STRUCTURED REPRESENTATION LEARNING IN MODEL-BASED MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose a novel value-based multi-agent reinforcement learning (MARL) method that integrates structured representation learning into world model imagination. We use latent states from recurrent and representation models to build the agent networks, injecting permutation-invariant and permutation-equivariant inductive biases into the networks and constraining the network embeddings to a product of probability simplices. These well-structured representations of the agents transfer smoothly across different team compositions. Moreover, we present a world model as the imagination module of our framework and apply the simplicial embeddings to the model latents. The world model supplies the mixing network with imagined trajectories, guiding agent learning through well-shaped representations and improving sample efficiency. Across different MARL benchmarks, our method demonstrates consistent gains over value-based and actor-critic baseline methods in both sample efficiency and asymptotic performance. We perform ablation experiments, design-space studies, and analyses of learned representations to highlight the importance of the main framework components within a model-based multi-agent paradigm.

1 INTRODUCTION

Agents in partially observable and highly dynamic environments require a large number of interactions with the environment to accomplish learning tasks. As the number of agents n increases, the joint action space $|\mathcal{A}|^n$ expands exponentially, leading to sample inefficiency and poor scalability. To address these problems, we introduce **M**odel-Based **S**implicial **EM**beddings for **M**ulti-Agent RL (MSEMM), a value-based multi-agent reinforcement learning (MARL) framework under the Centralized Training with Decentralized Execution (CTDE) paradigm (Rashid et al., 2020b). To our knowledge, MSEMM is the first value-based CTDE framework to jointly combine permutation-consistent agent representations, simplicial state-action geometry, and a centralized learned world model for imagination. This combination targets several limitations of model-free MARL, apart from the aforementioned challenges: sparse and non-stationary supervision by temporal-difference (TD) learning and the concatenation of entity features in arbitrary fixed orders by standard agent networks (Jianye et al., 2023). The latter treats different permutations of the same physical configuration as distinct inputs, enlarging the effective observation space and hindering generalization across team compositions.

The individual ingredients of MSEMM are motivated by successful ideas from single-agent RL, but their integration into value-based MARL is nontrivial. Structured representation learning, including simplicial embeddings (SEM) (Lavoie et al., 2023; Obando-Ceron et al., 2026), constrains latent geometry, while learned world models can provide predictive structure for value learning (Fujimoto et al., 2025; Chang et al., 2026). Standard value-decomposition networks, however, map agent histories directly to utilities and provide no explicit per-agent state-action representation on which such geometric constraints can act. World-model planning is also poorly suited to combinatorial discrete joint action spaces $|\mathcal{A}|^n$, and centralized model representations are unavailable during decentralized execution. Existing symmetric architectures (Jianye et al., 2023) enforce permutation structure at the utility level, but leave their internal representations unconstrained and disconnected from model-based supervision. MSEMM bridges these gaps through two complementary representation levels.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

On the decentralized side, each agent constructs a simplicial state-action embedding ϕ^{oa} from a permutation-invariant backbone network, a targeted-entity representation, and a target-free action descriptor. This yields permutation invariance (PI) for entity-independent actions and permutation equivariance (PE) for targeted actions by construction. On the centralized side, a simplicial world model learns separate global state and joint state-action representations, generating multi-step imagined rollouts. The trajectory encoding conditions the monotonic mixing network together with the decentralized utilities, connecting model-based predictive structure to multi-agent value decomposition without exposing the world model to the agents at execution time.

Contributions. We make three main contributions: **(1) Symmetry-consistent simplicial state-action representations.** We introduce an explicit per-agent state-action embedding ϕ^{oa} that combines permutation-consistent structures with simplicial regularization. To our knowledge, this is the first value-based MARL representation to jointly provide PI/PE guarantees and a geometrically constrained state-action interface for learning. **(2) Centralized simplicial world modeling for value factorization.** We introduce a world model with distinct simplicial global state and joint state-action representations, and connect its imagined rollouts to decentralized utilities through the monotonic mixing network. This combines structured decentralized representations with centralized predictive representations while preserving standard decentralized execution. **(3) Empirical gains and comprehensive studies on MARL benchmarks.** Across StarCraft Multi-Agent Challenges (SMAC) (Samvelyan et al., 2019), SMACv2 (Ellis et al., 2023), and Multi-Agent MuJoCo (Peng et al., 2021), MSEMM improves sample efficiency and asymptotic performance over competitive baselines. We conduct detailed studies on the ablations, design space, and learned strategies to show the individual contributions of MSEMM components and demonstrate effective model learning.

2 Related Work

Multi-agent value decomposition and permutation symmetry. Centralized Training with Decentralized Execution (CTDE) (Kraemer & Banerjee, 2016; Gronauer & Diepold, 2021) conditions centralized critics on global states while agents act on local histories. Value factorization methods (Sunehag et al., 2017; Rashid et al., 2018; 2020b; Son et al., 2019; Rashid et al., 2020a; Liu et al., 2023; Jeon et al., 2022; Zhang et al., 2023) enforce the Individual-Global-Max (IGM) property, aligning decentralized greedy actions with joint value maximization. Centralized-critic policy-gradient methods provide an alternative (Lowe et al., 2017; Foerster et al., 2018; Iqbal & Sha, 2018; Peng et al., 2021; de Witt et al., 2020; Yu et al., 2022; Zhou et al., 2020; Kuba et al., 2022; Zhong et al., 2024) under CTDE. Standard agent networks, however, treat all permutations of equivalent configurations as distinct (Jianye et al., 2023). MSEMM exploits PI and PE properties in MARL (Iqbal & Sha, 2018; Li et al., 2021; Jianye et al., 2023) and improves previous work by constraining internal representations and interfacing directly with centralized world-model supervision.

World models for reinforcement learning. Model-based RL uses learned dynamics for synthetic data augmentation (Sutton, 1991; Kuvayev & Sutton, 1996; Deisenroth & Rasmussen, 2011; Chua et al., 2018; Janner et al., 2019), value expansion (Feinberg et al., 2018; Buckman et al., 2018), policy optimization through imagined rollouts (Racaniere et al., 2017; Oh et al., 2017; Silver et al., 2017; Ha & Schmidhuber, 2018; Hafner et al., 2019b; 2020; 2021; 2022; 2023), and online planning (Schrittwieser et al., 2020; Ye et al., 2021; Hansen et al., 2022; Wang et al., 2024; Hansen et al., 2024; Lin et al., 2025; Wang et al., 2025). Long rollouts can suffer from diminishing returns, model error, and objective mismatch (Van Hasselt et al., 2019; Lambert et al., 2020; Palenicek et al., 2023). In model-based MARL (Yang & Wang, 2020; Luo et al., 2024), early development focuses on theoretical analyses (Brafman & Tennenholtz, 1999; 2003). Recent methods (Pasztor et al., 2021; Zhang et al., 2020; Wang et al., 2022; Egorov & Shpilman, 2022; Sessa et al., 2022; Venugopal et al., 2024; Ma et al., 2024) study various application domains and communication problems, but the research in well-structured representations and permutation symmetry remains unexplored.

Structured representation learning. Structured representation learning spans bisimulation abstractions (Li et al., 2006; Ferns et al., 2011), auxiliary prediction (Watter et al., 2015; Jaderberg et al., 2017), self-predictive dynamics (Guo et al., 2020; Schwarzer et al., 2021; Guo et al., 2022; Zhao et al., 2023; Schwarzer et al., 2023), contrastive objectives (Zheng et al., 2024; Assran et al., 2023), and goal-reaching objectives (Liu et al., 2023; Nimonkar et al., 2025). Explicit state-action embeddings in OFENet (Ota et al., 2020), SALE (Fujimoto et al., 2024), and MR.Q (Fujimoto et al., 2025) provide intermediate representations for critic learning. Geometric constraints such as vector

quantization (Scannell et al., 2024), hyperspherical normalization (Lee et al., 2025), and Simplicial Embeddings (SEM) (Lavoie et al., 2023; Hansen et al., 2024; Chang et al., 2026) regularize representations architecturally without auxiliary-loss tuning. In actor-critic networks, SEM bounds feature activations and stabilizes bootstrapping (Obando-Ceron et al., 2026). To our knowledge, MSEMM is the first to apply simplicial regularization across both decentralized and centralized multi-agent tiers. SEM constrains the symmetry-consistent state-action embedding ϕ^{oa} on the decentralized side, and the world-model encoder and predicted categorical latents on the centralized side.

3 Background

Decentralized Partially Observable MDP. We model cooperative partially observable multi-agent tasks as the Dec-POMDP, defined by $M := \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, R, \Omega, O, \gamma \rangle$ (Oliehoek & Amato, 2016), where $\mathcal{N} := \{1, \dots, n\}$ is the agent set, $\mathcal{S}$ the state space, $\mathcal{A} := \times_{i \in \mathcal{N}} \mathcal{A}^i$ the joint action space, P the transition function, Ω and O the observation space and function. At time t, agent i selects $a_t^i \in \mathcal{A}^i$ from its local action-observation history τ_t^i according to $\pi^i(a_t^i \mid \tau_t^i)$, forming the joint action $\mathbf{a}_t = (a_t^1, \dots, a_t^n)$. All agents receive $r_t = R(s_t, \mathbf{a}_t)$ and optimize the discounted return $\mathcal{G}_t = \sum_{k \geq 0} \gamma^k r_{t+k}$. The joint Q function is $Q^{\boldsymbol{\pi}}(s_t, \mathbf{a}_t) = \mathbb{E}[\mathcal{G}_t \mid s_t, \mathbf{a}_t]$, with $\boldsymbol{\pi} := (\pi^1, \dots, \pi^n)$.

Entity-factored modeling. In entity-factored MARL (Jianye et al., 2023), the environment contains m entities: n learning agents and $m - n$ non-player or adversarial entities. States and observations are represented by entity features, $s \in \mathbb{R}^{m \times d_s}$ and $o^i \in \mathbb{R}^{m \times d_o}$, where d_s and d_o are the corresponding per-entity feature dimensions. The rows of o^i comprise the agent itself, $n - 1$ allies, and $m - n$ non-player entities. Each action space contains entity-independent actions $\mathcal{A}_{\text{inv}}$ and entity-targeted actions $\mathcal{A}_{\text{eqv}}$, i.e., $\mathcal{A}^i := (\mathcal{A}_{\text{inv}}, \mathcal{A}_{\text{eqv}})$. Detailed explanations can be found in Appx. A.

Permutation invariance and equivariance. Let S_k denote the permutations of $\{1, \dots, k\}$ and P_σ the permutation matrix of $\sigma \in S_k$. A function f is permutation-equivariant (PE) if $f(P_\sigma X) = P_\sigma f(X)$, and permutation-invariant (PI) if $f(P_\sigma X) = f(X)$, for $X \in \mathbb{R}^{k \times d}$. Under entity-factored modeling, let $g = (\sigma, \sigma') \in G \cong S_{m-n} \times S_{n-1}$, where σ and σ' permute adversarial and allied rows, respectively. For $a_i = (a_{\text{inv}}, a_{\text{eqv}}) \in \mathcal{A}^i$, we define $g a_i = (a_{\text{inv}}, g a_{\text{eqv}})$.

Recurrent state-space models. Recurrent state-space models (RSSMs) summarize an agent's observation history using a deterministic recurrent state and a stochastic latent state (Hafner et al., 2019a; 2021). Given observation o_t and previous action a_{t-1}, the deterministic state is updated as $h_t = f_\psi(h_{t-1}, z_{t-1}, a_{t-1})$, and the stochastic state is sampled from the representation model as $z_t \sim q_\psi(z_t \mid h_t, o_t)$. The pair (h_t, z_t) therefore provides a compact latent representation of the agent's action-observation history. We use this recurrent deterministic–stochastic decomposition as the basis for the per-agent latent states introduced in Section 4.1.

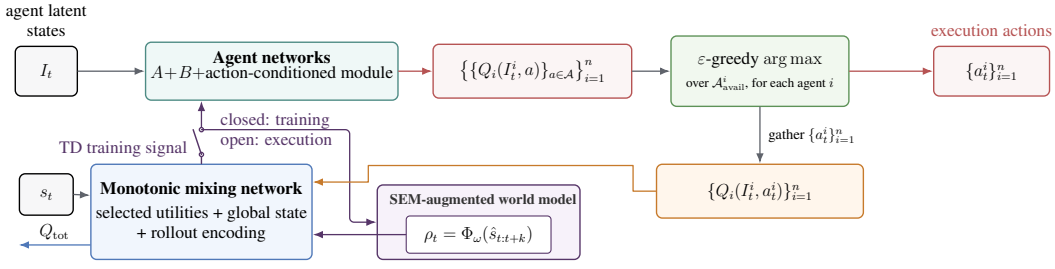

Figure 1: A high-level overview of MSEMM. The utilities of decentralized agents, which are represented by the agent networks, determine execution actions, while SEM-equipped world-model rollouts augment value decomposition during centralized training.

4 Methodology

We present MSEMM, a value-based MARL framework for Centralized Training with Decentralized Execution (CTDE) that combines structured agent representations with model-based imagination. Figure 1 shows the overall information flow.

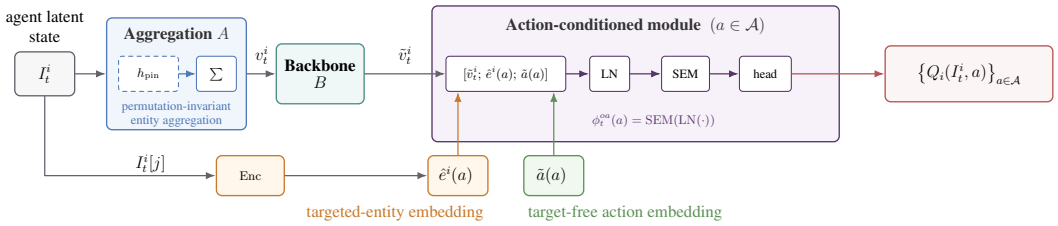

Figure 2: The Q-network (utility network) of an agent, with I_t^i as the input and the utilities as the output. SEM is applied to the action-conditioned module of the Q-network.

