gao, Z., & Prorok, A. (2023a). Constrained Environment Optimization for Prioritized Multi-Agent Navigation. *IEEE Open Journal of Control Systems*, 2, 337–355.
- Gao, Z., & Prorok, A. (2023b). Environment Optimization for Multi-Agent Navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3440–3446.
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Nets. In Ghahramani,

- Z., Welling, M., Cortes, C., Lawrence, N., & Weinberger, K. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc.
- Gronauer, S., & Diepold, K. (2022). Multi-Agent Deep Reinforcement Learning: A Survey. *Artificial Intelligence Review*, 55(2), 895–943.
- Ha, D., & Schmidhuber, J. (2018). Recurrent World Models Facilitate Policy Evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 31. Curran Associates, Inc.
- Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2020). Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*.
- Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning Latent Dynamics for Planning from Pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR.
- Hafner, D., Lillicrap, T. P., Norouzi, M., & Ba, J. (2021). Mastering Atari with Discrete World Models. In *International Conference on Learning Representations*.
- Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering Diverse Domains through World Models. In *arXiv preprint arXiv:2301.04104*.
- Hansen, E. A., Bernstein, D. S., & Zilberstein, S. (2004). Dynamic Programming for Partially Observable Stochastic Games. In *Proceedings of the 19th National Conference on Artificial Intelligence*, AAAI’04, p. 709–715. AAAI Press.
- Hansen, N. (2016). The CMA Evolution Strategy: A Tutorial. In *arXiv preprint arXiv:1604.00772*.
- Hernandez-Leal, P., Kaisers, M., Baarslag, T., & De Cote, E. M. (2017). A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. In *arXiv preprint arXiv:1707.09183*.
- Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). A Survey and Critique of Multiagent Deep Reinforcement Learning. *Autonomous Agents and Multi-Agent Systems*, 33(6), 750–797.
- Ho, F., Geraldès, R., Gonçalves, A., Rigault, B., Sportich, B., Kubo, D., Cavazza, M., & Prendinger, H. (2022a). Decentralized multi-agent path finding for uav traffic management. *IEEE Transactions on Intelligent Transportation Systems*, 23(2), 997–1008.
- Ho, F., Gonçalves, A., Rigault, B., Geraldès, R., Chicharo, A., Cavazza, M., & Prendinger, H. (2022b). Multi-Agent Path Finding in Unmanned Aircraft System Traffic Management With Scheduling and Speed Variation. *IEEE Intelligent Transportation Systems Magazine*, 14(5), 8–21.
- Huang, T., Dilkina, B., & Koenig, S. (2021). Learning Node-Selection Strategies in Bounded Suboptimal Conflict-Based Search for Multi-Agent Path Finding. In *International Joint Conference on Autonomous Agents and Multiagent Systems*.

- Huang, T., Li, J., Koenig, S., & Dilkina, B. (2022). Anytime Multi-Agent Path Finding via Machine Learning-Guided Large Neighborhood Search. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 9368–9376.
- Japkowicz, N., Hanson, S. J., & Gluck, M. A. (2000). Nonlinear Autoassociation is Not Equivalent to PCA. *Neural computation*, 12(3), 531–545.
- Jiang, M., Dennis, M., Parker-Holder, J., Foerster, J., Grefenstette, E., & Rocktäschel, T. (2021). Replay-Guided Adversarial Environment Design. *Advances in Neural Information Processing Systems*, 34, 1884–1897.
- Kaduri, O., Boyarski, E., & Stern, R. (2020). Algorithm Selection for Optimal Multi-Agent Pathfinding. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 30, pp. 161–165.
- Kingma, D. P. (2013). Auto-Encoding Variational Bayes. In *arXiv preprint arXiv:1312.6114*.
- Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. In *arXiv preprint arXiv:1609.02907*.
- Kramer, M. A. (1991). Nonlinear Principal Component Analysis using Autoassociative Neural Networks. *AIChE journal*, 37(2), 233–243.
- Kramer, M. A. (1992). Autoassociative Neural Networks. *Computers & Chemical Engineering*, 16(4), 313–328.
- Laurent, G. J., Matignon, L., Fort-Piat, L., et al. (2011). The World of Independent Learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(1), 55–64.
- Lehman, J., & Stanley, K. O. (2011). *Novelty Search and the Problem with Objectives*, pp. 37–56. Springer New York, New York, NY.
- Li, J., Chen, Z., Harabor, D., Stuckey, P. J., & Koenig, S. (2021). Anytime Multi-Agent Path Finding via Large Neighborhood Search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 4127–4135.
- Li, J., Chen, Z., Harabor, D., Stuckey, P. J., & Koenig, S. (2022). MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9), 10256–10265.
- Li, J., Felner, A., Boyarski, E., Ma, H., & Koenig, S. (2019a). Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 442–449. International Joint Conferences on Artificial Intelligence Organization.
- Li, J., Harabor, D., Stuckey, P. J., Ma, H., & Koenig, S. (2019b). Symmetry-Breaking Constraints for Grid-Based Multi-Agent Path Finding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01), 6087–6095.
- Li, J., Hoang, T. A., Lin, E., Vu, H. L., & Koenig, S. (2023). Intersection Coordination with Priority-Based Search for Autonomous Vehicles. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10), 11578–11585.

