COORDINATION FROM WITHIN: DECENTRALIZED SEQUENCE MODELING FOR MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cooperative multi-agent reinforcement learning (MARL) often requires agents to account for one another's behavior from partial observations. Under centralized training with centralized execution (CTCE), sequence modeling can capture inter-agent interactions by jointly processing team observations. Extending this approach to centralized training with decentralized execution (CTDE) is challenging because each actor must select actions from its own local history. We propose Decentralized Sequence Modeling (DSM), equipping each actor with a persistent sequence of coordination tokens that represents cooperative behavior in latent space. Our Coordination-Token Actor uses a retention layer to process these tokens together with the current local input, producing a decision representation and updating the tokens for later decisions. We introduce Coordination-Token Distillation to supervise the updated tokens with a centralized teacher's representations of joint observations. Used only during training, the teacher also guides the local policy through its action distributions. Across 44 tasks in six benchmarks, DSM achieves the highest aggregate performance among the CTDE baselines on all six and matches or exceeds the best CTCE baseline on five benchmarks. Compared with the reference gated recurrent unit (GRU) actor, DSM achieves these gains with about 68% fewer deployed parameters.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) studies how agents learn to act together toward a shared objective. Applications include multi-robot pathfinding (Sartoretti et al., 2019), warehouse logistics (Papoudakis et al., 2021), and traffic signal control (Chen et al., 2020). Under partial observability, agents have different local views of the environment, yet the consequences of their actions depend on the behavior of the team (Oliehoek & Amato, 2016). Effective cooperation therefore requires agents to learn how their decisions interact with those of others, beyond interpreting each observation in isolation.

Sequence modeling (SM) offers a direct way to capture these interactions. Centralized sequence models extract contextualized representations from agents' joint observations and use them to generate joint actions autoregressively (Wen et al., 2022; Mahjoub et al., 2025). This computation incorporates cross-agent information exchange into the policy, but requires joint inputs under centralized training with centralized execution (CTCE). Under centralized training with decentralized execution (CTDE), centralized critics (Lowe et al., 2017) or value factorization (Rashid et al., 2018) can use global information to guide learning, but that information is available only during training. Each deployed actor must rely on its local history, without the current joint observations needed for explicit cross-agent computation. The challenge is thus to learn the cooperative behavior captured by sequence models without requiring their centralized execution interface.

Existing approaches seek to strengthen decentralized cooperation by enriching local inputs or providing supervision beyond the return objective. Reconstruction methods infer missing teammate information with masked models (Kang et al., 2025) or estimate global states through diffusion (Xu et al., 2024). Other routes improve representations through auxiliary learning (Song et al., 2023; Feng et al., 2025; 2026), process observed entities as sequences (Hu et al., 2021; Gallici et al.,

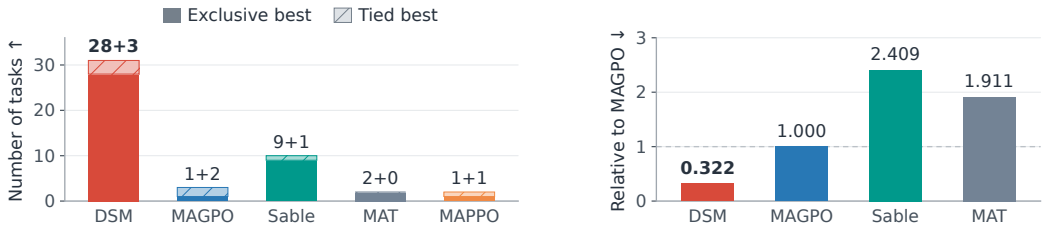
Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping 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.

2023), or guide local policies with centralized action distributions (Li et al., 2026). The actor's recurrent structure also determines how context persists across decisions. A conventional actor uses an observation encoder, a gated recurrent unit (GRU), and a policy head (Cho et al., 2014; Yu et al., 2022), compressing perceptual context and learned coordination patterns into a single hidden vector. A natural question then arises: could an actor maintain a persistent sequence of latent representations and update it jointly with its current input?

We propose Decentralized Sequence Modeling (DSM), which learns such a sequence inside each actor. Its coordination tokens represent cooperative behavior in latent space and evolve with the agent's experience. Our Coordination-Token Actor (CTA) processes the tokens together with the current input embedding through a retention layer (Sun et al., 2023). The layer lets the actor read from the sequence to make its current decision and revise the tokens for subsequent timesteps. This couples interaction within the sequence with memory across time, while both computations remain local to the agent. Figure 2 contrasts the CTA with a conventional GRU-based actor.

The token architecture determines how memory is updated, but not which information it should encode for cooperation. We therefore propose Coordination-Token Distillation (CTD) to supervise the contents of this memory during centralized training. A centralized SM teacher extracts a sequence of contextualized agent representations from the joint observations. A learned projector maps each representation to targets for the corresponding actor's updated coordination tokens. The teacher supervises the actor at two levels: its action distributions guide policy learning, while its projected features supervise the updated coordination tokens.

We evaluate DSM on 44 tasks across six cooperative benchmarks. It achieves the highest aggregate performance among the CTDE baselines on all six and the best or tied-best performance on five when CTCE methods are included. The deployed CTA uses approximately 68% fewer parameters than the conventional GRU-based reference actor used by MAGPO (Figure 1). Further analysis and ablations favor a memory with multiple coordination tokens, directly supervised by teacher targets matched to the same agent and timestep. Integrations with MAPPO and QMIX also improve learning when the targets come from centralized critics or value mixers rather than an SM teacher. Together, these results identify locally maintained sequences as an effective way to use centralized supervision in compact decentralized policies.



(a) Best-performing tasks

(b) Actor parameters

Figure 1: **Performance and deployed actor size.** (a) Exclusive and tied-best task counts among the five displayed methods across 44 tasks in six benchmarks. Labels give exclusive + tied counts. (b) Geometric-mean ratios of deployed actor parameters to MAGPO across the same tasks.

2 Preliminaries and Background

2.1 Problem Formulation

We use a decentralized partially observable Markov decision process (Dec-POMDP) to formulate cooperative MARL (Oliehoek & Amato, 2016). The process is specified by $\mathcal{G} = (\mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, P, r, \{\mathcal{O}_i\}_{i\in\mathcal{N}}, \Omega, \gamma, \rho_0)$. Here $\mathcal{N} = \{1, \ldots, N\}$ is the agent set, $\mathcal{S}$ is the state space, and $\mathcal{A}_i$ and $\mathcal{O}_i$ are agent i's action and observation spaces. Starting from $s_0 \sim \rho_0$, agent i receives its component o_t^i of the joint observation $o_t^{1:N} \sim \Omega(\cdot \mid s_t)$. The joint action $a_t^{1:N}$ produces the shared reward $r_t = r(s_t, a_t^{1:N})$, and the environment transitions to $s_{t+1} \sim P(\cdot \mid s_t, a_t^{1:N})$. Let

$\tau_t^i = (o_0^i, a_0^i, \ldots, o_t^i)$ and $\tau_t = (\tau_t^1, \ldots, \tau_t^N)$; the objective is to maximize expected discounted return with a factorized policy:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r_t \right], \qquad \pi(a_t^{1:N} \mid \tau_t) = \prod_{i=1}^{N} \pi^i(a_t^i \mid \tau_t^i), \quad \gamma \in [0, 1). \tag{1}$$

CTDE allows global states or joint observations to guide learning, while each agent executes a policy conditioned on its own local history. With CTCE, centralized information is also available when the agents select actions.

2.2 Recurrent Decentralized Policies

A conventional recurrent actor uses a GRU to summarize local history and a multilayer perceptron (MLP) to map the resulting representation to an action distribution (Cho et al., 2014; Yu et al., 2022). Its input embedding $e_t^i = f_{\text{emb}}(o_t^i, a_{t-1}^i)$ encodes the current observation and previous action. Agents share network parameters but maintain separate recurrent states:

$$(z_t^i, h_t^i) = \text{GRU}(e_t^i, h_{t-1}^i), \qquad \pi^i(\cdot \mid \tau_t^i) = \text{MLP}(z_t^i). \tag{2}$$

The recurrent state lets each decision depend on the agent's past experience. Throughout the paper, f_{emb} denotes local input encoding, and MLP maps representations to action distributions.

2.3 Sequence Modeling for Centralized Execution

An encoder–decoder sequence model can represent a centralized policy by jointly encoding observations and generating actions autoregressively, as in MAT (Wen et al., 2022) and Sable (Mahjoub et al., 2025). The encoder extracts contextualized agent representations from joint observations; the decoder combines them with preceding actions to produce each agent's decision representation:

$$\hat{o}_t^{1:N} = \text{Encoder}(o_t^{1:N}), \qquad \bar{z}_t^i = \text{Decoder}(\hat{o}_t^{1:N}, a_t^{<i})_i. \tag{3}$$

Here $a_t^{<i} = (a_t^1, \ldots, a_t^{i-1})$ contains actions generated earlier in the same environment timestep; recurrent encoder and decoder states are omitted from the notation. The decoder's MLP maps $\bar{z}_t^i$ to the conditional action distribution $q^i = \text{MLP}(\bar{z}_t^i)$, giving the joint policy

$$q(a_t^{1:N} \mid \tau_t) = \prod_{i=1}^{N} q^i(a_t^i \mid \hat{o}_t^{1:N}, a_t^{<i}). \tag{4}$$

Conditioning on the action prefix explicitly models dependencies between agents' decisions. Joint observation features supply the team context for these decisions at execution time.

3 Decentralized Sequence Modeling

DSM combines a persistent local token sequence with centralized supervision of its updates (Figure 2). The CTA processes the tokens jointly with the current input, while CTD supervises the updated tokens and policy guidance trains the action distribution.

3.1 Coordination-Token Actor

We build the actor's sequence from persistent coordination tokens, using its own recurrent memory as the context for each new local input. With agent indices omitted on internal representations, each actor's memory comprises K persistent coordination tokens, $C_t^{1:K} = [c_t^1; \ldots; c_t^K] \in \mathbb{R}^{K \times d}$, and a retention matrix state h_t. The tokens form the sequence processed alongside the local input, while the matrix state carries the accumulated key–value information used by retention. The tokens are learned latent representations of cooperative behavior, with K independent of the number of agents. Their positions are not assigned to individual teammates: each actor maintains a latent sequence rather than reconstructing the team's observations. Agents share the learned token initialization and network parameters, but keep separate token sequences and matrix states.

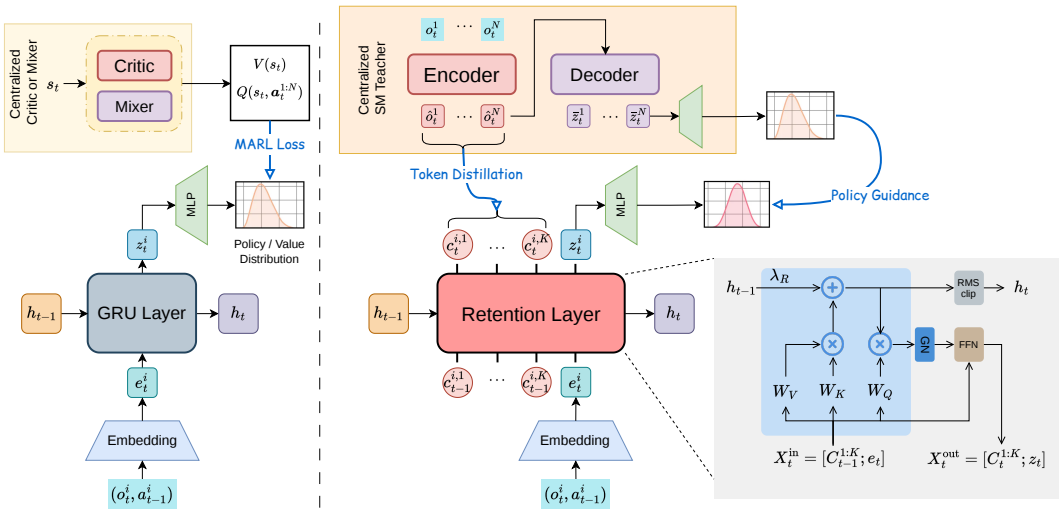


Figure 2: **Conventional GRU-based actor (left) and DSM's CTA (right).** The CTA processes coordination tokens and the local input through retention, while a centralized teacher supplies token targets and policy guidance during training.