At time t, each agent i maintains a local latent state I_t^i constructed from its action-observation history. A joint permutation-symmetric agent network, consisting of individual agent networks, independently evaluates the available actions, producing the utilities $\{Q_i(I_t^i, a)\}_{a\in\mathcal{A}}$. Within this network, an invariant backbone representation is combined with action-dependent information to construct the simplicial state-action representation $\phi_t^{oa}(a)$ for each candidate action. The corresponding utility is then computed from $\phi_t^{oa}(a)$. Each agent selects a_t^i using only its local latent state and per-action utilities, so execution remains fully decentralized.

During the centralized training phase, the agents' utilities corresponding to the selected actions, $\{Q_i(I_t^i, a_t^i)\}_{i=1}^n$, are combined by a monotonic mixing network. The mixer is additionally conditioned on the global state s_t and on a rollout encoding $\rho_t = \Phi_\omega(\hat{s}_{t:t+k})$ produced by the SEM-augmented world model. The world model generates multi-step imagined trajectories from the current joint latent state, allowing the mixer to incorporate predictive information beyond the current global state. Neither the world model nor the mixer is required for decentralized execution.

Centralized training optimizes the agent, world-model, and mixing-network parameters using TD regression, together with auxiliary latent-prediction, dynamics, and reward-prediction objectives. We develop these components in the order in which they appear in the framework. We first define the entity-factored latent state I_t^i for each agent i. Then, we introduce the permutation-symmetric Q-network and its action-conditioned simplicial representation. We next present the centralized SEM-augmented world model and its imagined rollout encoding, followed by the details of the mixing network and the joint learning objective of MSEMM.

4.1 Entity-factored latent state.

We now detail our approach for uniting entity-factored models with well-structured latent representation methods. Each agent $i \in \mathcal{N}$ maintains two models: the recurrent model $f_\psi(\cdot)$ and the representation model $q_\psi(\cdot)$, for each entity, j, which can be written as

$$h_t^i[j] = f_\psi\big(h_{t-1}^i[j],\ z_{t-1}^i[j],\ \alpha_{t-1}^i[j]\big)\,, \qquad z_t^i[j] \sim q_\psi\big(z_t^i[j] \mid h_t^i[j],\ o_t^i[j]\big)\,.$$

We use ψ to denote the parameters of both models, and ψ is shared by all components of the same entity class (adversarial, allied, and own classes), so that h_t^i and z_t^i are row-indexed. q_ψ is a categorical distribution sampled with straight-through gradients. The latent state of agent i is denoted as $I_t^i := (h_t^i,\ z_t^i)$, with components $I_t^i[j] = [\,h_t^i[j];\ z_t^i[j]\,]$. The input $\alpha_{t-1}^i[j]$ is defined as follows: if j is an adversarial or allied entity, $\alpha_{t-1}^i[j] = \mathbf{1}[a_{t-1}^i = a_{\text{eqv}}(j)]$, the indicator of the action targeting j; if j is the agent's own entity, $\alpha_{t-1}^i[j] = \tilde{a}(a_{t-1}^i)$, an embedding we define in Equation (2). Since $h_t^i[j]$ and $z_t^i[j]$ are only conditioned on agent i's history, I_t^i can be viewed as a function of τ_t^i alone. The joint representation of $q_\psi(\cdot)$ can be factorized as a product over all agents, i.e., $q_\psi\big(z_t^{1:n} \mid h_t^{1:n}, o_t^{1:n}\big) = \prod_{i=1}^n q_\psi\big(z_t^i \mid h_t^i, o_t^i\big)$. The latent state I_t^i is used as the input to the Q-network of the agent module.

4.2 Q-Network with PI and PE Properties

As shown in Fig. 2, the Q-network maps the latent state I_t^i to the utilities $Q_i(I_t^i, a)$ for all $a \in \mathcal{A}$. It consists of a PI entity-aggregation layer A, a backbone B, and an action-conditioned module that preserves PI for entity-independent actions and PE for entity-targeted actions.

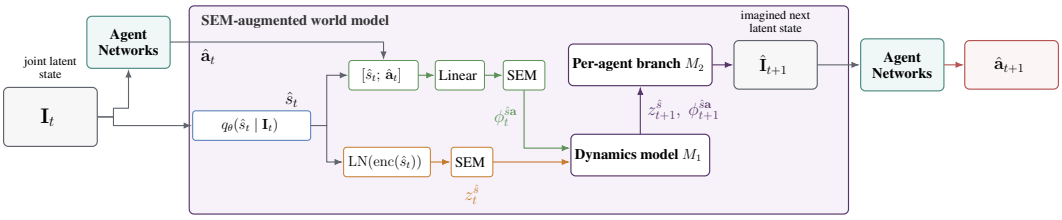

Figure 3: One rollout step of the imagination of the SEM-augmented world model.

Entity-aggregation layer. Let $c \in \{\text{adv}, \text{ally}\}$ denote the adversarial and allied entity classes, $I_t^i[j_c]$ denote the j-th entity component of class c, and m_c denote the number of entity components in the class c, $m_{\text{adv}} = m - n$ and $m_{\text{ally}} = n - 1$. For the entity-aggregation layer A, we use a weight-generating hypernetwork $h_{\text{pin}}^{(c)}$, with parameters shared by all entities of c. It produces a weight matrix for each component,

$$w_c[j] = h_{\text{pin}}^{(c)}\big(I_t^i[j_c]\big) = \text{MLP}^{(c)}\big(\big[\, h_t^i[j_c];\; z_t^i[j_c] \,\big]\big)\,, \qquad j \in \{1, \ldots, m_c\}.$$

The layer output sums the products within each class and adds the agent's own component:

$$v_t^i = A(I_t^i) = \sum_{c \in \{\text{adv},\text{ally}\}} \sum_{j_c=1}^{m_c} I_t^i[j_c]\, w_c[j] \;+\; \mathbf{w}_{\text{own}}\, I_t^i[\text{own}],$$

with entity components treated as row vectors. The weight $w_c[j]$ is generated only from component j by a network shared with all components of the class of j. Therefore, the component j contributes the same $I_t^i[j_c]\, w_c[j]$ under permutations. The products of a class are combined by a sum and the own component, which is handled by the fixed matrix $\mathbf{w}_{\text{own}}$. Formally, for any $g = (\sigma, \sigma') \in G$ that acts on I_t^i through its entity components, $(g(I_t^i))[\sigma(j)_{\text{adv}}] = I_t^i[j_{\text{adv}}]$, $(g(I_t^i))[\sigma'(j)_{\text{ally}}] = I_t^i[j_{\text{ally}}]$, and g leaves the own component in place. Therefore, $A(g(I_t^i)) = A(I_t^i)$, and A satisfies PI.

Backbone. The backbone network B of an agent is typically an RNN. It maps v_t^i and its own previous state $\tilde{v}_{t-1}^i$ to the output $\tilde{v}_t^i = B(v_t^i, \tilde{v}_{t-1}^i)$, from an initial state $\tilde{v}_0^i$ that does not depend on the history. Because $\tilde{v}_0^i$ is fixed, unrolling the recurrence yields $\tilde{v}_t^i = \bar{B}(v_1^i, \ldots, v_t^i)$ for a function $\bar{B}$ determined by B and $\tilde{v}_0^i$.

Proposition 1. *Let $g \in G$ act on a history $(I_1^i, \ldots, I_t^i)$ by operating on every element through its entity components. If A is permutation-invariant, i.e. $A(g(I)) = A(I)$ for all $g \in G$ and all I, then for any function $\bar{B}$, the backbone output $\tilde{v}_t^i = \bar{B}\big(A(I_1^i), \ldots, A(I_t^i)\big)$ is permutation-invariant, i.e. it takes the same value on $(g(I_1^i), \ldots, g(I_t^i))$ as on $(I_1^i, \ldots, I_t^i)$ for every $g \in G$.*

Because A is PI, $\bar{B}$ receives the same sequence under any admissible entity permutation and therefore returns the same output. The full proof can be found in Appx. B.

Action-conditioned module. The action-conditioned module in Fig. 2 combines the invariant backbone representation with a targeted-entity embedding $\hat{e}^i(a)$ and a target-free action embedding $\tilde{a}(a)$:

$$\phi_t^{oa}(a) \;=\; \text{SEM}\Big(\text{LN}\big(\big[\, \tilde{v}_t^i\,;\; \hat{e}^i(a)\,;\; \tilde{a}(a) \,\big]\big)\Big), \quad Q_i\big(I_t^i, a\big) \;=\; h_{\text{head}}(\phi_t^{oa}(a))\,, \quad a \in \mathcal{A}. \tag{1}$$

In (1), the embeddings $\hat{e}^i(a)$ and $\tilde{a}(a)$ are defined as

$$\begin{aligned} \hat{e}^i(a) &= \mathbf{1}[a \in \mathcal{A}_{\text{eqv}}]\, \text{Enc}\big(I_t^i[j]\big) + \mathbf{1}[a \in \mathcal{A}_{\text{inv}}]\, \hat{e}_f, \\ \tilde{a}(a) &= \big[\, \mathbf{1}[a \in \mathcal{A}_{\text{inv}}]\, \text{onehot}_{\mathcal{A}_{\text{inv}}}(a)\;;\; \mathbf{1}[a \in \mathcal{A}_{\text{eqv}}] \,\big], \end{aligned} \tag{2}$$

where j is the entity targeted by $a = a_{\text{eqv}}(j)$, $\hat{e}_f$ is a learned vector shared by all entity-independent actions, and $\text{onehot}_{\mathcal{A}_{\text{inv}}}(a)$ distinguishes those actions. Thus, for entity-targeted actions, target identity enters through $\hat{e}^i(a)$ while $\tilde{a}(a)$ contains no target-slot information.

We use SEM to constrain $\phi_t^{oa}(a)$ to a product of probability simplices. For $\phi \in \mathbb{R}^{L \times V}$,

$$\text{SEM}(\phi)_{\ell,v} = (\exp(\phi_{\ell,v}/\tau))(\textstyle\sum_{v'=1}^{V} \exp(\phi_{\ell,v'}/\tau))^{-1},\ \ell \in \{1, \ldots, L\},\ v \in \{1, \ldots, V\}, \tag{3}$$

where $\tau > 0$ is the temperature, and $\text{SEM}(\phi) \in (\Delta^{V-1})^L$. The LN and h_{head} in (1) denote the layer normalization and the output head mapping $\phi_t^{oa}(a)$ to $Q_i\big(I_t^i, a\big)$, respectively.

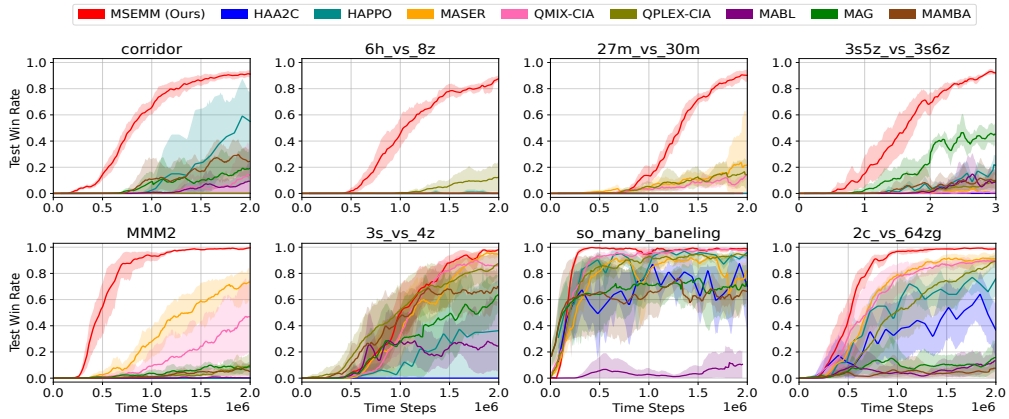

Figure 4: Test Win Rate of MSEMM compared to the other MARL algorithms on SMAC scenarios.

Proposition 2. *Let $g \in G$ act on the history $(I_1^i, \ldots, I_t^i)$ as in Proposition 1 and on $\mathcal{A}$ as in Section 3. Suppose the action-conditioned module receives (i) a permutation-invariant backbone output $\tilde{v}_t^i$; (ii) a function $\hat{e}^i$ that satisfies $\hat{e}^i(a_{\mathrm{eqv}}(j)) = \mathrm{Enc}(I_t^i[j])$ for all $a_{\mathrm{eqv}}(j) \in \mathcal{A}_{\mathrm{eqv}}$ and $\hat{e}^i(a) = \hat{e}_f$ for all $a \in \mathcal{A}_{\mathrm{inv}}$; and (iii) an embedding $\tilde{a}(a)$ that is constant on $\mathcal{A}_{\mathrm{eqv}}$. Then the utility satisfies PI on $\mathcal{A}_{\mathrm{inv}}$ and PE on $\mathcal{A}_{\mathrm{eqv}}$, i.e., $Q_i(g(I_t^i), g \cdot a) = Q_i(I_t^i, a)$ for all $g \in G$ and $a \in \mathcal{A}$, where $Q_i(g(I_t^i), \cdot)$ is the utility computed on the permuted history.*

The full proof is given in Appx. B, which shows that each input to the action-conditioned module transforms consistently under g. Proposition 1 provides the invariant backbone representation, while Equation (2) ensures that the targeted-entity embedding follows the permuted target and that the action code remains target-slot independent.