- Li, J., Tinka, A., Kiesel, S., Durham, J. W., Kumar, T. K. S., & Koenig, S. (2021). Life-long Multi-Agent Path Finding in Large-Scale Warehouses. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13), 11272–11281.
- Li, Q., Gama, F., Ribeiro, A., & Prorok, A. (2020). Graph Neural Networks for Decentralized Multi-Robot Path Planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11785–11792. IEEE.
- 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.
- 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), 529–533.
- Mouret, J.-B., & Clune, J. (2015). Illuminating Search Spaces by Mapping Elites. In *arXiv preprint arXiv:1504.04909*.
- 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).
- Narvekar, S., Sinapov, J., Leonetti, M., & Stone, P. (2016). Source Task Creation for Curriculum Learning. In *AAMAS*.
- Okumura, K. (2023). Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding. In *Proceedings of the International Joint Conference on Artificial Intelligence*.
- Okumura, K. (2024). Engineering LaCAM*: Towards Real-Time, Large-Scale, and Near-optimal Multi-Agent Pathfinding. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1501–1509.
- Oliehoek, F. A., & Amato, C. (2016). *A Concise Introduction to Decentralized POMDPs*. Springer.
- 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.
- Parker-Holder, J., Jiang, M., Dennis, M., Samvelyan, M., Foerster, J., Grefenstette, E., & Rocktäschel, T. (2022). Evolving Curricula with Regret-Based Environment Design. In *International Conference on Machine Learning*, pp. 17473–17498. PMLR.
- Peng, B., Rashid, T., de Witt, C. S., Kamienny, P.-A., Torr, P., Boehmer, W., & Whiteson, S. (2021). FACMAC: Factored Multi-Agent Centralised Policy Gradients. In Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W. (Eds.), *Advances in Neural Information Processing Systems*.
- Pham, P., & Bera, A. (2023). Crowd-Aware Multi-Agent Pathfinding with Boosted Curriculum Reinforcement Learning. In *arXiv preprint arXiv:2309.10275*.
- Phan, T., Belzner, L., Gabor, T., Sedlmeier, A., Ritz, F., & Linnhoff-Popien, C. (2021). Resilient Multi-Agent Reinforcement Learning with Adversarial Value Decomposition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13), 11308–11316.

- Phan, T., Driscoll, J., Romberg, J., & Koenig, S. (2024a). Confidence-Based Curricula for Multi-Agent Path Finding via Reinforcement Learning. In *Preprint at Research Square (Extended version of AAMAS 2024)*.
- Phan, T., Driscoll, J., Romberg, J., & Koenig, S. (2024b). Confidence-Based Curriculum Learning for Multi-Agent Path Finding. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pp. 1558–1566.
- 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 *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '20*, p. 1055–1063, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- Phan, T., Huang, T., Dilkina, B., & Koenig, S. (2024). Adaptive Anytime Multi-Agent Path Finding Using Bandit-Based Large Neighborhood Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16), 17514–17522.
- 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 Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., & Scarlett, J. (Eds.), *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202 of *Proceedings of Machine Learning Research*, pp. 27840–27853. PMLR.
- Phan, T., Ritz, F., Belzner, L., Altmann, P., Gabor, T., & Linnhoff-Popien, C. (2021). VAST: Value Function Factorization with Variable Agent Sub-Teams. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., & Vaughan, J. W. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 34, pp. 24018–24032. Curran Associates, Inc.
- Phan, T., Zhang, B., Chan, S.-H., & Koenig, S. (2025). Anytime Multi-Agent Path Finding with an Adaptive Delay-Based Heuristic. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(22), 23286–23294.
- Plappert, M., Houthoofd, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., & Andrychowicz, M. (2018). Parameter Space Noise for Exploration. In *International Conference on Learning Representations*.
- Plaut, E. (2018). From Principal Subspaces to Principal Components with Linear Autoencoders. In *arXiv preprint arXiv:1804.10253*.
- Qian, C., Zhang, Y., & Li, J. (2024). A Quality Diversity Approach to Automatically Generate Multi-Agent Path Finding Benchmark Maps. In *Proceedings of the International Symposium on Combinatorial Search*, Vol. 17, pp. 279–280.
- Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., & Whiteson, S. (2020). Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Journal of Machine Learning Research*, 21(178), 1–51.