At timestep t, the actor encodes its observation and previous action, then appends the embedding to the incoming coordination tokens:

$$e_t = f_{\text{emb}}(o_t^i, a_{t-1}^i), \qquad X_t^{\text{in}} = [C_{t-1}^{1:K}; e_t]. \tag{5}$$

The retention layer updates all $K+1$ positions together, returning $X_t^{\text{out}} = [C_t^{1:K}; z_t]$ and h_t:

$$(X_t^{\text{out}}, h_t) = \text{Retention}(X_t^{\text{in}}, h_{t-1}), \qquad \pi^i(\cdot \mid \tau_t^i) = \text{MLP}(z_t). \tag{6}$$

The policy head reads z_t for the current decision, while $C_t^{1:K}$ and h_t carry the updated memory into the next timestep. A fresh embedding occupies the local-input position at the next timestep, keeping the sequence length fixed as the token contents evolve.

The retention layer exchanges information between the local embedding and coordination tokens through a shared matrix (Sun et al., 2023; Mahjoub et al., 2025). For one head, the query, key, and value projections are $Q_t = X_t^{\text{in}} W_Q$, $K_t = X_t^{\text{in}} W_K$, and $V_t = X_t^{\text{in}} W_V$, omitting fixed positional offsets. Their key–value products update the recurrent matrix state:

$$h_t = \text{RMSClip}\big(\lambda_R h_{t-1} + K_t^\top V_t\big). \tag{7}$$

All $K+1$ positions contribute to the accumulated matrix. The current queries read it before clipping through $Q_t(\lambda_R h_{t-1} + K_t^\top V_t)$. The local-input query can therefore access the coordination tokens, while the token queries incorporate the new embedding. The temporal decay λ_R acts between environment timesteps, and RMSClip bounds the root-mean-square (RMS) magnitude of the stored state by M. Group normalization (GN) and the residual feed-forward network (FFN) then transform the readout into X_t^{out}, with the residual carrying the entire input sequence.

3.2 Coordination-Token Distillation

CTD supervises token updates using the team context encoded by a centralized SM teacher. The teacher maps joint observations to contextualized agent representations $\hat{o}_t^{1:N}$, where $\hat{o}_t^i$ summarizes the context associated with agent i. For that agent, a learned projector g maps the feature to Kd values arranged as K targets for the updated tokens $C_t^{i,1:K}$ at the same timestep:

$$\hat{C}_t^{i,1:K} = g(\text{stopgrad}(\hat{o}_t^i)), \qquad \mathcal{L}_{\text{align}} = \frac{1}{NTKd} \sum_{t,i} \|C_t^{i,1:K} - \hat{C}_t^{i,1:K}\|_F^2. \tag{8}$$

Here T is the training sequence length, and the loss is also averaged over sampled trajectories. The teacher feature is detached before entering g, so alignment updates the actor and projector

without backpropagating into the teacher. Teacher features supervise the tokens through the training objective and are not additional inputs to the CTA. The supervised tokens enter the next local input sequence, so CTD acts directly on the actor's recurrent memory.

This alignment alone also admits constant features, since both the actor tokens and their projected targets are learned. We therefore follow the variance-floor principle of VICReg (Bardes et al., 2022) and penalize insufficient variation on both sides:

$$\mathcal{L}_{\mathrm{CTD}} = \lambda_a \mathcal{L}_{\mathrm{align}} + \lambda_v \big(v(C) + v(\hat{C})\big). \tag{9}$$

Here $v(F) = \frac{1}{Kd}\sum_{j=1}^{Kd}[\sigma_0 - \sqrt{\mathrm{Var}_{\mathrm{samples}}(F_j) + \varepsilon_v}]_+$, with $[x]_+ = \max(x, 0)$. C and $\hat{C}$ collect token features across training samples, and the penalty encourages each coordinate to vary across those samples. CTD requires a centralized feature that can be projected into token space, not a particular teacher architecture; §4.5 uses critic and mixer features in the same construction.

3.3 Training DSM

With a centralized SM teacher, DSM combines CTD with MAGPO's policy-guidance training scheme (Li et al., 2026). Let θ, ϕ, and ψ denote the actor, teacher, and projector parameters, respectively. The teacher collects trajectories, while local actors advance their memories along the same observation sequences. For a sampled action a_t^i, let b_t^i be its recorded behavior-teacher probability and $\hat{A}_t^i$ its advantage computed by generalized advantage estimation (GAE; Schulman et al., 2016). For $\rho_t^i = \pi_\theta^i(a_t^i \mid \tau_t^i)/b_t^i$, the actor's return-based auxiliary objective uses proximal policy optimization (PPO; Schulman et al., 2017) clipping with width ϵ:

$$\mathcal{L}_{\mathrm{aux}} = -\mathbb{E}\Big[\min\Big(\rho_t^i \hat{A}_t^i, \mathrm{clip}(\rho_t^i, 1-\epsilon, 1+\epsilon)\hat{A}_t^i\Big)\Big]. \tag{10}$$

Policy guidance uses Kullback–Leibler (KL) divergence; with auxiliary weight α, the actor and projector jointly minimize

$$\mathcal{L}_{\mathrm{actor}}(\theta, \psi) = \mathbb{E}\big[D_{\mathrm{KL}}\big(\mathrm{stopgrad}(q_\phi^i) \,\|\, \pi_\theta^i\big)\big] + \alpha \mathcal{L}_{\mathrm{aux}} + \mathcal{L}_{\mathrm{CTD}}. \tag{11}$$

The teacher distribution is conditioned on the joint features and action prefix in Eq. (4), whereas the local policy conditions on its own history. Meanwhile, the teacher learns through its policy and value objectives, an entropy term, and a guidance term that discourages large deviations from the local actor. Algorithm 1 in Appendix D gives the complete update.

At execution, the teacher and projector are removed. Each agent updates its own tokens and matrix state from its observation and previous action, then selects an action through Eq. (6). These independently executed policies form the factorized joint policy in Eq. (1).

4 Experiments

We evaluate DSM with the following questions in mind: (i) Can DSM improve cooperative performance with a smaller deployed actor? (ii) Which aspects of the token memory and its supervision contribute to these gains? (iii) Can the same representation supervision benefit CTDE learners without a centralized SM teacher?

4.1 Environments and Evaluation Protocol

The evaluation covers 44 tasks from six cooperative benchmarks: CoordSum (Li et al., 2026), Level-Based Foraging (LBF) (Christianos et al., 2020), Connector (Bonnet et al., 2024), the Multi-Agent Particle Environment (MPE) (Lowe et al., 2017), Robotic Warehouse (RWARE) (Papoudakis et al., 2021), and SMAX (Rutherford et al., 2024). We compare against four CTDE baselines, MAGPO, CTDS, HAPPO, and MAPPO, and two CTCE baselines, MAT and Sable, implemented in the JAX/Mava ecosystem (de Kock et al., 2023). Appendix C specifies the tasks and their observation and action spaces.

Main comparisons use ten random seeds and 20 million environment steps per task; ablations use five seeds. We follow the evaluation protocol of Gorsane et al. (2022), also adopted by Sable and MAGPO (Mahjoub et al., 2025; Li et al., 2026). Benchmark aggregates use normalized IQMs (Agarwal et al., 2021), with SMAX evaluated by win rate. See Appendix C.7.

Table 1: *Per environment episode return.* Inter-quartile mean of the absolute episode returns with 95% stratified bootstrap confidence intervals.Bold values indicate the highest score per environment.

	CTDE					CTCE	
Environment	**DSM**	**MAGPO**	**CTDS**	**HAPPO**	**MAPPO**	**MAT**	**Sable**
CoordSum	**0.97**$_{(0.97,0.98)}$	0.89$_{(0.88,0.90)}$	0.31$_{(0.27,0.34)}$	0.70$_{(0.68,0.72)}$	0.85$_{(0.84,0.86)}$	0.44$_{(0.41,0.46)}$	0.51$_{(0.48,0.54)}$
LBF	**1.00**$_{(1.00,1.00)}$	**1.00**$_{(1.00,1.00)}$	0.99$_{(0.99,0.99)}$	0.99$_{(0.99,0.99)}$	**1.00**$_{(1.00,1.00)}$	0.99$_{(0.98,0.99)}$	**1.00**$_{(1.00,1.00)}$
Connector	**0.95**$_{(0.94,0.95)}$	0.94$_{(0.94,0.94)}$	0.92$_{(0.92,0.92)}$	0.89$_{(0.88,0.89)}$	0.91$_{(0.91,0.92)}$	0.88$_{(0.87,0.89)}$	**0.95**$_{(0.95,0.95)}$
MPE	0.86$_{(0.86,0.87)}$	0.84$_{(0.83,0.85)}$	0.83$_{(0.83,0.84)}$	0.76$_{(0.75,0.76)}$	0.79$_{(0.78,0.79)}$	0.66$_{(0.63,0.67)}$	**0.94**$_{(0.94,0.95)}$
RWARE	**0.76**$_{(0.75,0.77)}$	0.64$_{(0.63,0.65)}$	0.40$_{(0.39,0.41)}$	0.21$_{(0.19,0.22)}$	0.45$_{(0.42,0.48)}$	0.58$_{(0.56,0.60)}$	0.68$_{(0.66,0.70)}$
SMAX	**0.98**$_{(0.98,0.98)}$	0.95$_{(0.94,0.95)}$	0.93$_{(0.91,0.94)}$	0.73$_{(0.72,0.75)}$	0.86$_{(0.84,0.87)}$	0.92$_{(0.91,0.94)}$	0.94$_{(0.92,0.95)}$

4.2 Cooperative Performance and Deployment Cost

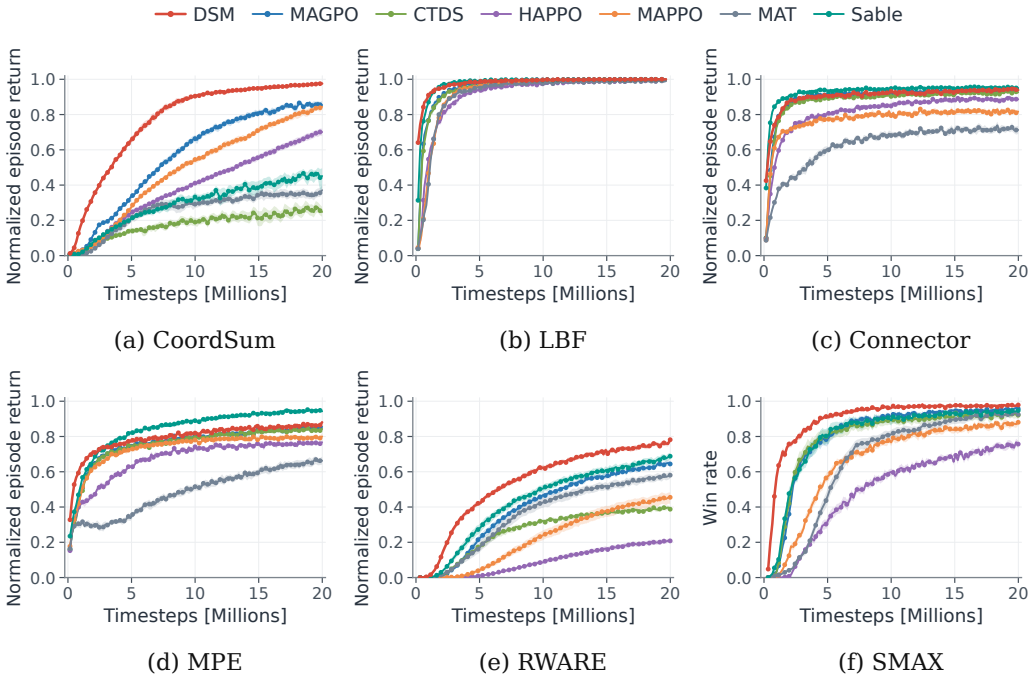


Figure 3: *Sample efficiency aggregated by environment suite.* Curves show normalized episode-return IQMs for the first five suites and win-rate IQMs for SMAX, aggregated over tasks and ten seeds per task, with 95% stratified bootstrap confidence intervals.