At execution time, unavailable actions are masked and agent i selects from available actions $a_t^i = \arg\max_{a \in \mathcal{A}_{\mathrm{avail}}^i} Q_i\big(I_t^i, a\big)$. The utilities corresponding to the selected actions are passed to the centralized mixing network during training.

4.3 SEM-Augmented World Model

The previous subsection defines the decentralized utilities $Q_i(I_t^i, a)$ and the actions selected by each agent. During centralized training, MSEMM augments these utilities with predictive information from a SEM-augmented world model. Given the agents' joint latent state, the world model generates a multi-step imagined rollout whose encoding ρ_t is provided to the mixing network together with the selected utilities and the global state.

Simulation. Figure 3 shows one step of world-model imagination. At time t, the joint latent state $\mathbf{I}_t = (I_t^1, \ldots, I_t^n)$ is provided to the representation model q_θ, while the agent networks from Section 4.2 produce the joint action $\hat{\mathbf{a}}_t$. The representation model infers $\hat{s}_t \sim q_\theta(\hat{s}_t \mid \mathbf{I}_t)$. We encode this state as $z_t^{\hat{s}} = \mathrm{SEM}(\mathrm{LN}(\mathrm{enc}(\hat{s}_t)))$, while the joint state–action representation is $\phi_t^{\hat{s}\mathbf{a}} = \mathrm{SEM}(\mathrm{Linear}([\hat{s}_t; \hat{\mathbf{a}}_t]))$. The dynamics model M_1 maps $(z_t^{\hat{s}}, \phi_t^{\hat{s}\mathbf{a}})$ to predicted next-step representations $(z_{t+1}^{\hat{s}}, \phi_{t+1}^{\hat{s}\mathbf{a}})$. The same simplicial normalization is used for encoded and predicted representations so that they remain in the same representation spaces. For rollout notation, we set $\hat{s}_{t+1} := z_{t+1}^{\hat{s}}$. A second dynamics branch M_2 uses $(z_{t+1}^{\hat{s}}, \phi_{t+1}^{\hat{s}\mathbf{a}})$ to infer the imagined joint agent latent state $\hat{\mathbf{I}}_{t+1}$. The agent networks then produce the next joint action $\hat{\mathbf{a}}_{t+1}$ from $\hat{\mathbf{I}}_{t+1}$, completing one imagination step. Repeating this procedure produces a rollout of horizon k. SEM therefore constrains the state, state–action, and predicted latent representations throughout the world model.

The imagined sequence $\hat{s}_{t:t+k} = (\hat{s}_t, \ldots, \hat{s}_{t+k})$ is summarized by the trajectory encoder as $\rho_t = \Phi_\omega(\hat{s}_{t:t+k})$. This rollout encoding provides the mixer with predictive context in addition to the current global state. Together with the selected utilities from Section 4.2, the monotonic mixing network computes $Q_{\mathrm{tot}} = \mathrm{Mixer}\big(\{Q_i(I_t^i, a_t^i)\}_{i=1}^n,\ s_t,\ \rho_t\big)$.

4.4 Learning Objective

MSEMM is trained jointly with a TD objective and four auxiliary losses. We denote the agent, world-model, and mixer parameters by ψ, θ, and ξ, with target-network parameters ψ^- and ξ^-.

Latent state predictions. The loss functions for the latent states in the world model minimize the mean-squared error (MSE) between the predicted latent states and the replayed next-step latent states. The function for the latents $(z^{\hat{s}}_{t+1}, \phi^{\hat{s}\mathbf{a}}_{t+1})$ is defined as $\mathcal{L}_{\mathrm{LS}}(\theta) = \mathrm{MSE}\big(z^{\hat{s}}_{t+1}, \mathrm{sg}(\hat{s}^{\mathrm{rep}}_{t+1})\big) + \mathrm{MSE}\big(\phi^{\hat{s}\mathbf{a}}_{t+1}, \mathrm{sg}(\phi^{\mathrm{rep}}_{t+1})\big)$, where $\hat{s}^{\mathrm{rep}}_{t+1}$ is the replayed next state embedding, $\phi^{\mathrm{rep}}_{t+1}$ is the replayed next state-action embedding, and sg is the stop-gradient operation. Moreover, the loss function for the latent states corresponding to the agents is defined as $\mathcal{L}_{\mathrm{ALS}}(\theta) = \sum_{i=1}^{n} \mathrm{MSE}\big(\hat{I}^i_{t+1}, \mathrm{sg}(I^i_{t+1})\big)$, where I^i_{t+1} denotes the replayed next latent state of agent i.

Dynamics loss of agents. The embedding of agent i's taken action, $\phi^{oa,i}_t := \phi^{oa}_t(a^i_t)$, predicts the agent's next backbone embedding. The dynamics loss function for all agents is written as $\mathcal{L}_{\mathrm{dyn}}(\psi, \theta) = \sum_{i=1}^{n} \mathrm{MSE}\big((\phi^{oa,i}_t), \mathrm{sg}(\tilde{v}^i_{t+1})\big)$.

Reward prediction. The per-agent embeddings $\phi^{oa,i}_t$ jointly predict the team reward through a shared linear head, which can be expressed as $\mathcal{L}_{\mathrm{rew}}(\psi, \theta) = \big(\sum_{i=1}^{n} w_r^{\mathsf{T}} \phi^{oa,i}_t - r_t\big)^2$. The weight $w_r \in \mathbb{R}^{L \times V}$ is a learned vector shared by all agents.

TD loss. The joint action-value function is trained by TD regression with double-Q target selection. The online Q-network selects the agent's next action, evaluated by the target network:

$\mathcal{L}_{\mathrm{TD}}(\psi, \xi, \theta) = \big(y_t - Q_{\mathrm{tot}}\big)^2$, where $y_t = r_t + \gamma\, \overline{\mathrm{Mixer}}\big(\{\bar{Q}_i(\bar{I}^i_{t+1}, a^{i\star}_{t+1})\}_{i=1}^{n}\,,\ s_{t+1}\,, \mathrm{sg}(\rho_{t+1})\big)$, and $a^{i\star}_{t+1} = \arg\max_a Q_i(I^i_{t+1}, a)$. The bars denote the computations performed by the target networks with parameters ψ^- and ξ^-. The loss is averaged over valid steps of the replayed episodes.

Overall learning objective. Finally, the learning objective for the MSEMM method is written as:

$$\mathcal{L}_{\mathrm{total}}(\psi, \xi, \theta) = \mathcal{L}_{\mathrm{TD}} + c_{\mathrm{LS}}\, \mathcal{L}_{\mathrm{LS}} + c_{\mathrm{ALS}}\, \mathcal{L}_{\mathrm{ALS}} + c_{\mathrm{dyn}}\, \mathcal{L}_{\mathrm{dyn}} + c_{\mathrm{rew}}\, \mathcal{L}_{\mathrm{rew}}, \tag{4}$$

with weights $c_{\mathrm{LS}}, c_{\mathrm{ALS}}, c_{\mathrm{dyn}}, c_{\mathrm{rew}} > 0$ that set the scale of each term relative to $\mathcal{L}_{\mathrm{TD}}$.

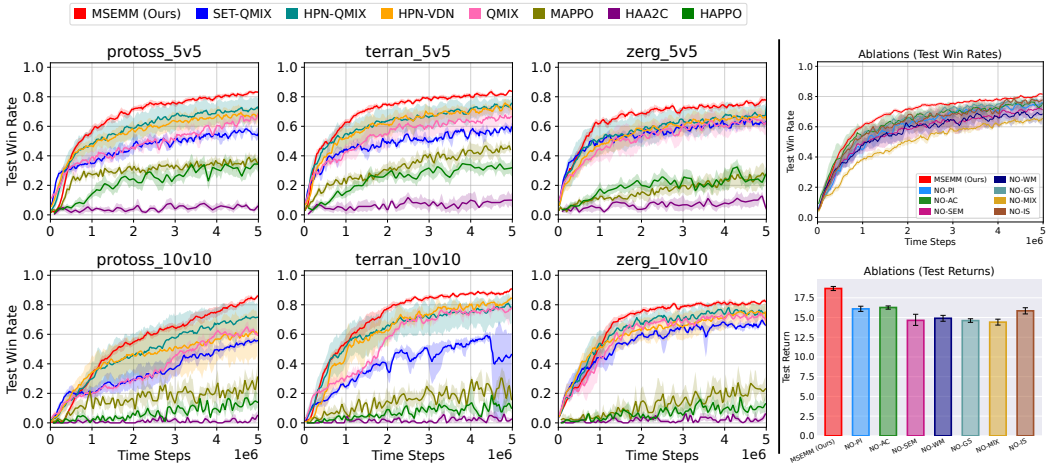

Figure 5: **SMACv2 evaluation and ablations**. The three columns on the left show the performance of MSEMM compared to MARL baselines in the six SMACv2 environments: the Protoss, Terran, and Zerg races at 5v5 (top) and 10v10 (bottom). The rightmost column displays the test win rate (top) and test return (bottom) for the ablations of MSEMM, averaged over the SMACv2 scenarios.

5 Experiments

We demonstrate that MSEMM improves sample efficiency and performance over value-based and actor-critic baseline methods by conducting experiments on different MARL benchmarks, including

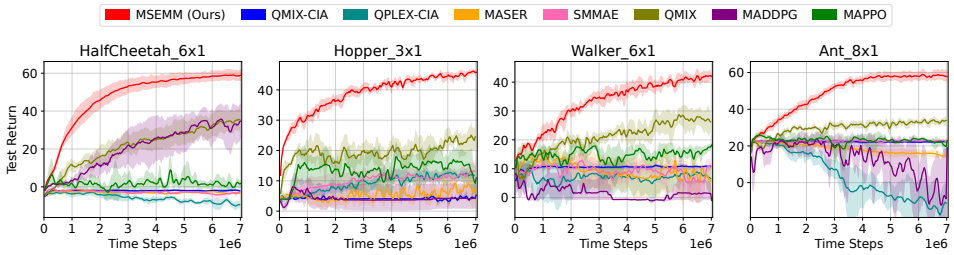

Figure 6: Test return of MSEMM against the baselines on four Multi-Agent MuJoCo tasks: *HalfCheetah_6x1*, *Hopper_3x1*, *Walker_6x1*, and *Ant_8x1*. The shaded region is the 95% bootstrap CI.

SMAC, SMACv2, and Multi-Agent MuJoCo (MAMuJoCo). We study the impact of the MSEMM components using ablation experiments and design space exploration. Moreover, we show that the world model learns useful features by analyzing the battle scenarios and learned representations.

Baselines. MSEMM is evaluated against a broad suite of model-based MARL methods (MAG (Wu et al., 2023), MAMBA (Egorov & Shpilman, 2022), and MABL (Venugopal et al., 2024)), model-free value-based MARL algorithms (VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2020b), SET-QMIX (Li et al., 2021), MASER (Jeon et al., 2022), QMIX-CIA (Liu et al., 2023), QPLEX-CIA (Liu et al., 2023), SMMAE (Zhang et al., 2023), HPN-QMIX (Jianye et al., 2023), and HPN-VDN (Jianye et al., 2023)), and model-free actor-critic MARL methods (MADDPG (Lowe et al., 2017), MAPPO (Yu et al., 2022), HAPPO (Kuba et al., 2022; Zhong et al., 2024), and HAA2C (Zhong et al., 2024)). The SMAC environments are run for 2M steps, SMACv2 for 5M steps, and MAMuJoCo for 7M steps. Each experiment in a single environment is performed with five random seeds. Each learning curve shows the mean test win rate or test return across seeds, depending on the specific task; the shaded region represents the 95% bootstrap confidence interval (CI).

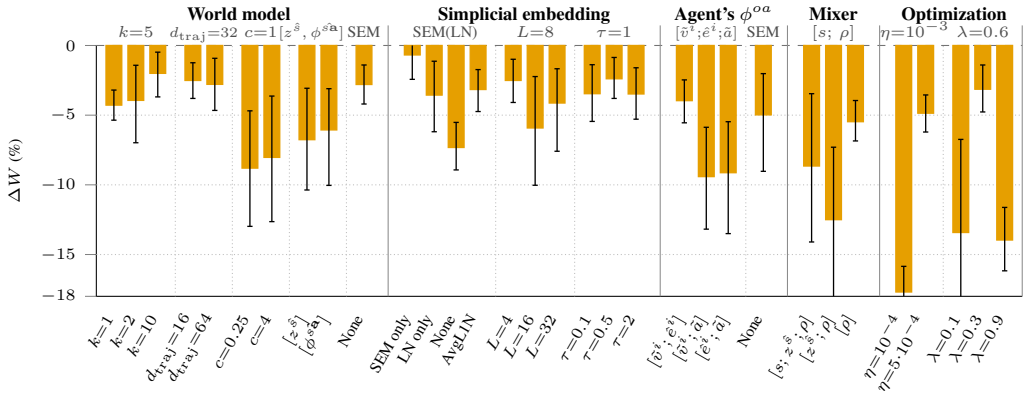

Figure 7: **Design space study on SMACv2.** ΔW shows the change in final test win rate (%) when modifying individual design choices of MSEMM relative to the full method represented by the zero line with an 80.0% win rate. The label above each group gives the full method's design choice. Performance is averaged over 3.9-4.0M steps. Error bars show 95% bootstrap confidence intervals. Details of the design choices can be found in Appx. D.