- Ratner, D., & Warmuth, M. (1986). Finding a Shortest Solution for the $N \times N$ Extension of the 15-Puzzle is Intractable. In *Proceedings of the Fifth AAAI National Conference on Artificial Intelligence*, AAAI’86, p. 168–172. AAAI Press.
- Ren, J., Ewing, E., Kumar, T. K. S., Koenig, S., & Ayanian, N. (2024). Map Connectivity and Empirical Hardness of Grid-based Multi-Agent Pathfinding Problem. In *34th International Conference on Automated Planning and Scheduling*.
- Ren, J., Sathiyarayanan, V., Ewing, E., Senbaslar, B., & Ayanian, N. (2021). MAP-FAST: A Deep Algorithm Selector for Multi Agent Path Finding using Shortest Path Embeddings. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS ’21, p. 1055–1063, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695.
- Rubinstein, R. Y. (1997). Optimization of Computer Simulation Models with Rare Events. *European Journal of Operational Research*, 99(1), 89–112.
- Rumelhart, D. E., McClelland, J. L., Group, P. R., et al. (1986). *Parallel Distributed Processing, Volume 1: Explorations in the Microstructure of Cognition: Foundations*. The MIT Press.
- Sachdeva, N., & McAuley, J. (2023). Data Distillation: A Survey. In *Transactions on Machine Learning Research*. Survey Certification.
- Samvelyan, M., Khan, A., Dennis, M. D., Jiang, M., Parker-Holder, J., Foerster, J. N., Raileanu, R., & Rocktäschel, T. (2023). MAESTRO: Open-Ended Environment Design for Multi-Agent Reinforcement Learning. In *The 11th International Conference on Learning Representations*.
- 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.
- Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The Graph Neural Network Model. *IEEE transactions on neural networks*, 20(1), 61–80.
- Schmidhuber, J. (1991). Curious Model-Building Control Systems. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pp. 1458–1463. IEEE.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. In *arXiv preprint arXiv:1707.06347*.
- Schwefel, H.-P. P. (1993). *Evolution and Optimum Seeking: The Sixth Generation*. John Wiley & Sons, Inc.
- Sharon, G., Stern, R., Felner, A., & Sturtevant, N. R. (2015). Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 219, 40–66.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1), 1–48.

- 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.
- 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.
- Skalse, J. M. V., Howe, N. H. R., Krashenninikov, D., & Krueger, D. (2022). Defining and Characterizing Reward Gaming. In Oh, A. H., Agarwal, A., Belgrave, D., & Cho, K. (Eds.), *Advances in Neural Information Processing Systems*.
- Skrynnik, A., Andreychuk, A., Nesterova, M., Yakovlev, K., & Panov, A. (2024a). Learn to Follow: Decentralized Lifelong Multi-Agent Pathfinding via Planning and Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, pp. 17541–17549.
- Skrynnik, A., Andreychuk, A., Yakovlev, K., & Panov, A. (2024b). Decentralized Monte Carlo Tree Search for Partially Observable Multi-Agent Pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, pp. 17531–17540.
- 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 Chaudhuri, K., & Salakhutdinov, R. (Eds.), *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97 of *Proceedings of Machine Learning Research*, pp. 5887–5896. PMLR.
- Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., & Morcos, A. (2022). Beyond Neural Scaling Laws: Beating Power Law Scaling via Data Pruning. *Advances in Neural Information Processing Systems*, 35, 19523–19536.
- Soviany, P., Ionescu, R. T., Rota, P., & Sebe, N. (2022). Curriculum Learning: A Survey. *International Journal of Computer Vision*, 130(6).
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1), 1929–1958.
- Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., et al. (2019). Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, Vol. 10, pp. 151–158.
- Su, J., Adams, S., & Beling, P. (2021). Value-Decomposition Multi-Agent Actor-Critics. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13), 11352–11360.
- Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. (2018). Value-Decomposition Networks for Cooperative Multi-Agent Learning based on Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (Extended Abstract)*, AAMAS ’18, p. 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems.

- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.
- Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy Gradient Methods for Reinforcement Learning with Function Approximation. In Solla, S., Leen, T., & Müller, K. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 12, pp. 1057–1063. MIT Press.
- Tanenbaum, A. S., & Van Steen, M. (2007). *Distributed Systems: Principles and Paradigms*. Prentice-Hall.
- Tesauro, G., et al. (1995). Temporal Difference Learning and TD-Gammon. *Communications of the ACM*, 38(3), 58–68.
- Toneva, M., Sordoni, A., des Combes, R. T., Trischler, A., Bengio, Y., & Gordon, G. J. (2019). An Empirical Study of Example Forgetting during Deep Neural Network Learning. In *International Conference on Learning Representations*.
- Van Den Oord, A., Vinyals, O., et al. (2017). Neural Discrete Representation Learning. *Advances in Neural Information Processing Systems*, 30.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., & Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., & Garnett, R. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph Attention Networks. In *International Conference on Learning Representations*.
- Wager, S., Wang, S., & Liang, P. S. (2013). Dropout Training as Adaptive Regularization. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., & Weinberger, K. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 26. Curran Associates, Inc.
- Wang, J., Perez, L., et al. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *Convolutional Neural Networks Vis. Recognit*, 11 (2017), 1–8.
- Wang, J., Ren, Z., Liu, T., Yu, Y., & Zhang, C. (2020). QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *International Conference on Learning Representations*.
- Wang, R., Lehman, J., Clune, J., & Stanley, K. O. (2019). POET: Open-Ended Coevolution of Environments and their Optimized Solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 142–151. ACM.
- Wang, Y., Xiang, B., Huang, S., & Sartoretti, G. (2023a). SCRIMP: Scalable Communication for Reinforcement-and Imitation-Learning-Based Multi-Agent Pathfinding. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9301–9308. IEEE.
- Wang, Z., Jiang, Y., Zheng, H., Wang, P., He, P., Wang, Z. A., Chen, W., & Zhou, M. (2023b). Patch Diffusion: Faster and More Data-Efficient Training of Diffusion Models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., & Levine, S. (Eds.), *Advances in Neural Information Processing Systems*, Vol. 36, pp. 72137–72154. Curran Associates, Inc.
- Watkins, C. J., & Dayan, P. (1992). Q-Learning. *Machine Learning*, 8(3-4), 279–292.

- Xiao, Z., Kreis, K., & Vahdat, A. (2022). Tackling the Generative Learning Trilemma with Denoising Diffusion GANs. In *International Conference on Learning Representations*.
- Yan, Z., & Wu, C. (2024). Neural Neighborhood Search for Multi-Agent Path Finding. In *The 12th International Conference on Learning Representations*.
- Yang, S., Xie, Z., Peng, H., Xu, M., Sun, M., & Li, P. (2023). Dataset Pruning: Reducing Training Data by Examining Generalization Influence. In *The 11th International Conference on Learning Representations*.
- Yu, C., Velu, A., Vinitzky, 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, 24611–24624.
- Yu, J., & LaValle, S. (2013). Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1), 1443–1449.
- Zhang, S., Li, J., Huang, T., Koenig, S., & Dilkina, B. (2022). Learning a Priority Ordering for Prioritized Planning in Multi-Agent Path Finding. In *Proceedings of the Symposium on Combinatorial Search*, pp. 208–216.
- Zhang, Y., Fontaine, M. C., Bhatt, V., Nikolaidis, S., & Li, J. (2023a). Arbitrarily Scalable Environment Generators via Neural Cellular Automata. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pp. 57212–57225.
- Zhang, Y., Fontaine, M. C., Bhatt, V., Nikolaidis, S., & Li, J. (2023b). Multi-Robot Coordination and Layout Design for Automated Warehousing. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 5503–5511.
- Zhang, Y., Jiang, H., Bhatt, V., Nikolaidis, S., & Li, J. (2024). Guidance Graph Optimization for Lifelong Multi-Agent Path Finding. In Larson, K. (Ed.), *Proceedings of the 33rd International Joint Conference on Artificial Intelligence*, pp. 311–320. International Joint Conferences on Artificial Intelligence Organization. Main Track.
- 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*, pp. 1–8. IEEE.
- Zheng, K., Phan, T., Koenig, S., & Kumar, T. S. (2025). A FastMap-Based Encoder-Decoder Architecture and Its Applications. *Proceedings of the 11th International Conference on Learning, Optimization and Data*.