DSM achieves the highest CTDE aggregate on all six benchmarks (Table 1)[1] and the best or tied-best aggregate on five when CTCE methods are included. The largest gains over MAGPO occur on CoordSum and RWARE, where normalized IQMs increase from 0.89 to 0.97 and from 0.64 to 0.76, respectively. Differences are smaller on LBF and Connector, where several methods already achieve high scores. The results show that a locally maintained sequence can support strong cooperative performance without joint observations at execution.

The learning curves distinguish gains in learning speed from differences in attained performance (Figure 3). DSM reaches high returns earlier on CoordSum and RWARE, and high win rates earlier on SMAX. On RWARE, its lead persists through the end of training, so the improvement includes

[1]DSM exceeds previous reference maxima on several tasks; we therefore recompute normalization bounds across all compared methods, so normalized scores may differ from those reported by Sable and MAGPO.

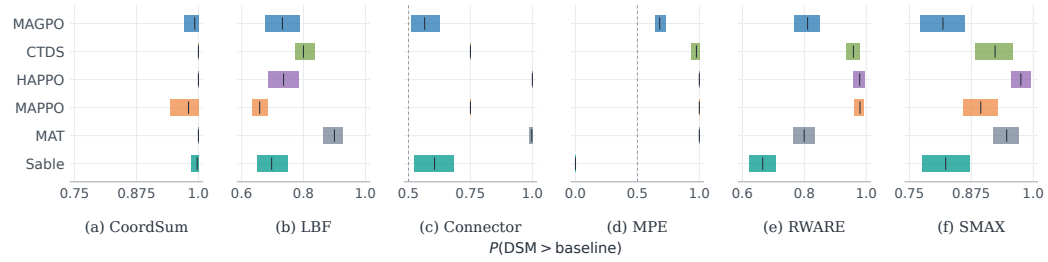


Figure 4: *Probability of improvement of DSM over each baseline.* Estimates average over common tasks, with ties counted as half a win; ticks and bands show estimates and 95% stratified bootstrap confidence intervals over ten seeds. Values above 0.5 favor DSM.

both faster learning and a higher attained return. Across these three suites, the probability of improvement over MAGPO exceeds 0.8 (Figure 4), supporting the aggregate gains through comparisons across tasks and runs.

At the task level, DSM ranks first on 31 of 44 tasks among the five methods in Figure 1, with 28 exclusive wins and three ties. These gains come with approximately 68% fewer deployed parameters than MAGPO's GRU actor, measured by the geometric mean of their per-task parameter ratios. The task-level advantage spans different team sizes and warehouse layouts; Sable remains stronger on the largest Connector task and all three MPE tasks (Table 2). Appendix A provides the complete task scores, learning curves, and performance profiles.

4.3 Contributions of Token Memory and Supervision

DSM changes both the structure of the actor's memory and the supervision used to train it. To separate their contributions, we compare DSM with a single-token actor, a GRU trained with representation distillation (GRU+CTD), and the token actor without CTD (DSM w/o CTD). The first two variants change the memory structure while keeping representation supervision; the third removes that supervision without changing the token actor. MAGPO serves as the policy-guidance baseline.

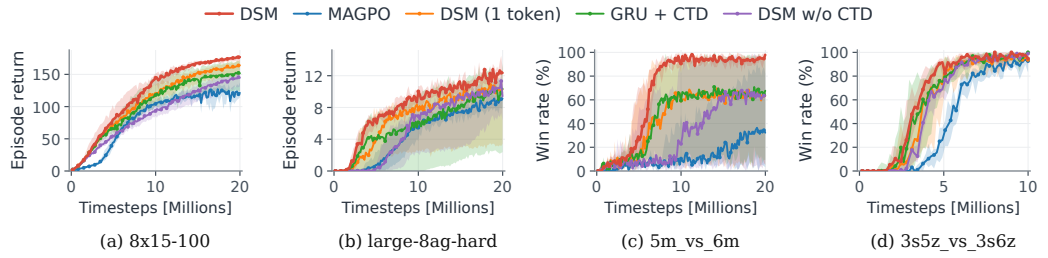
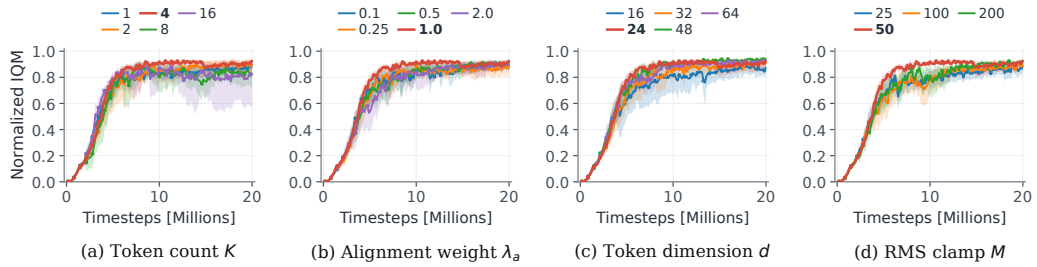


Figure 5: *Ablations of token memory and representation supervision.* Curves show episode-return or win-rate IQMs over five seeds, with shaded 95% bootstrap confidence intervals.

DSM has the highest performance averaged over training on all four tasks (Figure 5). The memory-structure difference is particularly clear on SMAX `5m_vs_6m`, where DSM reaches a 97.9% win rate, compared with 63.5% for a single token and 67.7% for GRU+CTD. The single-token actor retains both retention and CTD, so its lower win rate supports maintaining multiple recurrent tokens. On `3s5z_vs_3s6z`, GRU+CTD reaches the higher endpoint, whereas DSM has the higher win rate averaged over training, 64.87% versus 60.87%.

The sensitivity study further examines the size of this memory and the strength of its supervision (Figure 6; settings in Appendix E.2). Increasing the sequence from four to sixteen tokens lowers aggregate performance, whereas doubling token width brings a modest improvement. An intermediate alignment weight outperforms both weaker and stronger supervision, while RMS limits of 50



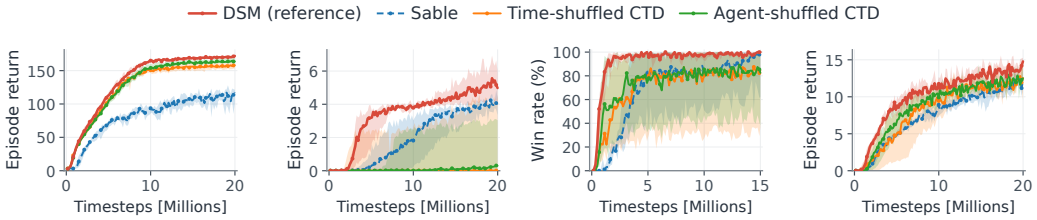
(a) Token count K (b) Alignment weight λ_a (c) Token dimension d (d) RMS clamp M

Figure 6: *Hyperparameter sensitivity across four tasks.* Curves show normalized IQMs over five seeds with shaded 95% bootstrap intervals; bold labels denote reference settings.

and 200 yield similar attained performance. These results favor a compact token sequence with appropriately weighted supervision.

4.4 Matching Teacher Targets to Local Updates

CTD assigns each teacher target to the corresponding agent and timestep, so a token update is supervised by its own centralized context. To test whether this assignment matters, we shuffle targets either across timesteps within an episode fragment or across agents at the same timestep. Each shuffle preserves the sampled targets and their dimensions but changes which update they supervise.



(a) 3x30-50 (b) large-4ag-hard (c) 10m_vs_11m (d) small-4ag-hard

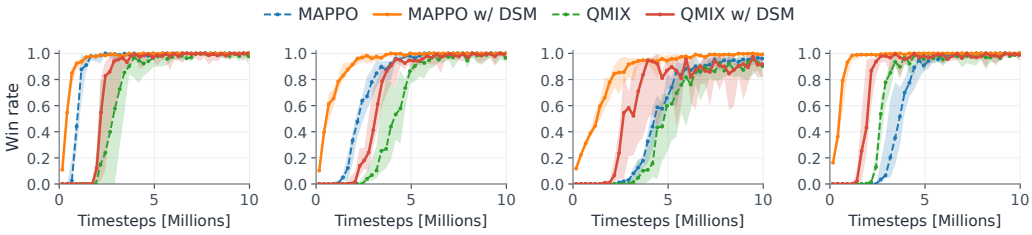
Figure 7: *Ablations of teacher-target correspondence.* DSM is compared with time-shuffled and agent-shuffled CTD using IQMs over five seeds and shaded 95% bootstrap confidence intervals.

Both forms of shuffling reduce performance on all four tasks (Figure 7). The effect is particularly marked on RWARE `large-4ag-hard`, where the shuffled variants remain near zero while DSM reaches a return of 4.98. On SMAX `10m_vs_11m`, time-shuffled and agent-shuffled CTD reach win rates of 84.6% and 82.5%, compared with 100% for DSM. These variants retain the token architecture and a representation-matching objective, but the teacher features are no longer paired with the local context that produced each update. The drop under both interventions supports matching centralized targets to the agent and timestep of the coordination tokens they supervise.

4.5 Integration with MAPPO and QMIX

CTD requires a inter-agent representation, but such representation need not come from an autoregressive policy. We examine this distinction by integrating the token memory and CTD with MAPPO and QMIX. Targets are projected from MAPPO's critic features and QMIX's mixer features.

The main benefit of these integrations is faster learning (Figure 8). On the four classical SMAX tasks, both DSM variants reach high win rates earlier, while the original learners eventually approach similar performance. The gains span actor–critic learning and value factorization, whose centralized features are trained for value estimation rather than encoder-decoder architecture. Their usefulness as targets shows that token supervision can draw on representations learned under different objectives, extending DSM beyond a centralized sequence-model teacher.