5.1 Main Results

SMAC. We evaluate MSEMM on eight SMAC maps, including five *super-hard* maps: *corridor*, *6h_vs_8z*, *27m_vs_30m*, *3s5z_vs_3s6z*, and *MMM2*, in Figure 4. MSEMM consistently outperforms eight baseline MARL approaches across all maps. Notably, on the five *super-hard* maps, MSEMM exhibits the fastest sample efficiency, reaching an 80% test win rate within 2M timesteps.

SMACv2. We compare MSEMM with seven MARL methods across six SMACv2 scenarios. While HPN-QMIX and HPN-VDN are the closest contenders, they rely on hypernetwork output layers without an imagination module. MSEMM achieves the highest overall win rate across all scenarios and exhibits sample efficiency compared to baselines within 2M timesteps, across all maps Figure 5.

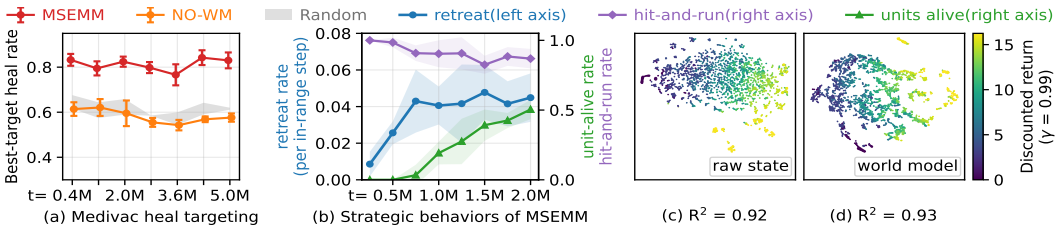

Figure 8: **Analysis of learned representations and strategic behavior. (a)** Medivac best-target heal rate on *terran_5v5* for MSEMM against NO-WM (gray band indicates random choice). **(b)** Kiting behavior of MSEMM on *6h_vs_8z* over training. **(c–d)** t-SNE visualizations of raw global states and world-model latents on *6h_vs_8z* colored by discounted return.

Multi-Agent MuJoCo. Figure 6 shows MSEMM compared against seven MARL baselines on the MAMuJoCo benchmark. We discretize the action space by dividing each joint's torque range into equal intervals. MSEMM outperforms all baselines across all four tasks. Specifically on *HalfCheetah_6x1* and *Ant_8x1*, MSEMM achieves a score $\sim 70\%$ higher than the second-best method.

5.2 Ablation and Design Space Studies

Ablation study. To evaluate each component of MSEMM, we test seven ablated variants (Figure 5): (1) **NO-PI** replaces the PI aggregation layer A with a flat network over concatenated observations; (2) **NO-AC** replaces the action-conditioned module with the per-slot output layers from HPN (Jianye et al., 2023); (3) **NO-SEM** removes the simplicial embedding entirely from the framework; (4) **NO-WM** removes the world model in MSEMM; (5) **NO-GS** conditions the mixing network only on the trajectory encoding ρ_t, omitting the global state; (6) **NO-MIX** uses a constant input for the mixing network, making mixing weights state-independent; and (7) **NO-IS** replaces each agent's latent state I^i with plain observations. Figure 5 shows that MSEMM consistently outperforms all ablation variants. This study indicates that the main components are indispensable to the framework. By unifying these key components, MSEMM achieves the best learning efficiency and outcome.

Design Choices. Figure 13 evaluates various design choices of MSEMM on SMACv2 by measuring their effect on the final test win rate (ΔW). We analyze options across five main components: the world model, simplicial embeddings (in world model and agent networks), the action-conditioned module, the mixer input, and the optimizer. Every variant in the design space is outperformed by the full MSEMM, confirming that each design component positively impacts performance.

Additional studies. Figure 8 analyzes the learned behavior and latent representations of the world model. As shown in Figure 8a, MSEMM maintains a Medivac best-target heal rate near 0.8, whereas NO-WM remains around 0.6 within the random selection baseline, confirming that effective targeting relies on the world model. Figure 8b illustrates the progressive emergence of strategic kiting behavior (retreat, hit-and-run, and unit survival) over 2M timesteps. Finally, the t-SNE projections in Figure 8c–8d show that the world-model latents predict discounted returns ($R^2 = 0.93$) nearly as accurately as the raw global states ($R^2 = 0.92$).

6 Conclusion

This paper presents MSEMM, a novel value-based MARL framework unifying structured representation learning with world model simulation. It aims to address several common challenges in MARL, including sparse and non-stationary TD learning, agents' dependency on the ordering of entities, and the exponential growth of the joint action space. The agent network of MSEMM utilizes latent states with recurrent and representation models. It deploys a PI backbone and an action-conditioned module to build simplicial state-action embeddings, ensuring that the utilities maintain PI and PE properties. A SEM-equipped world model in the framework learns joint state-action embeddings and generates simulated rollouts to augment the mixing network. Experiments across a wide variety of MARL environments show that MSEMM beats leading value-based, actor-critic, and model-based algorithms in final performance and learning efficiency. Ablation and design space studies demonstrate the positive influence of different MSEMM components.

AI USE STATEMENT

For the paper, LLMs were used to provide advice for improving the writing, debugging the implementation efficiency and verifying the correctness of our code. Generative AI tools were not used for literature retrieval and discovery, research ideation or execution, synthetic dataset generation, or proving mathematical claims.

ETHICS STATEMENT

There is no ethics concern for our work MSEMM.

REPRODUCIBILITY STATEMENT

In order to ensure reproducibility, we made use of open-source code and benchmarks. The codebase will be shared in the supplementary material. In the appendix of the paper, we state all the important hyperparameters for training MSEMM and replicating our results.

REFERENCES

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15619–15629, 2023.

Ronen Brafman and Moshe Tennenholtz. A near-optimal poly-time algorithm for learning in a class of stochastic games. *IJCAI*, pp. 734–739, 1999. ISSN 1045-0823.

Ronen Brafman and Moshe Tennenholtz. R-max–A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning. *Journal of Machine Learning Research*, 3(2):213, 2003. ISSN 15324435.

J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pp. 8234–8244, 2018.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. *arXiv preprint arXiv:2601.21306*, 2026.

Kurtland Chua, Roberto Calandra, Rowan Thomas McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Neural Information Processing Systems*, 2018.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge?, 2020. URL https://arxiv.org/abs/2011.09533.

Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, 2011.

Vladimir Egorov and Alexei Shpilman. Scalable multi-agent model-based reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 381–390, 2022.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Nicolaus Foerster, and Shimon Whiteson. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=5OjLGiJW3u.

V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011. doi: 10.1137/10080484X. URL https://epubs.siam.org/doi/10.1137/10080484X.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual Multi-Agent Policy Gradients. *AAAI*, 32, 2018. doi: 10.1609/aaai.v32i1.11794.

S. Fujimoto, P. D'Oro, A. Zhang, Y. Tian, and M. Rabbat. Towards general-purpose model-free reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 2021. ISSN 1573-7462. doi: 10.1007/s10462-021-09996-w. URL https://doi.org/10.1007/s10462-021-09996-w.

Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning, 2020. URL https://arxiv.org/abs/2004.14646.

Zhaohan Daniel Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, Michal Valko, Rémi Munos, Mohammad Gheshlaghi Azar, and Bilal Piot. Byol-explore: Exploration by bootstrapped prediction, 2022. URL https://arxiv.org/abs/2206.08332.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 6 2019b.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26091–26104, 2022.

Danijar Hafner, J. Pasukonis, Jimmy Ba, and Timothy P. Lillicrap. Mastering diverse domains through world models. *arXiv*, abs/2301.04104, 2023.

N. Hansen, H. Su, and X. Wang. Td-mpc2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

N. A. Hansen, H. Su, and X. Wang. Temporal difference learning for model predictive control. In *International Conference on Machine Learning*, pp. 8387–8406. PMLR, 2022.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. MASER: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In *PMLR*, volume 162, pp. 10041–10052, 2022.

HAO Jianye, Xiaotian Hao, Hangyu Mao, Weixun Wang, Yaodong Yang, Dong Li, Yan Zheng, and Zhen Wang. Boosting multiagent reinforcement learning via permutation invariant and permutation equivariant networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, pp. 82–94, 2016.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=EcGGFkNTxdJ.

Leonid Kuvayev and Richard S. Sutton. Model-based reinforcement learning with an approximate, learned model. In *Proceedings of the ninth Yale workshop on adaptive and learning systems*, volume 1996, pp. 101–105, 1996.

N. Lambert, B. Amos, O. Yadan, and R. Calandra. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control*, pp. 761–770. PMLR, 2020.

S. Lavoie, C. Tsirigotis, M. Schwarzer, A. Vani, M. Noukhovitch, K. Kawaguchi, and A. Courville. Simplicial embeddings in self-supervised learning and downstream classification. In *The Eleventh International Conference on Learning Representations*, 2023.

H. Lee, Y. Lee, T. Seno, D. Kim, P. Stone, and J. Choo. Hyperspherical normalization for scalable deep reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for mdps. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006. URL https://dblp.org/rec/conf/isaim/LiWL06.

Yan Li, Lingxiao Wang, Jiachen Yang, Ethan Wang, Zhaoran Wang, Tuo Zhao, and Hongyuan Zha. Permutation invariant policy optimization for mean-field multi-agent reinforcement learning: A principled approach. *arXiv preprint arXiv:2105.08268*, 2021. URL https://arxiv.org/abs/2105.08268.

H. Lin, P. Wang, J. Schneider, and G. Shi. Td-m(pc)^2: Improving temporal difference mpc through policy constraint. *arXiv preprint arXiv:2502.03550*, 2025.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 1994.

Shunyu Liu, Yihe Zhou, Jie Song, Tongya Zheng, Kaixuan Chen, Tongtian Zhu, Zunlei Feng, and Mingli Song. Contrastive identity-aware learning for multi-agent value decomposition. In *AAAI*, volume 37, 2023. doi: 10.1609/aaai.v37i10.26370.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6382–6393, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Fan-Ming Luo, Tian Xu, Hang Lai, Xiong-Hui Chen, Weinan Zhang, and Yang Yu. A survey on model-based reinforcement learning. *Science China Information Sciences*, 67(2):121101, 2024.

Chengdong Ma, Aming Li, Yali Du, Hao Dong, and Yaodong Yang. Efficient and scalable reinforcement learning for large-scale network control. *Nature Machine Intelligence*, 6(9):1006–1020, 2024.

Chirayu Nimonkar, Shlok Shah, Catherine Ji, and Benjamin Eysenbach. Self-supervised goal-reaching results in multi-agent cooperation and exploration. *arXiv preprint arXiv:2509.10656*, 2025.

J. Obando-Ceron, W. Mayor, S. Lavoie, S. Fujimoto, A. Courville, and P. S. Castro. Simplicial embeddings improve sample efficiency in actor–critic agents. In *The Fourteenth International Conference on Learning Representations*, 2026.

J. Oh, S. Singh, and H. Lee. Value prediction network. *Advances in neural information processing systems*, 30, 2017.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Kei Ota, Tomoaki Oiki, Devesh K. Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7424–7433. PMLR, 2020.

D. Palenicek, M. Lutter, J. Carvalho, and J. Peters. Diminishing return of value expansion methods in model-based reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Barna Pasztor, Ilija Bogunovic, and Andreas Krause. Efficient model-based multi-agent mean-field reinforcement learning. *arXiv:2107.04050*, 2021.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. In *NeurIPS*, 2021.

Sebastien Racaniere, Theophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *NeurIPS*, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *PMLR*, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: expanding monotonic value function factorisation. *CoRR*, abs/2006.10800, 2020a. URL `https://arxiv.org/abs/2006.10800`.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020b.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Aidan Scannell, Kalle Kujanpää, Yi Zhao, Mohammadreza Nakhaei, Arno Solin, and Joni Pajarinen. iqrl – implicitly quantized representations for sample-efficient reinforcement learning, 2024. URL https://arxiv.org/abs/2406.02696.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

M. Schwarzer, J. S. O. Ceron, A. Courville, M. G. Bellemare, R. Agarwal, and P. S. Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *International Conference on Machine Learning*, pp. 30365–30380. PMLR, 2023.

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations, 2021. URL https://arxiv.org/abs/2007.05929.

Pier Giuseppe Sessa, Maryam Kamgarpour, and Andreas Krause. Efficient model-based multi-agent reinforcement learning via optimistic equilibrium computation. In *International Conference on Machine Learning*, pp. 19580–19597. PMLR, 2022.

D. Silver, H. Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pp. 3191–3199. PMLR, 2017.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *PMLR*, pp. 5887–5896, 2019.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-Decomposition Networks For Cooperative Multi-Agent Learning. *arXiv preprint arXiv:1706.05296*, 2017. arXiv:1706.05296.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

H. P. Van Hasselt, M. Hessel, and J. Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Aravind Venugopal, Stephanie Milani, Fei Fang, and Balaraman Ravindran. Mabl: Bi-level latent-variable world model for sample-efficient multi-agent reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, pp. 1865–1873. International Foundation for Autonomous Agents and Multiagent Systems, 2024. ISBN 9798400704864.

S. Wang, S. Liu, W. Ye, J. You, and Y. Gao. Efficientzero v2: Mastering discrete and continuous control with limited data. In *International Conference on Machine Learning*, pp. 51041–51062. PMLR, 2024.

Xihuai Wang, Zhicheng Zhang, and Weinan Zhang. Model-based Multi-agent Reinforcement Learning: Recent Progress and Prospects, March 2022. URL http://arxiv.org/abs/2203.10603. arXiv:2203.10603 [cs].