(a) 2s3z (b) 3s5z (c) 3s_vs_5z (d) 6h_vs_8z

Figure 8: *DSM with MAPPO and QMIX on four SMAX tasks.* Solid and dashed curves show DSM variants and the original methods, respectively, using win-rate IQMs over five seeds with shaded 95% bootstrap confidence intervals.

5 Related Work

Learning decentralized cooperation. Centralized critics (Foerster et al., 2018; Yu et al., 2022), value factorization (Sunehag et al., 2017; Rashid et al., 2018; Wang et al., 2021a), and sequential policy updates (Kuba et al., 2022) train local decision rules toward a shared objective. Role-based methods organize cooperation through latent roles, role-related action spaces, or role-aware representations (Wang et al., 2020; 2021b; Hu et al., 2024). Rather than assigning responsibilities across agents, DSM structures what each actor remembers in a persistent token sequence. Supervising this sequence complements both actor–critic learning and value factorization.

Sequence models and local memory. MAT models dependencies between agents' actions through an autoregressive sequence policy (Wen et al., 2022), while Sable adds temporal memory through retention (Mahjoub et al., 2025). Both process joint observations at decision time and therefore rely on centralized execution. UPDeT and TransfQMix instead apply Transformers to entities within local observations (Hu et al., 2021; Gallici et al., 2023). Their observation tokens represent visible entities, whereas DSM's coordination tokens are latent representations carried between decisions. Persistent token memory also appears in Recurrent Memory Transformer (Bulatov et al., 2022); DSM trains such memory for cooperative control using centralized features, while its execution-time updates remain local.

Transferring centralized coordination. CTDS trains local utility networks to match centralized action values (Zhao et al., 2024), while PTDE learns local predictors of personalized centralized features (Chen et al., 2024). MAGPO transfers action-level coordination through an autoregressive teacher that guides decentralized policy distributions, while its own objective discourages disagreement with the local policies (Li et al., 2026). Conditional factorization and policy-consistency constraints connect joint and local policies (Wang et al., 2023; Li et al., 2025). Local representations can be enriched through observation or state reconstruction (Kang et al., 2025; Xu et al., 2024; Liang et al., 2025), masked reconstruction with contrastive alignment (Song et al., 2023; Feng et al., 2026), or joint-transition prediction (Feng et al., 2025). DSM directs centralized representation supervision to the updated coordination tokens, complementing policy guidance by shaping the memory used in later decisions. It requires no observation or state reconstruction as an intermediate control task. Appendix B compares these representation and supervision interfaces.

6 Conclusion and Future Work

In this work, we investigate how to bring sequence modeling to cooperative MARL under CTDE. We propose DSM, whose actor jointly processes local inputs and a persistent sequence of coordination tokens through retention. A centralized teacher distills its representations into these tokens and guides policy learning, while execution relies solely on local histories. Across 44 tasks, DSM achieves the highest CTDE aggregate on all six benchmarks and the best or tied-best aggregate on five including CTCE methods. Its actor uses approximately 68% fewer deployed parameters than MAGPO's GRU actor. Future work could improve distillation by selecting teacher representations that are informative for cooperation and predictable from each agent's local history.

AI USE STATEMENT

Generative AI tools assisted with manuscript revision and language polishing, literature retrieval and citation checking, and the preparation of plotting scripts. They also assisted with drafting the extended background and environment descriptions in the appendix. The research ideas and experiments were developed and conducted by the authors, who take responsibility for the final manuscript and all reported results.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In *Advances in Neural Information Processing Systems*, volume 34, pp. 29304–29320, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I. Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries P. Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn N. Waters, Mohamed A. Mimouni, Ulrich A. Mbou Sob, Ruan de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: A diverse suite of scalable reinforcement learning environments in JAX. In *International Conference on Learning Representations*, volume 2024, pp. 49264–49293, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/d6c38e81801a8d10afeabdb0a32fca8a-Abstract-Conference.html.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091, 2022. doi: 10.52202/068431-0805. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/47e288629a6996a17ce50b90a056a0e1-Abstract.html.

Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3414–3421, 2020. doi: 10.1609/aaai.v34i04.5744. URL https://ojs.aaai.org/index.php/AAAI/article/view/5744.

Yiqun Chen, Hangyu Mao, Jiaxin Mao, Shiguang Wu, Tianle Zhang, Bin Zhang, Wei Yang, and Hongxing Chang. PTDE: Personalized training with distilled execution for multi-agent reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 31–39, 2024. doi: 10.24963/ijcai.2024/4. URL https://www.ijcai.org/proceedings/2024/4.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10707–10717, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/7967cc8e3ab559e68cc944c44b1cf3e8-Abstract.html.

Ruan de Kock, Omayma Mahjoub, Sasha Abramowitz, Wiem Khlifi, Callum Rhys Tilbury, Claude Formanek, Andries P. Smit, and Arnu Pretorius. Mava: A research library for distributed multi-agent reinforcement learning in JAX. arXiv:2107.01460v2, 2023. URL https://arxiv.org/abs/2107.01460v2.

Mingxiao Feng, Yaodong Yang, Wengang Zhou, and Houqiang Li. TIMAR: Transition-informed representation for sample-efficient multi-agent reinforcement learning. *Neural Networks*, 184: 107081, 2025. doi: 10.1016/j.neunet.2024.107081. URL https://doi.org/10.1016/j.neunet.2024.107081.

Mingxiao Feng, Lin Liu, Wengang Zhou, and Houqiang Li. Learning to ground multi-agent reinforcement learning with masked multi-agent AutoEncoding. *Neural Networks*, 205(Part A): 109408, 2026. doi: 10.1016/j.neunet.2026.109408. URL https://doi.org/10.1016/j.neunet.2026.109408.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11794. URL https://ojs.aaai.org/index.php/AAAI/article/view/11794.

Matteo Gallici, Mario Martin, and Ivan Masmitja. TransfQMix: Transformers for leveraging the graph structure of multi-agent reinforcement learning problems. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 1679–1687, 2023. URL https://www.ifaamas.org/Proceedings/aamas2023/pdfs/p1679.pdf.

Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative MARL. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5510–5521, 2022. doi: 10.52202/068431-0398. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/249f73e01f0a2bb6c8d971b565f159a7-Abstract-Conference.html.

Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. UPDeT: Universal multi-agent reinforcement learning via policy decoupling with transformers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=v9c7hr9ADKx.

Zican Hu, Zongzhang Zhang, Huaxiong Li, Chunlin Chen, Hongyu Ding, and Zhi Wang. Attention-guided contrastive role representations for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/66e226469f20625aaebddbe47f0ca997-Abstract-Conference.html.

Sehyeok Kang, Yongsik Lee, Gahee Kim, Song Chong, and Se-Young Yun. MA^{2}E: Addressing partial observability in multi-agent reinforcement learning with masked auto-encoder. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/33301bb40020a56ef56b8b5081e5c4d5-Abstract-Conference.html.

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.

Yueheng Li, Guangming Xie, and Zongqing Lu. Multi-agent guided policy optimization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=OT8beoc0W0.

Zhiyuan Li, Wenshuai Zhao, Lijun Wu, and Joni Pajarinen. AgentMixer: Multi-agent correlated policy factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18611–18619, 2025. doi: 10.1609/aaai.v39i17.34048. URL https://ojs.aaai.org/index.php/AAAI/article/view/34048.

Qifan Liang, Yixiang Shan, Haipeng Liu, Zhengbang Zhu, Ting Long, Weinan Zhang, and Yuan Tian. Reconstruction-guided policy: Enhancing decision-making through agent-wise state consistency. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/9213010cbcd6ba8e1f1cf1533835d51c-Paper-Conference.pdf.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative–competitive environments. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html.

Omayma Mahjoub, Sasha Abramowitz, Ruan de Kock, Wiem Khlifi, Simon du Toit, Jemma Daniel, Louay Ben Nessir, Louise Beyers, Claude Formanek, Liam Clark, and Arnu Pretorius. Sable: a performant, efficient and scalable sequence model for MARL. In *Proceedings of the International Conference on Machine Learning*, volume 267, pp. 42579–42614, 2025. URL https://proceedings.mlr.press/v267/mahjoub25a.html.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016. doi: 10.1007/978-3-319-28929-8. URL https://doi.org/10.1007/978-3-319-28929-8.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/a8baa56554f96369ab93e4f3bb068c22-Abstract-round1.html.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 80, pp. 4295–4304, 2018. URL https://proceedings.mlr.press/v80/rashid18a.html.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktäschel, Chris Lu, and Jakob Foerster. JaxMARL: Multi-agent RL environments and algorithms in JAX. In *Advances in Neural Information Processing Systems*, volume 37, pp. 50925–50951, 2024. doi: 10.52202/079017-1612. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/5aee125f052c90e326dcf6f380df94f6-Abstract-Datasets_and_Benchmarks_Track.html.

Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T. K. Satish Kumar, Sven Koenig, and Howie Choset. PRIMAL: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019. doi: 10.1109/LRA.2019.2903261. URL https://arxiv.org/abs/1809.03531.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017. URL https://arxiv.org/abs/1707.06347.

Haolin Song, Mingxiao Feng, Wengang Zhou, and Houqiang Li. MA2CL: Masked attentive contrastive learning for multi-agent reinforcement learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 4226–4234, 2023. doi: 10.24963/ijcai.2023/470. URL https://www.ijcai.org/proceedings/2023/470.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. arXiv:2307.08621, 2023. URL https://arxiv.org/abs/2307.08621.

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:1706.05296, 2017. URL https://arxiv.org/abs/1706.05296.

Jiangxing Wang, Deheng Ye, and Zongqing Lu. More centralized training, still decentralized execution: Multi-agent conditional policy factorization. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=znLlSgN-4S0.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent Q-learning. In *International Conference on Learning Representations*, 2021a. URL https://arxiv.org/abs/2008.01062.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the International Conference on Machine Learning*, volume 119, pp. 9876–9886, 2020. URL https://proceedings.mlr.press/v119/wang20f.html.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: Learning roles to decompose multi-agent tasks. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=TTUVg6vkNjK.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16509–16521, 2022. doi: 10.52202/068431-1201. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/69413f87e5a34897cd010ca698097d0a-Abstract-Conference.html.

Zhiwei Xu, Hangyu Mao, Nianmin Zhang, Xin Xin, Pengjie Ren, Dapeng Li, Bin Zhang, Guoliang Fan, Zhumin Chen, Changwei Wang, and Jiangjin Yin. Beyond local views: Global state inference with diffusion models for cooperative multi-agent reinforcement learning. arXiv:2408.09501v1, 2024. URL https://arxiv.org/abs/2408.09501v1.

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 *Advances in Neural Information Processing Systems*, volume 35, pp. 24611–24624, 2022. doi: 10.52202/068431-1787. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract.html.

Jian Zhao, Xunhan Hu, Mingyu Yang, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. CTDS: Centralized teacher with decentralized student for multiagent reinforcement learning. *IEEE Transactions on Games*, 16(1):140–150, 2024. doi: 10.1109/TG.2022.3232390. URL https://doi.org/10.1109/TG.2022.3232390.

A Complete Experimental Results

A.1 Scores on All Tasks

Table 2 reports per-task performance after 20 million training interactions as mean episode return and sample standard deviation over ten seeds. SMAX entries are native returns, whereas its learning curves use win rates. Appendix C specifies the environment settings and evaluation protocol.

Excluding three tied tasks, exclusive-best counts are DSM 28, MAGPO 1, CTDS 0, HAPPO 0, MAPPO 1, MAT 2, and Sable 9 across seven methods; Figure 1 compares five.

Table 2: **Per-task performance after 20 million environment steps.** Entries are native episode-return means ± sample standard deviations over ten seeds, including for SMAX. Bold marks the highest reported mean in each row; dashes denote unavailable results.

		CTDE					CTCE	
	Scenario	DSM	MAGPO	CTDS	HAPPO	MAPPO	MAT	Sable
CoordSum	3x10-30	**160.02**$_{\pm 0.91}$	153.13$_{\pm 1.49}$	111.98$_{\pm 11.75}$	153.32$_{\pm 2.00}$	156.37$_{\pm 3.76}$	68.29$_{\pm 17.71}$	153.70$_{\pm 3.97}$
	3x30-50	**170.74**$_{\pm 1.45}$	156.62$_{\pm 1.68}$	76.27$_{\pm 14.87}$	129.35$_{\pm 5.51}$	158.05$_{\pm 4.64}$	87.79$_{\pm 9.26}$	125.04$_{\pm 7.57}$
	5x20-80	**168.80**$_{\pm 5.33}$	157.61$_{\pm 4.10}$	19.62$_{\pm 12.43}$	119.19$_{\pm 10.59}$	142.51$_{\pm 5.13}$	86.86$_{\pm 8.45}$	48.35$_{\pm 16.62}$
	8x15-100	**176.44**$_{\pm 3.80}$	129.94$_{\pm 8.93}$	23.96$_{\pm 16.78}$	77.60$_{\pm 5.37}$	127.32$_{\pm 13.45}$	57.22$_{\pm 5.24}$	28.91$_{\pm 19.77}$
LBF	8x8-2p-2f-coop	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$
	2s-8x8-2p-2f-coop	**1.00**$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$	1.00$_{\pm 0.00}$	**1.00**$_{\pm 0.00}$
	10x10-3p-3f	**1.00**$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	0.99$_{\pm 0.00}$	1.00$_{\pm 0.00}$
	2s-10x10-3p-3f	0.99$_{\pm 0.00}$	0.97$_{\pm 0.01}$	0.87$_{\pm 0.02}$	0.99$_{\pm 0.01}$	**1.00**$_{\pm 0.00}$	0.97$_{\pm 0.01}$	0.99$_{\pm 0.00}$
	15x15-3p-5f	**0.99**$_{\pm 0.00}$	0.99$_{\pm 0.00}$	0.96$_{\pm 0.02}$	0.91$_{\pm 0.02}$	0.97$_{\pm 0.02}$	0.91$_{\pm 0.02}$	0.96$_{\pm 0.01}$
	15x15-4p-3f	**1.00**$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	1.00$_{\pm 0.00}$	0.99$_{\pm 0.01}$	1.00$_{\pm 0.00}$
	15x15-4p-5f	**1.00**$_{\pm 0.00}$	0.99$_{\pm 0.00}$	0.99$_{\pm 0.01}$	0.89$_{\pm 0.02}$	0.98$_{\pm 0.01}$	0.97$_{\pm 0.01}$	0.99$_{\pm 0.00}$
Connector	con-5x5x3a	**0.88**$_{\pm 0.00}$	0.86$_{\pm 0.01}$	0.85$_{\pm 0.01}$	0.85$_{\pm 0.01}$	0.83$_{\pm 0.01}$	0.81$_{\pm 0.01}$	0.85$_{\pm 0.01}$
	con-7x7x5a	**0.81**$_{\pm 0.00}$	0.79$_{\pm 0.01}$	0.77$_{\pm 0.02}$	0.76$_{\pm 0.01}$	0.75$_{\pm 0.01}$	0.75$_{\pm 0.02}$	0.79$_{\pm 0.01}$
	con-10x10x10a	0.74$_{\pm 0.00}$	**0.74**$_{\pm 0.01}$	0.71$_{\pm 0.01}$	0.64$_{\pm 0.02}$	0.70$_{\pm 0.01}$	0.65$_{\pm 0.03}$	0.74$_{\pm 0.01}$
	con-15x15x23a	0.40$_{\pm 0.01}$	0.63$_{\pm 0.02}$	0.61$_{\pm 0.02}$	0.27$_{\pm 0.01}$	0.63$_{\pm 0.02}$	0.25$_{\pm 0.11}$	**0.70**$_{\pm 0.01}$
MPE	simple_spread_3ag	-5.66$_{\pm 0.04}$	-6.10$_{\pm 0.22}$	-6.34$_{\pm 0.30}$	-6.63$_{\pm 0.52}$	-6.72$_{\pm 0.23}$	-6.59$_{\pm 0.24}$	**-4.92**$_{\pm 0.31}$
	simple_spread_5ag	-19.24$_{\pm 0.22}$	-18.67$_{\pm 0.44}$	-20.38$_{\pm 0.41}$	-23.42$_{\pm 0.55}$	-22.84$_{\pm 0.24}$	-25.30$_{\pm 1.83}$	**-12.75**$_{\pm 0.95}$
	simple_spread_10ag	-39.05$_{\pm 0.42}$	-40.51$_{\pm 0.85}$	-40.09$_{\pm 0.77}$	-43.68$_{\pm 0.58}$	-41.83$_{\pm 0.55}$	-50.07$_{\pm 1.82}$	**-36.93**$_{\pm 0.34}$
RWARE	tiny-2ag	20.85$_{\pm 0.51}$	19.70$_{\pm 1.22}$	13.70$_{\pm 2.07}$	10.93$_{\pm 2.42}$	12.28$_{\pm 6.18}$	17.06$_{\pm 1.69}$	**21.17**$_{\pm 1.31}$
	tiny-2ag-hard	**17.84**$_{\pm 0.59}$	16.61$_{\pm 1.04}$	9.23$_{\pm 0.93}$	9.61$_{\pm 3.25}$	13.60$_{\pm 0.76}$	13.44$_{\pm 2.55}$	15.93$_{\pm 0.78}$
	tiny-4ag	34.13$_{\pm 1.83}$	31.02$_{\pm 2.05}$	14.42$_{\pm 1.22}$	17.84$_{\pm 2.29}$	26.29$_{\pm 3.03}$	28.19$_{\pm 1.07}$	**43.56**$_{\pm 2.83}$
	tiny-4ag-hard	26.30$_{\pm 1.44}$	20.49$_{\pm 2.76}$	11.31$_{\pm 0.93}$	16.66$_{\pm 2.64}$	19.01$_{\pm 1.38}$	20.54$_{\pm 12.64}$	**30.97**$_{\pm 1.74}$
	small-4ag	12.48$_{\pm 1.54}$	15.09$_{\pm 0.75}$	11.07$_{\pm 0.85}$	9.64$_{\pm 2.56}$	10.52$_{\pm 0.79}$	**18.27**$_{\pm 0.56}$	16.47$_{\pm 8.71}$
	small-4ag-hard	**13.33**$_{\pm 0.43}$	11.48$_{\pm 0.43}$	7.56$_{\pm 1.08}$	6.50$_{\pm 1.15}$	9.44$_{\pm 0.37}$	9.68$_{\pm 3.37}$	12.02$_{\pm 1.26}$
	medium-4ag	**12.86**$_{\pm 0.82}$	11.46$_{\pm 1.22}$	8.22$_{\pm 0.59}$	4.04$_{\pm 0.67}$	7.82$_{\pm 3.41}$	7.62$_{\pm 4.04}$	12.74$_{\pm 1.52}$
	medium-4ag-hard	**9.08**$_{\pm 0.37}$	8.49$_{\pm 0.57}$	4.81$_{\pm 0.83}$	3.75$_{\pm 0.67}$	2.80$_{\pm 2.92}$	4.64$_{\pm 2.68}$	6.79$_{\pm 1.41}$
	medium-6ag	**14.96**$_{\pm 1.70}$	14.78$_{\pm 3.46}$	8.49$_{\pm 1.13}$	4.99$_{\pm 0.86}$	12.13$_{\pm 0.56}$	13.32$_{\pm 0.66}$	12.97$_{\pm 1.08}$
	large-4ag	**9.35**$_{\pm 0.48}$	7.63$_{\pm 1.03}$	5.00$_{\pm 0.57}$	0.61$_{\pm 0.57}$	3.02$_{\pm 2.38}$	4.61$_{\pm 0.27}$	6.22$_{\pm 1.83}$
	large-4ag-hard	**5.13**$_{\pm 1.10}$	4.56$_{\pm 0.67}$	2.25$_{\pm 1.58}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.01}$	2.28$_{\pm 1.98}$	3.46$_{\pm 1.89}$
	large-8ag	14.43$_{\pm 0.65}$	10.40$_{\pm 0.55}$	7.68$_{\pm 0.73}$	0.00$_{\pm 0.00}$	8.35$_{\pm 0.70}$	**14.72**$_{\pm 0.84}$	11.01$_{\pm 0.54}$
	large-8ag-hard	**11.95**$_{\pm 0.63}$	8.66$_{\pm 0.64}$	4.32$_{\pm 3.02}$	0.00$_{\pm 0.00}$	3.38$_{\pm 3.27}$	9.07$_{\pm 0.75}$	9.22$_{\pm 0.50}$
	xlarge-4ag	**7.23**$_{\pm 0.40}$	5.74$_{\pm 0.75}$	3.02$_{\pm 1.32}$	0.00$_{\pm 0.00}$	3.73$_{\pm 1.26}$	4.71$_{\pm 0.46}$	3.76$_{\pm 2.42}$
	xlarge-4ag-hard	0.17$_{\pm 0.22}$	0.31$_{\pm 0.68}$	0.12$_{\pm 0.36}$	0.00$_{\pm 0.00}$	0.00$_{\pm 0.00}$	0.39$_{\pm 1.07}$	**0.70**$_{\pm 1.46}$
SMAX	2s3z	**2.00**$_{\pm 0.00}$	**2.00**$_{\pm 0.00}$	1.99$_{\pm 0.01}$	1.99$_{\pm 0.01}$	2.00$_{\pm 0.00}$	1.99$_{\pm 0.00}$	2.00$_{\pm 0.00}$
	3s5z	**2.00**$_{\pm 0.00}$	1.99$_{\pm 0.02}$	1.98$_{\pm 0.02}$	1.98$_{\pm 0.01}$	2.00$_{\pm 0.00}$	2.00$_{\pm 0.00}$	2.00$_{\pm 0.01}$
	3s_vs_5z	**2.00**$_{\pm 0.00}$	1.98$_{\pm 0.01}$	1.96$_{\pm 0.02}$	1.94$_{\pm 0.02}$	1.98$_{\pm 0.01}$	1.96$_{\pm 0.01}$	1.98$_{\pm 0.01}$
	3s5z_vs_3s6z	**1.98**$_{\pm 0.00}$	1.95$_{\pm 0.02}$	1.88$_{\pm 0.05}$	1.44$_{\pm 0.14}$	1.90$_{\pm 0.06}$	1.93$_{\pm 0.03}$	1.94$_{\pm 0.02}$
	5m_vs_6m	**1.94**$_{\pm 0.01}$	1.15$_{\pm 0.48}$	0.90$_{\pm 0.33}$	0.71$_{\pm 0.03}$	1.05$_{\pm 0.52}$	1.60$_{\pm 0.49}$	1.47$_{\pm 0.57}$
	6h_vs_8z	**2.00**$_{\pm 0.00}$	2.00$_{\pm 0.00}$	1.99$_{\pm 0.01}$	1.99$_{\pm 0.00}$	2.00$_{\pm 0.01}$	1.99$_{\pm 0.01}$	2.00$_{\pm 0.00}$
	10m_vs_11m	**1.98**$_{\pm 0.00}$	1.98$_{\pm 0.02}$	1.77$_{\pm 0.26}$	0.94$_{\pm 0.05}$	1.28$_{\pm 0.09}$	1.86$_{\pm 0.23}$	1.80$_{\pm 0.29}$
	27m_vs_30m	1.99$_{\pm 0.00}$	1.99$_{\pm 0.01}$	1.99$_{\pm 0.01}$	1.74$_{\pm 0.06}$	1.81$_{\pm 0.11}$	1.87$_{\pm 0.11}$	**2.00**$_{\pm 0.00}$
	smacv2_5_units	**1.83**$_{\pm 0.02}$	1.78$_{\pm 0.03}$	1.74$_{\pm 0.04}$	1.60$_{\pm 0.03}$	1.70$_{\pm 0.03}$	1.77$_{\pm 0.02}$	1.72$_{\pm 0.04}$
	smacv2_10_units	**1.70**$_{\pm 0.02}$	1.68$_{\pm 0.07}$	1.53$_{\pm 0.08}$	1.33$_{\pm 0.09}$	1.70$_{\pm 0.03}$	1.62$_{\pm 0.05}$	1.52$_{\pm 0.07}$
	smacv2_20_units	**1.64**$_{\pm 0.02}$	1.56$_{\pm 0.05}$	—	—	1.42$_{\pm 0.06}$	1.49$_{\pm 0.06}$	1.47$_{\pm 0.10}$