Y. Wang, H. Guo, S. Wang, L. Qian, and X. Lan. Bootstrapped model predictive control. In *The Thirteenth International Conference on Learning Representations*, 2025.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28 (NeurIPS 2015)*, pp. 2746–2754, 2015.

Zifan Wu, Chao Yu, Chen Chen, Jianye Hao, and Hankz Hankui Zhuo. Models as agents: optimizing multi-step predictions of interactive local models in model-based multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10435–10443, 2023.

Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *CoRR*, 2020. URL https://arxiv.org/abs/2011.00583.

W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao. Mastering atari games with limited data. *Advances in neural information processing systems*, 34:25476–25488, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, pp. 24611–24624, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Kaiqing Zhang, Sham Kakade, Tamer Basar, and Lin Yang. Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity. In *NeurIPS*, 2020.

Shaowei Zhang, Jiahan Cao, Lei Yuan, Yang Yu, and De-Chuan Zhan. Self-motivated multi-agent exploration. In *AAMAS*, pp. 476–484, 2023.

Yi Zhao, Wenshuai Zhao, Rinu Boney, Juho Kannala, and Joni Pajarinen. Simplified temporal consistency reinforcement learning, 2023. URL https://arxiv.org/abs/2306.09466.

Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning, 2024. URL https://arxiv.org/abs/2306.13229.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024. URL http://jmlr.org/papers/v25/23-0488.html.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for multi-agent actor-critic. *ArXiv*, abs/2007.02529, 2020.

A Additional Background

A.1 Markov Games

When several agents act in the same environment, the transition and the rewards depend on the joint action, and a single-agent MDP no longer describes the process. The standard generalization is the Markov game, also called a stochastic game (Littman, 1994), a tuple $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, P, \{R^i\}_{i\in\mathcal{N}}, \gamma\rangle$ in which $\mathcal{N} = \{1, \dots, n\}$ is the set of agents, $\mathcal{S}$ the state space, $\mathcal{A}^i$ the action set of agent i and $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$ the joint action space, $P(s' \mid s, \mathbf{a})$ the transition kernel, $r^i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function of agent i, and $\gamma \in [0, 1)$ the discount factor. At every step, all agents choose their actions simultaneously given the state s_t; the joint action $\mathbf{a}_t = (a_t^1, \dots, a_t^n)$ moves the environment to $s_{t+1} \sim P(\cdot \mid s_t, \mathbf{a}_t)$ and each agent receives its own reward $R^i(s_t, \mathbf{a}_t)$. With $n = 1$ the definition reduces to an MDP.

Each agent seeks a policy $\pi^i(a^i \mid s)$ that maximizes its own expected discounted return. Writing π^{-i} for the policies of the other agents, the value of agent i under the joint policy (π^i, π^{-i}) is

$$V^i_{\pi^i,\pi^{-i}}(s) = \mathbb{E}\Big[\sum_{t\geq 0} \gamma^t \, R^i(s_t, \mathbf{a}_t) \;\Big|\; s_0 = s,\ a_t^i \sim \pi^i,\ a_t^{-i} \sim \pi^{-i} \Big],$$

which depends on π^{-i} as much as on π^i.

Two special cases matter for this paper. When all agents receive the same reward, $R^i = r$ for every i, the Markov game is fully cooperative, and the agents share one objective. When, in addition, agents do not observe the state but each draws an observation $o^i \in \Omega$ from an observation function $O(s, i)$ and acts on its observation history, the model is the Dec-POMDP of Section 3, which adds Ω and O to the cooperative Markov game and replaces the per-agent action sets by the shared set $\mathcal{A}$ used throughout the paper.

A.2 Decentralized Partially Observable MDP

A cooperative task that involves multiple agents in a partially observable environment can be formally modeled as a decentralized partially observable Markov decision process (Dec-POMDP) (Oliehoek & Amato, 2016). A Dec-POMDP is defined by a tuple $M := \langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, R, \Omega, O, \gamma\rangle$. $\mathcal{N} := \{1, \dots, n\}$ represents the set of agents, $\mathcal{S}$ denotes the state space of all agents, $\mathcal{A}$ denotes the joint action space of agents, P is the state transition function, Ω denotes the observation space, O is the observation function, R is the reward function, and $\gamma \in [0, 1)$ is the discount factor with respect to time. At every time step, each agent $i \in \mathcal{N}$ selects an action a^i from its own action space $\mathcal{A}^i$ to form the joint action $\mathbf{a} := (a^1, \dots, a^n) \in \mathcal{A}$, where $\mathcal{A} := \times_{i\in\mathcal{N}}\mathcal{A}^i$. The joint action moves the environment to the next state according to $P(s' \mid s, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, and all agents receive the same reward $r = R(s, \mathbf{a})$. Each agent i receives an observation $o^i \in \Omega$ drawn from $O(s, i) : \mathcal{S} \times \mathcal{N} \to \Omega$ because of partial observability and acts on its action-observation history, which is denoted $\tau^i \in \mathcal{T}^i := (\Omega \times \mathcal{A}^i)^*$, through a policy $\pi^i(a^i \mid \tau^i) : \mathcal{T}^i \times \mathcal{A}^i \to [0, 1]$. The learning goal is to maximize the expected return $\mathbb{E}[\mathcal{G}_t]$ by optimizing the joint policy $\boldsymbol{\pi} := (\pi^1, \dots, \pi^n)$, where $\mathcal{G}_t = \sum_{k\geq 0} \gamma^k r_{t+k}$ is the discounted return. The joint action-value function of $\boldsymbol{\pi}$ is written as $Q^{\boldsymbol{\pi}}(s_t, \mathbf{a}_t) = \mathbb{E}\big[\mathcal{G}_t \mid s_t, \mathbf{a}_t\big]$.

A.3 Entity-factored modeling

A common setting for MARL environments is entity-factored modeling (Jianye et al., 2023), in which an environment consists of m entities in total. In this set of m entities, the number of learning agents is n, and the number of non-player (or adversarial) entities is $(m - n)$.[1] States and observations are factorizable, which can be represented by entity-related features and written as $s \in \mathbb{R}^{m\times d_s}$ and $o^i \in \mathbb{R}^{m\times d_o}$, where d_s and d_o define the dimensions of features of each entity in s and in o^i, respectively. The rows of o^i consist of the agent's own row, the $(n - 1)$ rows of allied agents, and the $(m-n)$ rows of non-player (or adversarial) entities. Each agent's action space

[1] Examples of non-player entities include the enemies in combat tasks and non-agent objects in other tasks.

consists of an entity-correlated action space $\mathcal{A}_{\text{eqv}}$ and an entity-uncorrelated action space $\mathcal{A}_{\text{inv}}$, i.e., $\mathcal{A}^i := (\mathcal{A}_{\text{inv}}, \mathcal{A}_{\text{eqv}})$.[2]

A.4 Permutation invariance and permutation equivariance

Let S_k be the set of all permutations of $\{1, \dots, k\}$ for any $k \in \mathbb{N}$. A function $f : X^k \to Y^k$ is permutation-equivariant (PE) if for any $\sigma \in S_k$ and an arbitrary input matrix $X \in \mathbb{R}^{k \times d}$, $f(P_\sigma X) = P_\sigma f(X)$, where $P_\sigma \in \mathbb{R}^{k \times k}$ is the permutation matrix of σ, which acts on X by permuting its rows. Alternatively, a function $f : X^k \to Y^k$ is permutation-invariant (PI) if for any $\sigma \in S_k$ and a matrix $X \in \mathbb{R}^{k \times d}$, $f(P_\sigma X) = f(X)$, where $P_\sigma \in \mathbb{R}^{k \times k}$ is the permutation matrix of σ. Under the entity-factored modeling, we define $g = (\sigma, \sigma')$ as an arbitrary pair of bijections σ and σ', in which σ permutes the adversarial rows and σ' permutes the allied rows. We define $G \cong S_{m-n} \times S_{n-1}$ as the set of all such pairs of bijections. For any $g \in G$ and $a_i = (a_{\text{inv}}, a_{\text{eqv}}) \in \mathcal{A}^i$, $ga_i = (a_{\text{inv}}, ga_{\text{eqv}})$.

A.5 Value Decomposition and Individual-Global-Max Condition

CTDE value-based methods represent the joint action value as

$$Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{a}, s) = f\big(Q_1(\tau^1, a^1), \dots, Q_n(\tau^n, a^n);\, s\big),$$

where $\boldsymbol{\tau} = (\tau^1, \dots, \tau^n)$ is the joint action-observation history and $\mathbf{a} = (a^1, \dots, a^n)$ the joint action. Training is TD regression of Q_{tot} on replayed episodes, and each agent acts greedily on its own utility. Value decomposition fits CTDE because the agents are trained on one shared objective while each of them acts on local information alone (Son et al., 2019). For this to be sound, the greedy actions of the individual agents must together form the greedy joint action of Q_{tot}. This requirement is the individual-global-max (IGM) condition.

Definition 1 (Individual-global-max)**.** *Let $\mathcal{T}$ be the space of joint action-observation histories and $\mathcal{A}^n$ the joint action space, and let $Q_{\text{tot}} : \mathcal{T} \times \mathcal{A}^n \times \mathcal{S} \to \mathbb{R}$ be a joint action-value function. Utilities $\{Q_i : \mathcal{T}^i \times \mathcal{A} \to \mathbb{R}\}_{i=1}^n$ satisfy IGM for Q_{tot} if, for every $\boldsymbol{\tau} \in \mathcal{T}$ and $s \in \mathcal{S}$,*

$$\arg\max_{\mathbf{a} \in \mathcal{A}^n} Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{a}, s) = \Big(\arg\max_{a^1 \in \mathcal{A}} Q_1(\tau^1, a^1),\ \dots,\ \arg\max_{a^n \in \mathcal{A}} Q_n(\tau^n, a^n) \Big).$$

When the condition holds, Q_{tot} is said to be decomposable by $\{Q_i\}_{i=1}^n$.

IGM states that a cooperative task can be solved in a decentralized manner as long as the local and the joint action values are consistent in this sense (Sunehag et al., 2017; Rashid et al., 2018; Son et al., 2019).

Different methods guarantee IGM through different sufficient conditions. VDN (Sunehag et al., 2017) uses *additivity*,

$$Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{a}) = \sum_{i=1}^{n} Q_i(\tau^i, a^i),$$

and factorizes the joint value as a sum of the utilities. QMIX (Rashid et al., 2018) uses *monotonicity*,

$$\frac{\partial Q_{\text{tot}}(\boldsymbol{\tau}, \mathbf{a}, s)}{\partial Q_i(\tau^i, a^i)} \geq 0 \quad \forall i \in \{1, \dots, n\},$$

and proves that IGM holds under this assumption. It implements f as a mixing network whose weights are generated from s by hypernetworks and constrained to be non-negative (Rashid et al., 2020b), with VDN as the special case $f = \sum_i Q_i$. QTRAN (Son et al., 2019) proposes to find utilities $\{Q_i\}_{i=1}^n$ that factorize the original joint action-value function Q_{jt} and to transform Q_{jt} into a new value function Q'_{jt} that shares the same optimal joint action with Q_{jt}. The sufficient condition under which $\{Q_i\}_{i=1}^n$ satisfy IGM is stated in Theorem 1 of (Son et al., 2019).

[2] For each action $a_{\text{eqv}} \in \mathcal{A}_{\text{eqv}}$, there is a correspondence between an entity and a_{eqv}. In contrast, $\mathcal{A}_{\text{inv}}$ includes actions that involve no other entity.

B PROOFS OF PROPOSITIONS 1 AND 2

Proposition 1 (Guaranteed PI through the backbone). Let $g \in G$ act on a history $(I_1^i, \ldots, I_t^i)$ by operating on every element through its entity components. If A is permutation-invariant, i.e. $A(g(I)) = A(I)$ for all $g \in G$ and all I, then for any function $\bar{B}$, the backbone output $\tilde{v}_t^i = \bar{B}\big(A(I_1^i), \ldots, A(I_t^i)\big)$ is permutation-invariant, i.e. it takes the same value on $(g(I_1^i), \ldots, g(I_t^i))$ as on $(I_1^i, \ldots, I_t^i)$ for every $g \in G$.

Proof of Proposition 1. Let $\mathbf{I} := (I_1^i, \ldots, I_t^i)$. Given two different inputs $\mathbf{I}$ and $g(\mathbf{I})$, for all $\mathbf{I}$ and for all $g \in G$, we have $A(\mathbf{I}) = A(g(\mathbf{I}))$ because of the permutation invariance of A. Therefore, for any function $\bar{B}$, $(\bar{B} \circ A)(g(\mathbf{I})) = (\bar{B} \circ A)(\mathbf{I})$, which is PI by definition. $\bar{B}$ receives the same inputs on the permuted history as those on the original history, and returns the same value. □

Proposition 2 (PI and PE of the utilities). Let $g \in G$ act on the history $(I_1^i, \ldots, I_t^i)$ as in Proposition 1 and on $\mathcal{A}$ as in Section 3. Suppose the action-conditioned module receives (i) a permutation-invariant backbone output $\tilde{v}_t^i$; (ii) a function $\hat{e}^i$ that satisfies $\hat{e}^i(a_{\text{eqv}}(j)) = \text{Enc}(I_t^i[j])$ for all $a_{\text{eqv}}(j) \in \mathcal{A}_{\text{eqv}}$ and $\hat{e}^i(a) = \hat{e}_f$ for all $a \in \mathcal{A}_{\text{inv}}$; and (iii) an embedding $\tilde{a}(a)$ that is constant on $\mathcal{A}_{\text{eqv}}$. Then the utility satisfies PI on $\mathcal{A}_{\text{inv}}$ and PE on $\mathcal{A}_{\text{eqv}}$, i.e., $Q_i(g(I_t^i), g \cdot a) = Q_i(I_t^i, a)$ for all $g \in G$ and $a \in \mathcal{A}$, where $Q_i(g(I_t^i), \cdot)$ is the utility computed on the permuted history.