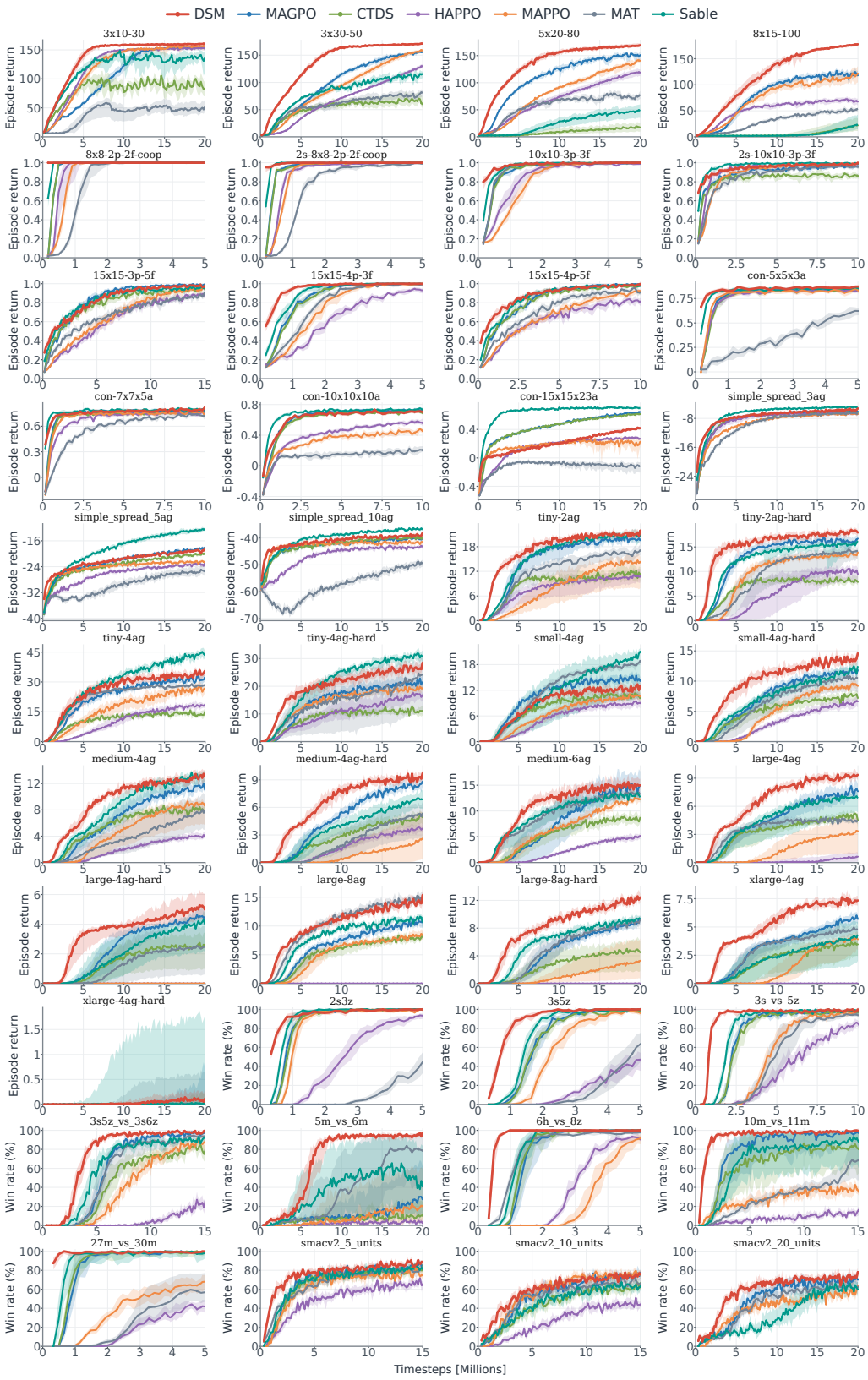


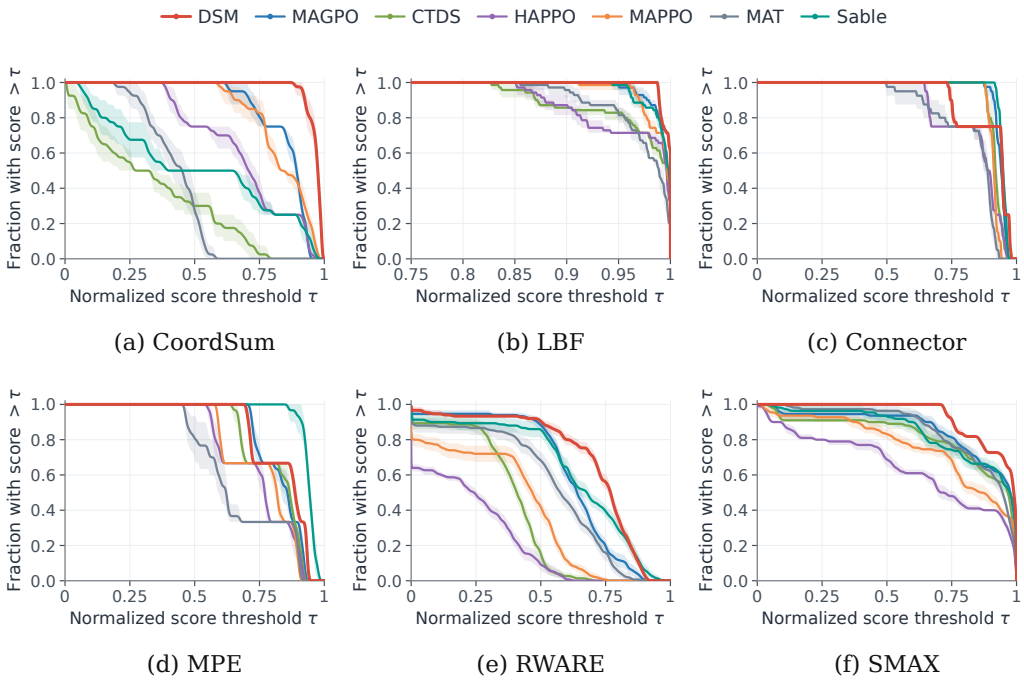
Figure 9: *Learning curves on all 44 tasks.* Curves show episode-return IQMs, or win-rate IQMs for SMAX, over ten seeds with shaded 95% bootstrap confidence intervals.

A.2 Learning Curves for All Tasks

Figure 9 traces learning on all 44 tasks, allowing the benchmark aggregates to be examined at the task level. The vertical axes report episode return, or win rate for SMAX, so each curve records both learning speed and attained performance.

A.3 Performance Profiles

The performance profiles in Figure 10 complement aggregate scores with the distribution of outcomes across tasks and runs. At each normalized score threshold, a curve gives the fraction of task–run outcomes that exceed it.



(a) CoordSum (b) LBF (c) Connector

(d) MPE (e) RWARE (f) SMAX

Figure 10: *Performance profiles by environment suite.* Curves show the fraction of normalized task–run scores above each threshold; shading gives 95% bootstrap intervals over ten seeds.

B Extended Background

This appendix outlines the policy and value-learning formulations most relevant to DSM, using the observation, history, and action notation from the main text. The superscripts "old" and "new" identify policies before and after an update. Where an expression isolates the policy objective, auxiliary value, entropy, and normalization terms are omitted.

B.1 MAT: Autoregressive Joint Decisions

MAT encodes joint observations with a Transformer and uses a masked decoder to generate each action conditioned on the preceding actions (Wen et al., 2022). Writing the encoded sequence as $\hat{o}_t^{1:N} = (\hat{o}_t^1, \dots, \hat{o}_t^N)$, its autoregressive policy is

$$\begin{aligned} \hat{o}_t^{1:N} &= \mathrm{Encoder}(o_t^{1:N}), \\ q_\phi(a_t^{1:N} \mid o_t^{1:N}) &= \prod_{i=1}^{N} q_\phi^i(a_t^i \mid \hat{o}_t^{1:N}, a_t^{<i}). \end{aligned} \tag{B.1}$$

The conditional policies are optimized with a clipped objective:

$$\begin{aligned} \rho^i_{t,\mathrm{MAT}} &= \frac{q^i_\phi(a^i_t \mid o^{1:N}_t, a^{<i}_t)}{q^i_{\phi_{\mathrm{old}}}(a^i_t \mid o^{1:N}_t, a^{<i}_t)}, \\ \mathcal{L}_{\mathrm{MAT}} &= -\mathbb{E}_{q_{\phi_{\mathrm{old}}}}\left[\frac{1}{N}\sum_{i=1}^{N}\min\Big(\rho^i_{t,\mathrm{MAT}}\hat{A}_t, \mathrm{clip}(\rho^i_{t,\mathrm{MAT}}, 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\right]. \end{aligned} \tag{B.2}$$

Here $\hat{A}_t$ estimates the team advantage, and each conditional probability uses the encoder with the corresponding policy parameters. The decoder generates actions sequentially at execution time, but recorded action prefixes allow the conditional probabilities to be evaluated in parallel during optimization. This factorization makes dependencies within the joint action explicit.

B.2 Sable: Joint Sequences with Temporal Memory

Sable combines joint observation encoding and autoregressive action generation with retention-based temporal memory (Mahjoub et al., 2025). For one encoder head within an episode, its matrix accumulation and query readout take the following form, with normalization and gating omitted:

$$\begin{aligned} h^{\mathrm{enc}}_t &= \lambda_R h^{\mathrm{enc}}_{t-1} + \sum_{i=1}^{N}(k^i_t)^\top v^i_t, \\ y^i_t &= q^i_t h^{\mathrm{enc}}_t. \end{aligned} \tag{B.3}$$

The row vectors q^i_t, k^i_t, v^i_t are query, key, and value projections of the agents' current observation embeddings. All current observations contribute to the matrix before the encoder reads it. The head's temporal decay is λ_R, and its state resets at episode boundaries. The decoder also applies causal retention to actions and cross-retention with the encoded observations. DSM adapts the accumulation and readout to a local sequence whose positions contain persistent coordination tokens and one local input, rather than observations from the whole team.