Proof of Proposition 2. Let $g = (\sigma, \sigma') \in G$, drop the indices i and t, and denote $I^g = g(I)$. Let j be the entity targeted by a, and it is correct to denote $a = a_{\text{eqv}}(j)$ if $a \in \mathcal{A}_{\text{eqv}}$. By the definition of σ and σ' in Section 3, applying g on I moves j to slot $\sigma(j)$ and slot $\sigma(j)$ of I^g returns exactly the entity that was in slot j of I. Therefore, $I^g[\sigma(j)] = I[j]$, i.e., $I^g[j] = I[\sigma^{-1}(j)]$, for the adversarial components. Let $\tilde{v}^g$ be the backbone output on the permuted history, and $\tilde{v}^g = \tilde{v}$ by Assumption (i). The action-conditioned module is one fixed map applied to the tuple $(\tilde{v}, \hat{e}(a), \tilde{a}(a))$, so it suffices to compare the tuple.

Invariance on $\mathcal{A}_{\text{inv}}$. For any $a \in \mathcal{A}_{\text{inv}}$, $\hat{e}(a) = \hat{e}_f$ by Assumption (ii), and $\tilde{a}(a)$ is a function of a alone. Therefore, neither of them depends on I. The tuple $(\tilde{v}, \hat{e}(a), \tilde{a}(a))$ is the same on I^g and on I, and therefore $Q(I^g, a) = Q(I, a)$.

Equivariance on $\mathcal{A}_{\text{eqv}}$. Let $a = a_{\text{eqv}}(j)$. By Assumption (ii), the embedding of a on I^g is $\text{Enc}(I^g[j]) = \text{Enc}(I[\sigma^{-1}(j)])$, which, by (ii), is the embedding of $a_{\text{eqv}}(\sigma^{-1}(j))$ on I. By Assumption (iii), $\tilde{a}$ takes the same value on both actions. The tuple on $(I^g, a_{\text{eqv}}(j))$ is therefore equal to the tuple on $(I, a_{\text{eqv}}(\sigma^{-1}(j)))$. For every j, we have

$$Q\big(g(I), a_{\text{eqv}}(j)\big) = Q\big(I, a_{\text{eqv}}(\sigma^{-1}(j))\big).$$

Substituting $j = \sigma(j')$ gives

$$Q\big(g(I), a_{\text{eqv}}(\sigma(j'))\big) = Q\big(I, a_{\text{eqv}}(j')\big),$$

which satisfies permutation equivariance because $g \cdot a_{\text{eqv}}(j') = a_{\text{eqv}}(\sigma(j'))$.

In both cases σ' enters only through $\tilde{v}$, which is invariant by Assumption (i). □

C Overview of the Benchmark Environments

C.1 SMACv1

SMAC (Samvelyan et al., 2019), also referred to as SMACv1, and its successor SMACv2 (Ellis et al., 2023) are micromanagement benchmarks built on the real-time strategy game StarCraft II. Each allied unit is controlled by one agent that observes only the units within its sight range and chooses, at every step, to move, to stop, or to attack one visible enemy; the opposing units are controlled by the built-in game AI. The team receives a shared reward for damage dealt and units killed, with a bonus for winning the battle, and the standard evaluation metric is the test win rate.

The SMACv1 scenarios differ in the number and the types of the units on each side, and together they cover a range of coordination problems. Symmetric scenarios pit identical teams against each other, from three Marines per side (*3m*) up to 25 (*25m*) or a mixed team of Stalkers and Zealots (*2s3z*, *3s5z*); the built-in AI is a strong opponent even then, because it focuses fire well. Asymmetric scenarios give the enemy more or stronger units and can only be won with a specific tactic. In *5m_vs_6m*, *8m_vs_9m*, *10m_vs_11m* and *27m_vs_30m*, Marines outnumbered by one to three units have to concentrate their fire. In *3s_vs_3z*, *3s_vs_4z* and *3s_vs_5z*, ranged Stalkers face melee Zealots and win only by kiting, alternating attack and retreat, which delays the reward. In *2c_vs_64zg*, two Colossi face 64 Zerglings, the largest enemy set in the benchmark, and must use their splash damage and the terrain. In *6h_vs_8z*, six Hydralisks face eight Zealots, and in *corridor*, six Zealots face 24 Zerglings and survive only by holding a narrow passage so that they are not surrounded. Heterogeneous scenarios add unit types with different roles: *MMM* and *MMM2* combine Marines, Marauders and a Medivac, a unit that heals allies instead of attacking, and *1c3s5z* combines a Colossus with Stalkers and Zealots. Two scenarios are built around Banelings, suicide units that explode on contact: in *bane_vs_bane*, each side has 20 Zerglings and 4 Banelings, and in *so_many_baneling*, seven Zealots face 32 Banelings and have to spread out so that every explosion hits as few of them as possible. The benchmark groups its scenarios into easy, hard and super-hard ones; the super-hard group consists of *MMM2*, *3s5z_vs_3s6z*, *27m_vs_30m*, *6h_vs_8z* and *corridor*, on which the standard value-decomposition methods learn slowly or not at all.

C.2 SMACv2

SMACv2 addresses a weakness of SMACv1: its scenarios are deterministic enough for open-loop policies to succeed. It keeps the game and the action space but procedurally generates each episode through three changes. (1) *Randomized start positions.* Two spawn patterns replace the fixed openings: *surrounded*, in which the allied units start in the center and the enemies on several flanks, and *reflect_position*, in which the allied units are placed uniformly at random and the enemies are placed at their mirror images about the center of the map. (2) *Probabilistic unit types.* Instead of a fixed army, the unit types of both teams are sampled from a race-specific distribution over three unit types per race (Protoss, Terran and Zerg), so team compositions vary from episode to episode. (3) *Conical partial observability.* Sight and attack ranges can be restricted to a forward-facing cone, closer to the game's own mechanics, which requires the agents to gather information actively. Training and test episodes are drawn from the same generative distribution, but no two episodes share a layout, so there is no fixed opening to memorize. We use the three races at two team sizes, 5 allied against 5 enemy units (*protoss_5v5*, *terran_5v5*, *zerg_5v5*) and 10 against 10 (*protoss_10v10*, *terran_10v10*, *zerg_10v10*).

C.3 Multi-Agent MuJoCo

Multi-Agent MuJoCo (MaMuJoCo) (Peng et al., 2021) is a benchmark for cooperative multi-agent continuous control built on the single-agent MuJoCo locomotion tasks. A robot is partitioned into disjoint groups of joints, and each group is an agent that controls the motors of its joints. The agents have to coordinate to make the whole robot move, and each agent observes only its own joints together with the joints within a fixed distance in the robot's kinematic graph, which makes the task partially observable. Figure 9 shows the partitions provided by the benchmark. A partition is denoted by the name of the robot followed by the number of agents and the number of joints per agent. We use four configurations: *Hopper_3x1* (three agents, one joint each; configuration B in Figure 9), *HalfCheetah_6x1* (six agents, one joint each; configuration D), *Walker2d_6x1* (six agents,

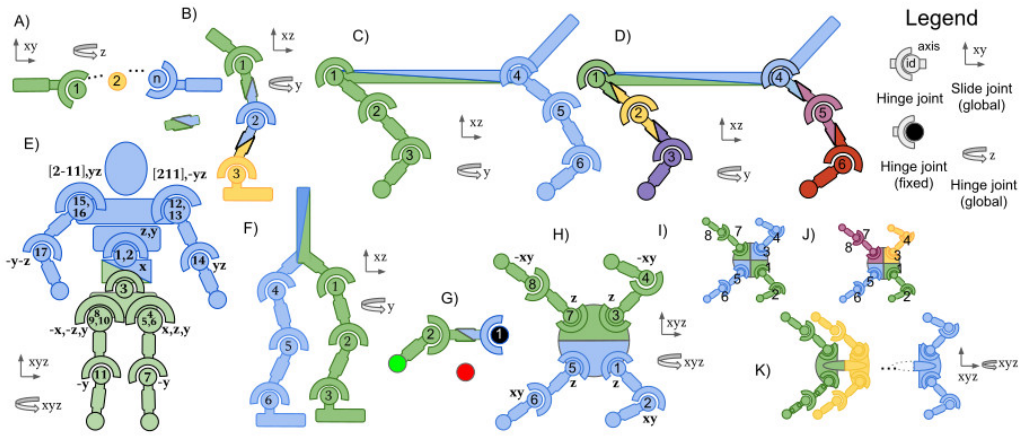

Figure 9: Agent partitions of the robots in Multi-Agent MuJoCo. A) Many-Agent Swimmer; B) Three-Agent Hopper; C) Two-Agent Half Cheetah; D) Six-Agent Half Cheetah; E) Two-Agent Humanoid; F) Two-Agent Walker2d; G) Two-Agent Reacher; H) Two-Agent Ant; I) Diagonal Two-Agent Ant; J) Four-Agent Ant; K) Many-Agent Ant. Figure reproduced from the FACMAC paper (Peng et al., 2021).

one joint each) and *Ant_8x1* (eight agents, one joint each), each agent observing joints up to distance two in the kinematic graph.

Since our agents select discrete actions, we discretize the action space of MaMuJoCo. The continuous torque range of each joint is divided into N equally spaced levels, with N an adjustable number, and each agent picks one level for each of its joints at every step. We use $N = 30$ in all four configurations.

D SUPPLEMENTARY RESULTS

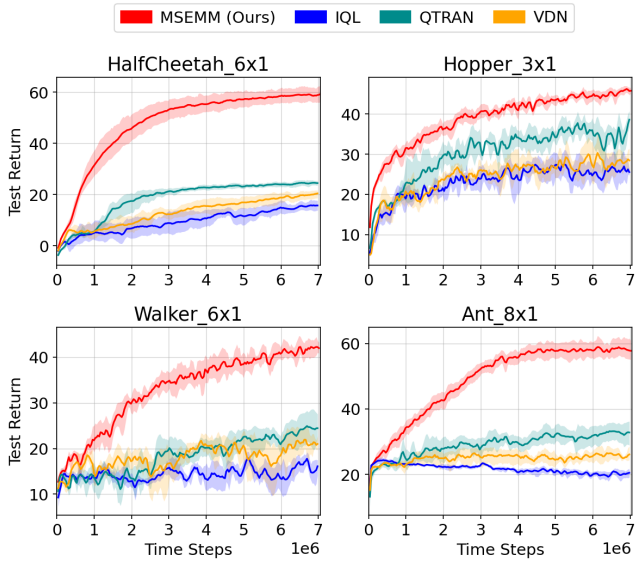

Figure 10: Test return on the four Multi-Agent MuJoCo tasks, comparing MSEMM with IQL, QTRAN and VDN, with a budget of 7M steps. Each curve is the mean over five seeds of runs. The shaded region is the 95% bootstrap confidence interval of the mean.

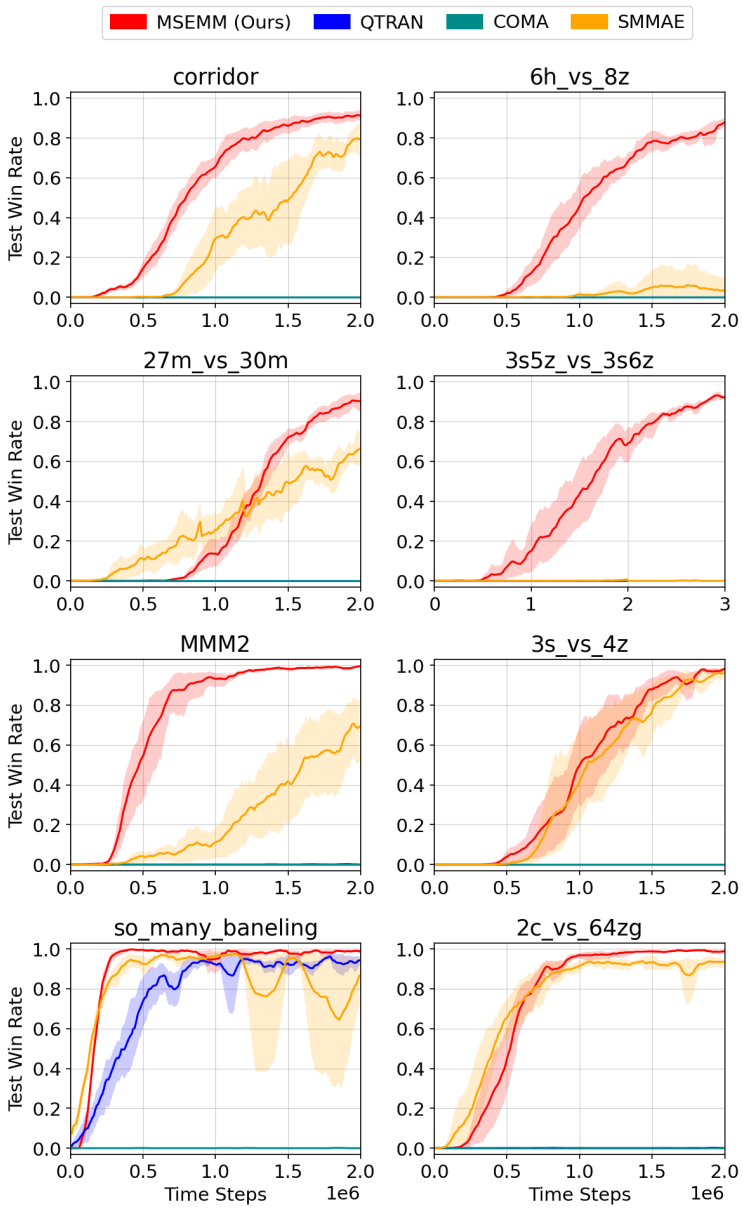

Figure 11: Test win rate on the eight SMAC maps of the main results, comparing MSEMM with QTRAN, COMA and SMMAE. The budget is 2M steps, 3M on *3s5z_vs_3s6z*. Each curve is the mean over five seeds of runs. The shaded region is the 95% bootstrap confidence interval of the mean.