B.3 MAPPO: Local Policies with a Centralized Critic

MAPPO trains decentralized policies with PPO and estimates advantages using a centralized value function (Yu et al., 2022). For a shared-parameter recurrent policy, the clipped policy objective is

$$\begin{aligned} \rho^i_{t,\mathrm{MAPPO}} &= \frac{\pi_\theta(a^i_t \mid \tau^i_t)}{\pi_{\theta_{\mathrm{old}}}(a^i_t \mid \tau^i_t)}, \\ \mathcal{L}_{\mathrm{MAPPO}} &= -\mathbb{E}_{\pi_{\theta_{\mathrm{old}}}}\left[\frac{1}{N}\sum_{i=1}^{N}\min\Big(\rho^i_{t,\mathrm{MAPPO}}\hat{A}^i_t, \mathrm{clip}(\rho^i_{t,\mathrm{MAPPO}}, 1-\epsilon, 1+\epsilon)\hat{A}^i_t\Big)\right]. \end{aligned} \tag{B.4}$$

The advantage $\hat{A}^i_t$ is computed from centralized value estimates. The critic may use global state and agent-specific information available during training, while each deployed actor conditions on τ^i_t. Parameter sharing is an architectural choice, separate from the CTDE information constraint. In the integration in §4.5, the critic's internal representation also supplies token targets.

B.4 HAPPO: Sequential Policy Updates

HAPPO accounts for previously updated agents when optimizing the next policy (Kuba et al., 2022). For an update order $i_1, \ldots, i_N$, its practical surrogate reweights the team-advantage estimate with the already-updated policies:

$$
\begin{aligned}
M_t^{(m)} &= \left(\prod_{j<m} \frac{\pi_{\text{new}}^{i_j}(a_t^{i_j} \mid \tau_t^{i_j})}{\pi_{\text{old}}^{i_j}(a_t^{i_j} \mid \tau_t^{i_j})} \right) \hat{A}_t, \\
\rho_{t,\text{HAPPO}}^{i_m} &= \frac{\pi_{\theta^{i_m}}^{i_m}(a_t^{i_m} \mid \tau_t^{i_m})}{\pi_{\text{old}}^{i_m}(a_t^{i_m} \mid \tau_t^{i_m})}, \\
\mathcal{L}_{\text{HAPPO}}^{i_m} &= -\mathbb{E}_{\pi_{\text{old}}} \left[\min\left(\rho_{t,\text{HAPPO}}^{i_m} M_t^{(m)}, \text{clip}(\rho_{t,\text{HAPPO}}^{i_m}, 1-\epsilon, 1+\epsilon) M_t^{(m)} \right) \right].
\end{aligned}
\tag{B.5}
$$

Here $M_t^{(m)}$ is the reweighted surrogate signal, with the empty product equal to one when $m = 1$. While agent i_m is optimized, the policies of agents $i_1, \ldots, i_{m-1}$ remain fixed at their updated values. Unlike MAT's autoregressive action conditioning, this order governs training updates rather than action exchange at execution time. HAPPO's decentralized policies need not share parameters.

B.5 QMIX: Monotonic Value Factorization

QMIX learns a centralized joint action-value function by combining local utilities through a state-conditioned monotonic mixer (Rashid et al., 2018):

$$
\begin{aligned}
Q_{\text{tot}}(\tau_t, a_t^{1:N}, s_t) &= f_{\text{mix}}\big(Q_1(\tau_t^1, a_t^1), \ldots, Q_N(\tau_t^N, a_t^N); s_t\big), \\
\frac{\partial Q_{\text{tot}}}{\partial Q_i} &\geq 0, \qquad i = 1, \ldots, N.
\end{aligned}
\tag{B.6}
$$

By monotonicity, locally greedy actions jointly maximize the joint value:

$$
\begin{aligned}
& a_t^{i,*} \in \arg\max_{a^i} Q_i(\tau_t^i, a^i), \quad i = 1, \ldots, N, \\
\implies \quad & (a_t^{1,*}, \ldots, a_t^{N,*}) \in \arg\max_{a} Q_{\text{tot}}(\tau_t, a, s_t).
\end{aligned}
\tag{B.7}
$$

The joint value is trained by temporal-difference regression:

$$
\begin{aligned}
y_t^{\text{tot}} &= r_t + \gamma \max_{a'} Q_{\text{tot}}^{-}(\tau_{t+1}, a', s_{t+1}), \\
\mathcal{L}_{\text{QMIX}} &= \mathbb{E}\Big[\big(Q_{\text{tot}}(\tau_t, a_t^{1:N}, s_t) - y_t^{\text{tot}} \big)^2 \Big].
\end{aligned}
\tag{B.8}
$$

The constraint in Eq. (B.6) therefore permits decentralized greedy action selection through Eq. (B.7). In Eq. (B.8), Q_{tot}^{-} denotes the target network, and the bootstrap term is omitted at terminal transitions. The mixer is used during learning, not to supply additional observations to local utilities at execution. Our QMIX integration in §4.5 uses its internal representations as targets for coordination tokens while preserving local action selection.

B.6 CTDS: Centralized Teacher with Decentralized Student

CTDS transfers centralized value estimates to decentralized utility networks (Zhao et al., 2024). Its teacher estimates each agent's action values from a history $\hat{\tau}_t^i$ of full observations and its own past actions, and combines them through a state-conditioned mixer trained by temporal-difference learning. The student has separate parameters and receives only the local history τ_t^i. For each training sample, it regresses the teacher's values over the action space:

$$
\mathcal{L}_{\text{CTDS}} = \sum_{i=1}^{N} \sum_{a \in \mathcal{A}_i} \Big(Q_i(\tau_t^i, a) - \hat{Q}_i(\hat{\tau}_t^i, a) \Big)^2. \tag{B.9}
$$

Here $\hat{Q}_i$ and Q_i are the teacher and student utilities, respectively. The teacher generates training trajectories, while execution uses each student's locally greedy action.

B.7 Representation and Execution Interfaces

Table 3 distinguishes the representations used for decisions from the information available at execution and the targets supplied during training. Here E_t^i denotes locally observed entity embeddings, while $f_t^{i,\text{cent}}$ and $\hat{f}_t^i$ denote PTDE's centralized feature and local prediction. For DSM, the policy head reads z_t^i, while $C_t^{i,1:K}$ and h_t^i are the coordination tokens and retention matrix state in Figure 2; $\hat{C}_t^{i,1:K}$ contains the projected targets in Eq. (8). Policy and feature guidance shape the local actor during training without making the teacher's inputs available at deployment.

Table 3: **Decision representations, execution information, and distillation targets.** $x \rightsquigarrow y$ denotes training-time supervision of y by x; a dash denotes no explicit distillation target in the listed formulation. Symbols are defined in Appendix B.7.

Method	Decision representation	Execution information	Distillation target
MAT / Sable	Joint observation features; temporal state in Sable	$\tau_t, a_t^{<i}$	—
MAPPO / HAPPO	Local actor representation	τ_t^i	—
QMIX	Local utility $Q_i(\tau_t^i, \cdot)$	τ_t^i	—
CTDS	Student utility $Q_i(\tau_t^i, \cdot)$	τ_t^i	$\hat{Q}_i(\hat{\tau}_t^i, \cdot) \rightsquigarrow Q_i(\tau_t^i, \cdot)$
UPDeT / TransfQMix	E_t^i, h_{t-1}^i	τ_t^i	—
MAGPO	Local recurrent representation	τ_t^i	$q_\phi^i \rightsquigarrow \pi_\theta^i$
PTDE	$h_t^i, \hat{f}_t^i$	τ_t^i	$f_t^{i,\text{cent}} \rightsquigarrow \hat{f}_t^i$
DSM	$C_t^{i,1:K}, h_t^i$	τ_t^i	$\hat{C}_t^{i,1:K} \rightsquigarrow C_t^{i,1:K}$; $q_\phi^i \rightsquigarrow \pi_\theta^i$

C Environment Settings and Evaluation Protocol

The six cooperative benchmarks below comprise the 44 tasks reported in Appendix A (Li et al., 2026; Bonnet et al., 2024; Rutherford et al., 2024; Mahjoub et al., 2025).

Experiments use Mava[2], with benchmark implementations from Jumanji[3] and JaxMARL[4].

C.1 CoordSum

Task and reward. Agents select integers that must sum to a shared target. A history-based guesser predicts the first agent's action for that target, discouraging a fixed response pattern. The team receives 0 for an incorrect sum, 1 for a correct sum when the guesser correctly predicts the action, and 2 for a correct sum when the guesser fails (Li et al., 2026).

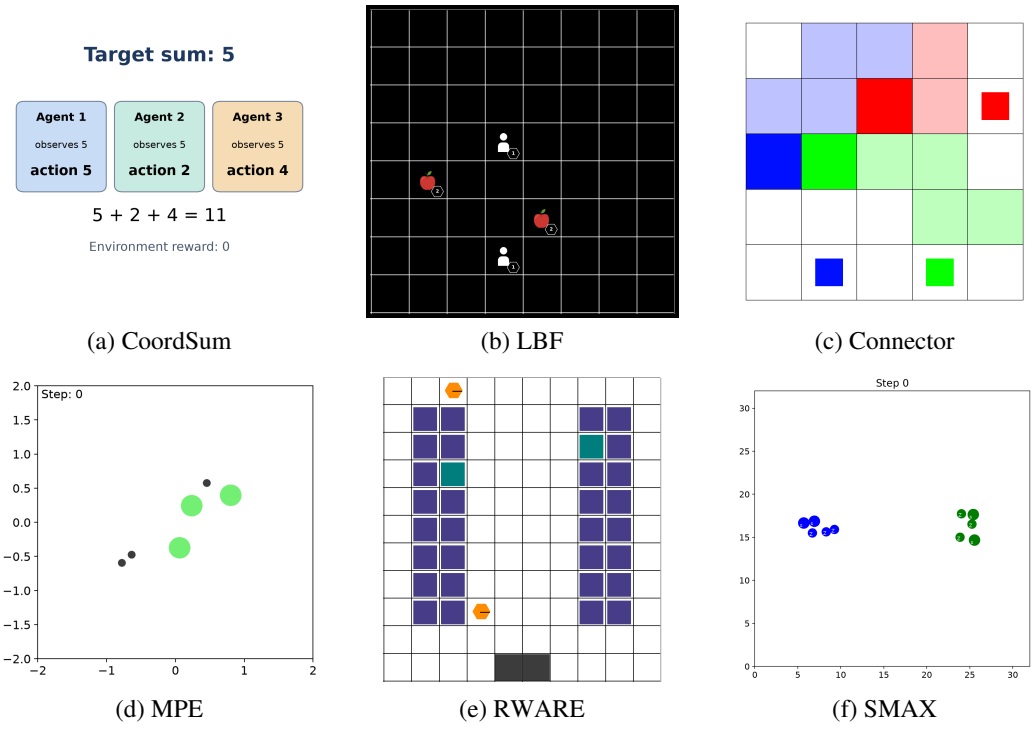
Observations and actions. All agents observe the current target and timestep, then choose an integer from the task's discrete action set. In `CoordSum-<agents>x<actions>-<max-target>`, the three fields specify team size, number of actions, and maximum target. We evaluate `3x10-30`, `3x30-50`, `5x20-80`, and `8x15-100`.