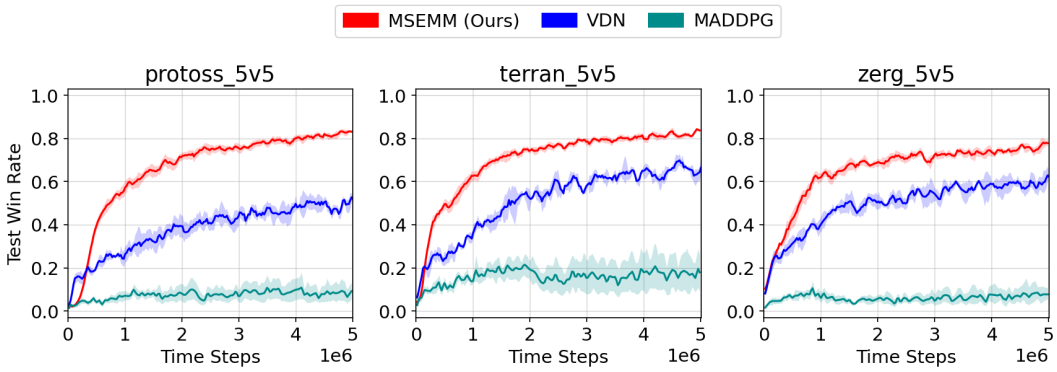

Figure 12: Test win rate on the three SMACv2 scenarios, comparing MSEMM with VDN and MADDPG, with a budget of 5M steps. Each curve is the mean over five seeds of runs. The shaded region is the 95% bootstrap confidence interval of the mean.

Table 1: Final test return on the Multi-Agent MuJoCo tasks of the main results. A run's final level is the mean of its evaluations over its last 100k training steps within the plotted budget; each cell gives the mean over runs followed, in gray, by its 95% bootstrap confidence interval. Best mean per task in bold.

Task	MSEMM (Ours)	QMIX-CIA	QPLEX-CIA	MASER	SMMAE	QMIX
HalfCheetah_6x1	**58.9** [55.9, 62.2]	−2.0 [−2.7, −1.4]	−9.4 [−11.3, −7.5]	−3.3 [−3.6, −2.9]	−3.4 [−4.0, −2.8]	35.6 [29.6, 41.6]
Hopper_3x1	**45.8** [45.1, 46.5]	4.6 [3.9, 5.6]	10.9 [10.1, 11.9]	4.8 [2.8, 6.8]	10.7 [8.6, 12.9]	23.6 [20.1, 27.5]
Walker_6x1	**42.1** [40.9, 43.5]	10.8 [10.2, 11.4]	6.9 [1.7, 10.1]	6.3 [1.5, 11.3]	9.3 [6.3, 12.0]	26.5 [22.2, 30.3]
Ant_8x1	**57.8** [55.1, 61.0]	22.3 [21.7, 23.0]	−9.6 [−19.5, −1.4]	15.9 [14.6, 17.3]	23.0 [22.2, 23.7]	33.6 [30.8, 35.9]

Task	MADDPG	MAPPO	IQL	QTRAN	VDN
HalfCheetah_6x1	34.6 [25.6, 43.7]	1.5 [−0.6, 3.2]	15.8 [13.8, 17.7]	24.5 [23.6, 25.5]	20.4 [18.7, 21.7]
Hopper_3x1	4.3 [0.8, 9.0]	9.3 [4.8, 14.8]	25.7 [24.9, 27.1]	37.7 [36.1, 40.2]	28.6 [25.7, 31.1]
Walker_6x1	1.4 [−0.9, 5.8]	17.5 [16.2, 18.5]	15.4 [12.6, 17.5]	24.3 [21.1, 27.7]	20.8 [19.8, 21.9]
Ant_8x1	19.0 [17.8, 21.2]	19.8 [13.6, 23.3]	20.4 [18.9, 21.6]	32.8 [29.1, 36.1]	25.9 [24.2, 28.1]

Table 2: Final test win rate on the SMACv2 5v5 scenarios of the main results. A run's final level is the mean of its evaluations over its last 100k training steps within the plotted budget; each cell gives the mean over runs followed, in gray, by its 95% bootstrap confidence interval. Best mean per scenario in bold.

Scenario	MSEMM (Ours)	SET-QMIX	HPN-QMIX	HPN-VDN	QMIX
protoss_5v5	**0.83** [0.83, 0.84]	0.56 [0.53, 0.58]	0.73 [0.64, 0.80]	0.66 [0.61, 0.72]	0.65 [0.64, 0.67]
terran_5v5	**0.84** [0.83, 0.85]	0.56 [0.54, 0.60]	0.74 [0.70, 0.78]	0.72 [0.68, 0.77]	0.66 [0.58, 0.73]
zerg_5v5	**0.78** [0.77, 0.79]	0.62 [0.59, 0.64]	0.67 [0.58, 0.75]	0.66 [0.60, 0.71]	0.61 [0.54, 0.67]

Scenario	MAPPO	HAA2C	HAPPO	VDN	MADDPG
protoss_5v5	0.36 [0.34, 0.38]	0.06 [0.02, 0.09]	0.29 [0.23, 0.35]	0.51 [0.48, 0.55]	0.09 [0.05, 0.13]
terran_5v5	0.45 [0.42, 0.48]	0.11 [0.06, 0.16]	0.31 [0.25, 0.37]	0.63 [0.61, 0.65]	0.18 [0.11, 0.25]
zerg_5v5	0.26 [0.19, 0.33]	0.04 [0.02, 0.05]	0.25 [0.21, 0.30]	0.61 [0.57, 0.65]	0.08 [0.06, 0.10]

Table 3: Final test win rate on the SMACv2 10v10 scenarios of the main results. A run's final level is the mean of its evaluations over its last 100k training steps within the plotted budget; each cell gives the mean over runs followed, in gray, by its 95% bootstrap confidence interval. Best mean per scenario in bold.

Scenario	MSEMM (Ours)	SET-QMIX	HPN-QMIX	HPN-VDN
protoss_10v10	**0.85** [0.84, 0.87]	0.56 [0.54, 0.58]	0.72 [0.62, 0.81]	0.61 [0.42, 0.76]
terran_10v10	**0.90** [0.90, 0.91]	0.50 [0.33, 0.60]	0.78 [0.66, 0.85]	0.84 [0.83, 0.86]
zerg_10v10	**0.83** [0.82, 0.84]	0.59 [0.42, 0.68]	0.75 [0.72, 0.79]	0.72 [0.65, 0.80]

Scenario	QMIX	MAPPO	HAA2C	HAPPO
protoss_10v10	0.61 [0.60, 0.62]	0.26 [0.20, 0.33]	0.04 [0.01, 0.07]	0.15 [0.10, 0.22]
terran_10v10	0.77 [0.75, 0.78]	0.21 [0.19, 0.24]	0.02 [0.01, 0.03]	0.14 [0.10, 0.20]
zerg_10v10	0.71 [0.68, 0.73]	0.22 [0.14, 0.28]	0.04 [0.02, 0.07]	0.11 [0.07, 0.16]

Table 4: Final test win rate on the SMAC maps of the main results. A run's final level is the mean of its evaluations over its last 100k training steps within the plotted budget; each cell gives the mean over runs followed, in gray, by its 95% bootstrap confidence interval. Best mean per map in bold.

Map	MSEMM (Ours)	HAA2C	HAPPO	MASER	QMIX-CIA	QPLEX-CIA
corridor	**0.91** [0.89, 0.93]	0.00 [0.00, 0.00]	0.57 [0.18, 0.81]	0.00 [0.00, 0.00]	0.13 [0.00, 0.36]	0.00 [0.00, 0.00]
6h_vs_8z	**0.86** [0.84, 0.89]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.12 [0.01, 0.23]
27m_vs_30m	**0.91** [0.88, 0.93]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.22 [0.00, 0.63]	0.12 [0.10, 0.14]	0.17 [0.08, 0.26]
3s5z_vs_3s6z	**0.93** [0.91, 0.95]	0.00 [0.00, 0.00]	0.15 [0.10, 0.21]	0.01 [0.00, 0.01]	0.01 [0.00, 0.01]	0.09 [0.03, 0.15]
MMM2	**0.99** [0.98, 1.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.74 [0.65, 0.79]	0.47 [0.17, 0.76]	0.09 [0.02, 0.17]
3s_vs_4z	**0.97** [0.96, 0.99]	0.00 [0.00, 0.00]	0.36 [0.00, 0.73]	0.95 [0.92, 0.98]	0.85 [0.73, 0.95]	0.84 [0.74, 0.93]
so_many_baneling	**0.99** [0.98, 1.00]	0.79 [0.57, 0.94]	0.94 [0.92, 0.95]	0.83 [0.73, 0.91]	0.98 [0.97, 0.98]	0.94 [0.92, 0.97]
2c_vs_64zg	**0.99** [0.98, 0.99]	0.43 [0.23, 0.61]	0.71 [0.58, 0.86]	0.91 [0.87, 0.95]	0.89 [0.87, 0.92]	0.88 [0.83, 0.93]

Map	MABL	MAG	MAMBA	QTRAN	COMA	SMMAE
corridor	0.09 [0.00, 0.21]	0.19 [0.10, 0.32]	0.26 [0.13, 0.35]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.79 [0.74, 0.85]
6h_vs_8z	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.04 [0.00, 0.11]
27m_vs_30m	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.66 [0.58, 0.75]
3s5z_vs_3s6z	0.04 [0.00, 0.08]	0.45 [0.37, 0.54]	0.10 [0.04, 0.18]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.01 [0.00, 0.01]
MMM2	0.00 [0.00, 0.00]	0.07 [0.03, 0.10]	0.06 [0.01, 0.11]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.71 [0.55, 0.84]
3s_vs_4z	0.26 [0.06, 0.60]	0.63 [0.50, 0.78]	0.69 [0.56, 0.83]	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	0.96 [0.93, 0.98]
so_many_baneling	0.04 [0.00, 0.09]	0.76 [0.65, 0.86]	0.73 [0.62, 0.84]	0.93 [0.89, 0.96]	0.00 [0.00, 0.00]	0.87 [0.73, 0.96]
2c_vs_64zg	0.09 [0.04, 0.15]	0.13 [0.04, 0.22]	0.05 [0.00, 0.10]	0.00 [0.00, 0.01]	0.00 [0.00, 0.00]	0.93 [0.91, 0.96]

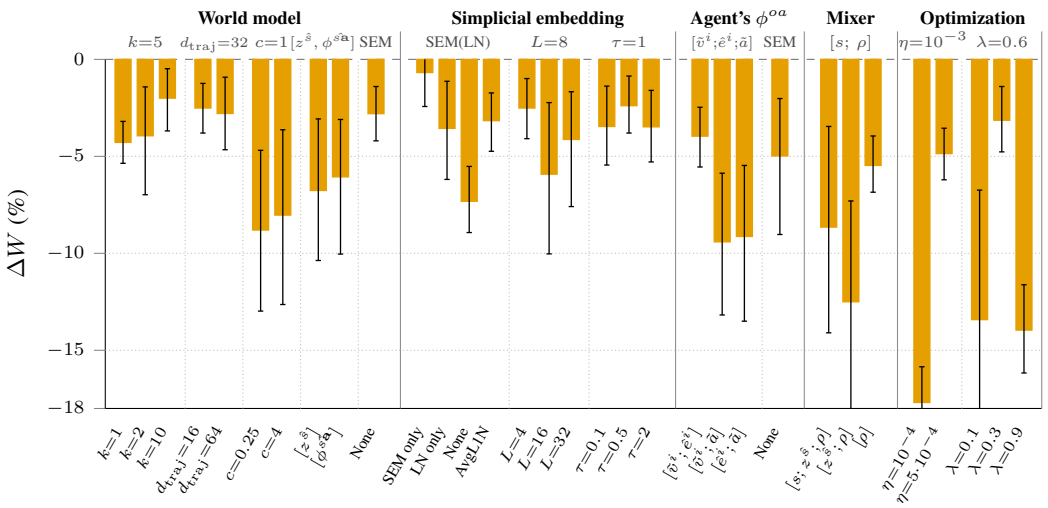

Figure 13: Design space study: change in final test win rate ΔW (%) when one design choice of MSEMM is modified. Each win rate is the mean over 3.9–4.0M steps. Error bars denote the 95% bootstrap confidence interval, truncated at the axis range. Above each group of bars is the full method's value of that design choice (the zero line). Summary width d_{traj} represents the dimension of the encoding of the world model rollouts, ρ, which the mixing network receives alongside the global state. It sets how much of the imagined future the mixer can see. Simplices L: in the action-conditioned module, each candidate embedding is split into L equal groups, and a softmax is applied within each group. This is the simplicial embedding (SEM). The temperature τ scales the embedding before each group's softmax, i.e., $\text{softmax}(z/\tau)$. A small τ, such as 0.1, makes each group nearly one-hot. A large τ, such as 2, makes it flatter and closer to uniform. TD(λ) sets how the value targets mix one-step bootstrapped targets with longer multi-step returns. $\lambda = 0$ uses one-step targets only. World model weight c: the value set for the weights of the loss functions, $c = c_{\text{LS}} = c_{\text{ALS}} = c_{\text{dyn}} = c_{\text{rew}}$.

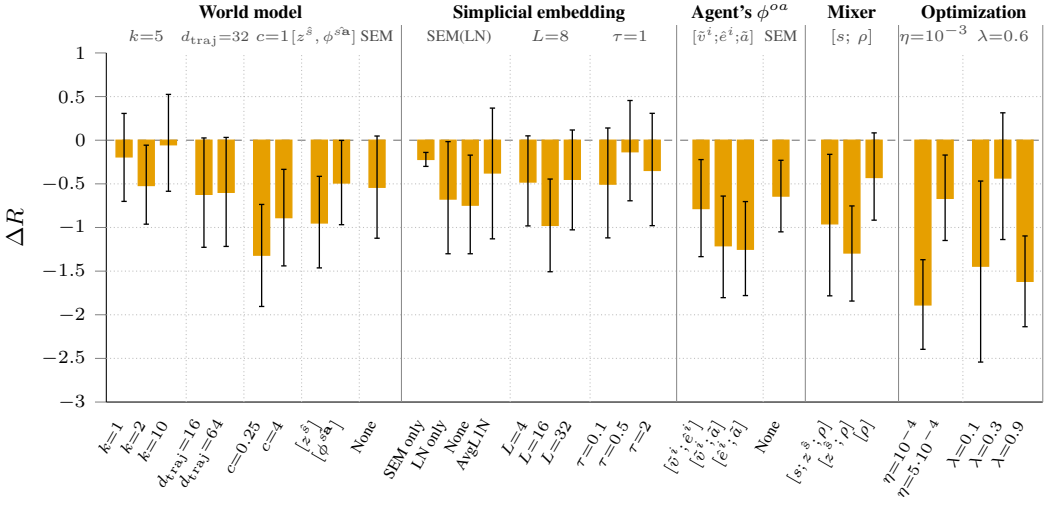

Figure 14: Design space study: change in final test return ΔR when one design choice of MSEMM is changed. Each return is the mean over 3.9–4.0M steps. Error bars stand for the 95% bootstrap confidence interval. Above each group of bars sits the full method's value of that design choice (the zero line).

Table 5: Design space study on SMACv2. ΔW: change in final test win rate (%) relative to MSEMM, with a 95% bootstrap confidence interval in brackets. Each win rate is the mean over 3.9–4.0M steps. Shading marks losses beyond the 5-point resolution threshold: $(-15, -5]$, $(-30, -15]$, ≤ -30. Design choices are described in the caption of Figure 13.

Design choice	Variant	ΔW (%) [95% CI]
MSEMM (full method)		$W = 80.0\%$
World model		
horizon k (5)	1	-4.3 $[-5.4, -3.2]$
	2	-3.9 $[-7.0, -1.4]$
	10	-2.0 $[-3.7, -0.5]$
summary width d_{traj} (32)	16	-2.5 $[-3.8, -1.2]$
	64	-2.8 $[-4.7, -0.9]$
world model weight c (1)	0.25	-8.8 $[-13.0, -4.7]$
	4	-8.0 $[-12.6, -3.6]$
latent states ($[z^{\hat{s}}, \phi^{\hat{s}\mathbf{a}}]$)	$[z^{\hat{s}}]$	-6.8 $[-10.4, -3.1]$
	$[\phi^{\hat{s}\mathbf{a}}]$	-6.1 $[-10.0, -3.1]$
normalization (SEM(LN))	None	-2.8 $[-4.2, -1.4]$
Simplicial embedding (changes applied to both the world model and the agent networks)		
normalization form (SEM(LN))	SEM only	-0.7 $[-2.4, +1.2]$
	LN only	-3.6 $[-6.2, -1.1]$
	None	-7.3 $[-8.9, -5.5]$
	AvgL1Norm	-3.2 $[-4.7, -1.7]$
simplices L (8)	2	-2.5 $[-4.5, -0.6]$
	4	-2.5 $[-4.1, -1.0]$
	16	-5.9 $[-10.0, -2.2]$
	32	-4.1 $[-7.6, -1.7]$
temperature τ (1)	0.1	-3.5 $[-5.5, -1.4]$
	0.5	-2.4 $[-3.8, -0.9]$
	2	-3.5 $[-5.3, -1.6]$
Agent's ϕ^{oa}		
ϕ^{oa} input ($[\tilde{v}_t^i; \hat{e}^i(a); \tilde{a}(a)]$)	$[\tilde{v}_t^i; \hat{e}^i(a)]$	-4.0 $[-5.5, -2.5]$
	$[\tilde{v}_t^i; \tilde{a}(a)]$	-9.4 $[-13.2, -5.9]$
	$[\hat{e}^i(a); \tilde{a}(a)]$	-9.1 $[-13.5, -5.5]$
normalization (SEM(LN))	None	-5.0 $[-9.0, -2.0]$
Mixer		
mixer input ($[s; \rho]$)	$[s; z^{\hat{s}}; \rho]$	-8.6 $[-14.1, -3.5]$
	$[z^{\hat{s}}; \rho]$	-12.5 $[-18.5, -7.3]$
	$[\rho]$	-5.5 $[-6.9, -4.0]$
Optimization		
learning rate η (10^{-3})	10^{-4}	-17.7 $[-19.6, -15.8]$
	$5{\cdot}10^{-4}$	-4.9 $[-6.2, -3.5]$
	10^{-2}	-23.7 $[-31.4, -18.2]$
TD(λ) (0.6)	0.1	-13.4 $[-20.7, -6.7]$
	0.3	-3.1 $[-4.8, -1.4]$
	0.9	-14.0 $[-16.2, -11.6]$

Table 6: Design space study on SMACv2. ΔR: change in final test return relative to MSEMM, with a 95% bootstrap confidence interval in brackets. Each return is the mean over 3.9–4.0M steps. Shading marks losses beyond the 0.93-point resolution threshold: $(-2.79, -0.93]$, $(-5.58, -2.79]$, ≤ -5.58.

Design choice	Variant	ΔR [95% CI]
MSEMM (full method)		$R = 18.60$
World model		
horizon k (5)	1	-0.19 $[-0.70, +0.31]$
	2	-0.52 $[-0.96, -0.06]$
	10	-0.05 $[-0.58, +0.53]$
summary width d_{traj} (32)	16	-0.62 $[-1.23, +0.03]$
	64	-0.60 $[-1.22, +0.03]$
world model weight c (1)	0.25	-1.32 $[-1.91, -0.74]$
	4	-0.89 $[-1.44, -0.33]$
latent states ($[z^{\hat{s}}, \phi^{\hat{sa}}]$)	$[z^{\hat{s}}]$	-0.95 $[-1.46, -0.41]$
	$[\phi^{\hat{sa}}]$	-0.49 $[-0.97, -0.00]$
normalization (SEM(LN))	None	-0.54 $[-1.12, +0.05]$
Simplicial embedding (changes applied to both the world model and the agent networks)		
normalization form (SEM(LN))	SEM only	-0.22 $[-0.30, -0.14]$
	LN only	-0.67 $[-1.30, -0.02]$
	None	-0.74 $[-1.30, -0.17]$
	AvgL1Norm	-0.38 $[-1.13, +0.37]$
simplices L (8)	2	-0.54 $[-1.17, +0.12]$
	4	-0.48 $[-0.98, +0.05]$
	16	-0.98 $[-1.51, -0.45]$
	32	-0.45 $[-1.03, +0.12]$
temperature τ (1)	0.1	-0.50 $[-1.12, +0.14]$
	0.5	-0.13 $[-0.69, +0.46]$
	2	-0.35 $[-0.98, +0.31]$
Agent's ϕ^{oa}		
ϕ^{oa} input ($[\tilde{v}_t^i;\ \hat{e}^i(a);\ \tilde{a}(a)]$)	$[\tilde{v}_t^i;\ \hat{e}^i(a)]$	-0.78 $[-1.33, -0.22]$
	$[\tilde{v}_t^i;\ \tilde{a}(a)]$	-1.21 $[-1.80, -0.64]$
	$[\hat{e}^i(a);\ \tilde{a}(a)]$	-1.25 $[-1.78, -0.70]$
normalization (SEM(LN))	None	-0.64 $[-1.05, -0.23]$
Mixer		
mixer input ($[s;\ \rho]$)	$[s;\ z^{\hat{s}};\ \rho]$	-0.96 $[-1.78, -0.16]$
	$[z^{\hat{s}};\ \rho]$	-1.29 $[-1.84, -0.75]$
	$[\rho]$	-0.43 $[-0.92, +0.08]$
Optimization		
learning rate η (10^{-3})	10^{-4}	-1.89 $[-2.40, -1.37]$
	$5{\cdot}10^{-4}$	-0.67 $[-1.15, -0.17]$
	10^{-2}	-3.79 $[-5.79, -2.47]$
TD(λ) (0.6)	0.1	-1.44 $[-2.54, -0.47]$
	0.3	-0.43 $[-1.14, +0.31]$
	0.9	-1.62 $[-2.14, -1.10]$

E EXPERIMENTAL SETTINGS

In Table 7 and Table 9, we show the hyperparameter settings for MSEMM and the environmental setup for SMACv2, respectively. The probability distributions of randomized units by race are displayed in Table 8. The settings for Multi-Agent MuJoCo are exhibited in Table 10. To make clear graphs for Multi-Agent MuJoCo, we scale the returns for the four tasks in MAMuJoCo. We have specified this in Table 10. The hyperparameters of the other MARL baselines are consistent with their official implementations. Our implementation for MSEMM is based on PyTorch.

Table 7: The setup of hyperparameters for the MSEMM method.

Name	**Value**
Action selector	Epsilon-greedy
Batch size	128
Buffer size	5000
Number of environments to run in parallel	8
Learning rate	1×10^{-3}
Discount hyperparameter γ	0.99
Optimizer	Adam
Number of hypernetwork layers	2
Runner	Parallel runner
Starting value of exploration rate	1
Finishing value of exploration rate	0.05
Number of time steps after which the target networks are updated	200
Rollout horizon	5
Number of time steps between test intervals	20000
Number of episodes for testing every time	32
Dimension of the hidden state for agent network	64
Latent dimension of the state embedding	64
Latent dimension of the joint state-action embedding	64
Latent dimension of the trajectory embedding	32
Dimension of the embedding layer for mixing network	32
Dimension of the embedding layer for hypernetwork	64

Table 8: The generation probabilities of different unit types under the three races: Protoss, Terran, and Zerg.

Race	Unit Type	Generation Probability
Protoss	Stalker	0.45
	Zealot	0.45
	Colossus	0.1
Terran	Marauder	0.45
	Marine	0.45
	Medivac	0.1
Zerg	Zergling	0.45
	Hydralisk	0.45
	Baneling	0.1

Table 9: The default settings for SMAC and SMACv2 environments.

Description	**Value**
Whether to consider episodes continuing or finished after the time limit is reached	False
Whether agents receive the last actions of all units within the sight range as part of observations	False
Whether agents receive pathing values within the sight range as part of observations	False
Whether to log messages about observations, states, actions, and rewards	False
Whether to use a combination of agents' observations as the global state	False
Whether agents receive the health of other agents within the sight range	True
Whether to restrict every agent's field-of-view to a cone	False
Whether to use a non-learning heuristic AI	False
Whether agents receive their own health	True
Whether to scale down the reward for every episode	True
Number of time steps	2000000 (SMAC), 5000000 (SMACv2)
Reward scaling rate	20
Reward for winning (all enemies die)	$+200$
Reward for defeating one enemy	$+10$
Reward when one ally dies	-5
Basic Actions	{attack, move, stop, heal, no-op}
Directions	{East, West, South, North}
Difficulty	7 (very difficult)
Game version	SC2.4.10

Table 10: The default settings for the Multi-Agent MuJoCo control suite.

Name	**Value**
Partitioning of the robot *Ant*	(hip4, ankle4, hip1, ankle1, hip2, ankle2, hip3, ankle3)
Partitioning of the robot *Half Cheetah*	(bthigh, bshin, bfoot, fthigh, fshin, ffoot)
Partitioning of the robot *Hopper*	(thigh_joint, leg_joint, foot_joint)
Partitioning of the robot *Walker2d*	(foot_joint, leg_joint, thigh_joint, foot_left_joint, leg_left_joint, thigh_left_joint)
Number of time steps	7000000
Number of nearest joints to observe	2
Batch size	32
Number of episodes for testing during the test interval	32
Number of time steps between testing intervals	10000
Number of time steps between logging intervals	10000
Whether the agent receives its ID	True
Whether agents receive the last actions within the sight range	True
Scaling factor for the returns of *Ant*_8×1	0.2
Scaling factor for the returns of *HalfCheetah*_6×1	0.2
Scaling factor for the returns of *Hopper*_3×1	0.1
Scaling factor for the returns of *Walker2d*_6×1	0.1

F LIMITATIONS AND FUTURE WORK

Our focus in this work is primarily empirical, demonstrating the effectiveness of MSEMM across diverse MARL benchmarks. While we formally establish the permutation invariance (PI) and permutation equivariance (PE) properties of our architectural modules and adopt established definitions for simplicial embeddings(Lavoie et al., 2023), we are not able to provide theoretical error bounds or convergence guarantees for the overall framework. Deriving formal theoretical guarantees is challenging due to the architectural complexity of our work. Simplifying the framework to enable rigorous theoretical analysis, such as bounding world-model simulation error, remains a promising direction for future research.