C.2 Level-Based Foraging

Task and reward. Both agents and food items have levels. Adjacent agents can jointly load an item when their combined level meets its requirement, earning reward for successful collection (Christianos et al., 2020; Bonnet et al., 2024).

Observations and actions. Observations describe the positions and levels of visible agents and food items. The six actions comprise four cardinal moves, no operation, and loading. In the benchmark

[2] https://github.com/instadeepai/Mava
[3] https://github.com/instadeepai/jumanji
[4] https://github.com/FLAIROx/JaxMARL



(a) CoordSum (b) LBF (c) Connector

(d) MPE (e) RWARE (f) SMAX

Figure 11: Examples from the six cooperative benchmarks: (a) an actual CoordSum transition (`3x10-30`); (b) Level-Based Foraging (`2s-8x8-2p-2f-coop`); (c) Connector (`con-5x5x3a`); (d) MPE (`simple_spread_3ag`); (e) RWARE (`tiny-2ag`); and (f) SMAX (`2s3z`).

naming convention, `2s` restricts sight to a 5×5 neighborhood, and `coop` identifies tasks that require cooperative collection. The remaining fields specify grid size, the number of players (`p`), and food items (`f`) (Mahjoub et al., 2025).

Tasks. We use `8x8-2p-2f-coop`, `2s-8x8-2p-2f-coop`, `10x10-3p-3f`, `2s-10x10-3p-3f`, `15x15-3p-5f`, `15x15-4p-3f`, and `15x15-4p-5f`.

C.3 Connector

Task and reward. Each agent extends a path from its starting cell to an assigned destination. Because occupied path cells cannot be crossed, a locally short route can block another agent. Completing a connection yields $+1$; an unconnected agent incurs a step cost of -0.03, and completed agents receive no further reward (Bonnet et al., 2024; Mahjoub et al., 2025). Because episode returns include accumulated step costs, native Connector scores are not connection success rates; §C.7 explains their normalization.

Observations and actions. The agent-centered observation describes nearby paths and targets, together with the agent's position and destination. Actions consist of movement in four cardinal directions or no operation. We evaluate `con-5x5x3a`, `con-7x7x5a`, `con-10x10x10a`, and `con-15x15x23a`; the identifiers specify grid dimensions and team size.

C.4 Multi-Agent Particle Environment

Task and reward. In the cooperative `simple_spread` tasks, agents cover landmarks while avoiding collisions. Reward depends on each landmark's distance to the nearest agent and on collision penalties (Lowe et al., 2017; Rutherford et al., 2024).

Observations and actions. Each agent observes its own motion and the relative positions of landmarks and other agents. Actions control planar movement; the selected tasks have no commu-

nication channel. The tasks `simple_spread_3ag`, `simple_spread_5ag`, and `simple_spread_10ag` contain three, five, and ten agents, respectively, and equally many landmarks.

C.5 Robotic Warehouse

Task and reward. Robots retrieve requested shelves and deliver them to designated workstations, earning a shared reward of one for each delivery. Shared warehouse aisles make coordinated navigation and shelf handling necessary (Papoudakis et al., 2021; Bonnet et al., 2024).

Observations and actions. Local observations describe nearby robots and shelves, along with the observing robot's state. The five actions are no operation, forward motion, left turn, right turn, and loading or unloading a shelf. Task names specify warehouse size and agent count. Under the benchmark convention, the `hard` suffix reduces outstanding requests from N to $N/2$ (Mahjoub et al., 2025).

Tasks. The 15 configurations are `tiny-2ag`, `tiny-2ag-hard`, `tiny-4ag`, `tiny-4ag-hard`, `small-4ag`, `small-4ag-hard`, `medium-4ag`, `medium-4ag-hard`, `medium-6ag`, `large-4ag`, `large-4ag-hard`, `large-8ag`, `large-8ag-hard`, `xlarge-4ag`, and `xlarge-4ag-hard`.

C.6 SMAX

Task and reward. SMAX is a JAX-based combat benchmark rather than the StarCraft II engine. Agents control allied units against an opposing team; training rewards reflect damage and victory, while the main-text comparison reports win rate (Rutherford et al., 2024).

Observations and actions. Each unit receives local information about visible allies and enemies, including their combat attributes. Available actions include movement, stopping, and attacks on available targets. We use eight fixed-composition scenarios: `2s3z`, `3s5z`, `3s_vs_5z`, `3s5z_vs_3s6z`, `5m_vs_6m`, `6h_vs_8z`, `10m_vs_11m`, and `27m_vs_30m`. We also use three randomized-composition scenarios: `smacv2_5_units`, `smacv2_10_units`, and `smacv2_20_units`. All eleven tasks run in SMAX; the `smacv2` prefix identifies a task configuration, not a different simulator.

C.7 Evaluation Protocol

Runs and reporting. Main comparisons use ten random seeds and 20 million environment interactions per task; ablations use five seeds. Evaluation follows the marl-eval protocol[5] (Gorsane et al., 2022; Agarwal et al., 2021).

Scores and uncertainty. Table 1 reports normalized IQMs with 95% stratified bootstrap intervals; SMAX uses win-rate fractions. Table 2 reports native episode-return means and sample standard deviations over ten seeds, including for SMAX. Learning curves and performance profiles use 95% bootstrap confidence intervals; for suite curves, runs are resampled independently within each task. Scores averaged over training are the area under the ablation learning curve divided by its training interval.

Task normalization and aggregation. For each return-based task j, let R_j be one run's mean episode return. We normalize each task–run score before computing the suite IQM:

$$\widetilde{R}_j = \frac{R_j - R_j^{\min}}{R_j^{\max} - R_j^{\min} + 10^{-6}}. \tag{C.1}$$

Each task uses fixed empirical bounds across methods, seeds, and checkpoints. Following task-wise min–max normalization (Mahjoub et al., 2025), we fit these bounds over all methods, seeds, training checkpoints, and absolute evaluations in the current main comparison. Normalized scores are not clipped, and the suite IQM is computed from the pooled normalized task–run scores (Agarwal et al., 2021). SMAX uses win-rate fractions directly.

Connector example. For Connector, Table 2 and Figure 9 retain native returns, whereas Table 1 and Figure 3 use normalized suite aggregates. On `con-15x15x23a`, the reference bounds are

[5] https://github.com/instadeepai/marl-eval

$R_j^{\min} \approx -0.658$ and $R_j^{\max} \approx 0.757$, so a native return of 0.70 maps to $\widetilde{R}_j \approx 0.96$. The Connector suite score then aggregates all four tasks: a suite IQM near 0.95 therefore need not match a native task score near 0.7.

Complementary comparisons. Performance profiles give the fraction of scores above each threshold. Improvement probabilities compare runs on the same task, count ties as half a win, and average over tasks. Figure 1(a) separates exclusive and tied-best means among the five displayed methods, using win rates for SMAX and returns elsewhere. Ties require identical unrounded means. Panel (b) gives the geometric mean of per-task parameter ratios to MAGPO across all 44 tasks. Parameter counts include one shared policy and exclude value heads, optimizer state, and recurrent carry. MAT follows Sable's released network and observation settings, with task-specific overrides where available, and uses MAGPO's configurations for CoordSum (Mahjoub et al., 2025; Li et al., 2026).

D Training Procedure

Algorithm 1 describes DSM training with a centralized SM teacher. The teacher and the actor–projector pair maintain separate optimizer states, and both gradients are evaluated before either parameter update is applied. The teacher policy objective follows MAGPO's per-agent double-clipping and masked-KL update (Li et al., 2026). For readability, the pseudocode suppresses probability conditions: teacher probabilities depend on joint history and the recorded action prefix, whereas actor probabilities depend on local history τ_t^i. The actor objective has no entropy bonus. Table 4 lists the architecture and CTD defaults.

Algorithm 1 DSM training and decentralized execution.

Require: Teacher and value parameters ϕ, actor and projector parameters (θ, ψ), separate optimizers; PPO clip width $\epsilon > 0$, guidance bound $c > 1$, and value and entropy weights c_V, c_H.

Initialize each local memory with learned tokens and zero retention matrices; zero the teacher states.
while the training budget is not exhausted **do**
  Fix the behavior teacher $\phi_b \leftarrow \phi$ for the rollout and save its starting state.
  for each environment timestep **do**
    Apply episode resets; save each actor's input memory and update it from (o_t^i, a_{t-1}^i).
    Sample $a_t^{1:N}$ autoregressively from q_{ϕ_b}, applying legal-action masks.
    Step the environment and record observations, actions, rewards, masks, reset flags, teacher values $V_{\mathrm{old},t}^i$, and behavior probabilities $b_t^i = q_{\phi_b}^i(a_t^i)$.
  end for
  Compute GAE advantages and value targets $\hat{V}_t^i$, masking bootstrap terms at terminal transitions.
  for each optimization epoch and recurrent minibatch **do**
    Recompute teacher features, values V_t^i, and conditional policies q_ϕ^i on the recorded sequence.
    Recompute local actor outputs from the saved input memories, obtaining π_θ^i and $C_t^{i,1:K}$ at every timestep and applying episode resets.
    Normalize advantages within the minibatch and form $\hat{C}_t^{i,1:K} = g_\psi(\mathrm{stopgrad}(\hat{o}_t^i))$.
    Compute $\mathcal{L}_{\mathrm{actor}}$ from Eq. (11), with $\rho_t^i = \pi_\theta^i(a_t^i)/b_t^i$.
    Set $r_t^{g,i} = q_\phi^i(a_t^i)/b_t^i$ and $\delta_t^i = q_\phi^i(a_t^i)/\pi_\theta^i(a_t^i)$.
    Set $\tilde{r}_t^{g,i} = \mathrm{clip}(\delta_t^i, 1/c, c)\, \pi_\theta^i(a_t^i)/b_t^i$ and $m_t^i = \mathbf{1}\{\delta_t^i \notin [1/c, c]\}$.
    Compute the teacher policy loss
    $$\mathcal{L}_{g,\mathrm{policy}} = -\mathbb{E}\Big[\min\Big(r_t^{g,i}\hat{A}_t^i, \mathrm{clip}(\tilde{r}_t^{g,i}, 1-\epsilon, 1+\epsilon)\hat{A}_t^i\Big)\Big].$$
    Set $V_{\mathrm{clip},t}^i = V_{\mathrm{old},t}^i + \mathrm{clip}(V_t^i - V_{\mathrm{old},t}^i, -\epsilon, \epsilon)$ and
    $$\mathcal{L}_V = \tfrac{1}{2}\mathbb{E}\Big[\max\Big((V_t^i - \hat{V}_t^i)^2, (V_{\mathrm{clip},t}^i - \hat{V}_t^i)^2\Big)\Big].$$
    Compute the teacher loss
    $$\mathcal{L}_g = \mathcal{L}_{g,\mathrm{policy}} + \mathbb{E}[m_t^i D_{\mathrm{KL}}(q_\phi^i \| \,\mathrm{stopgrad}(\pi_\theta^i))] + c_V\mathcal{L}_V - c_H\mathbb{E}[H(q_\phi^i)].$$
    Compute $\nabla_\phi \mathcal{L}_g$ and $\nabla_{\theta,\psi}\mathcal{L}_{\mathrm{actor}}$ at the same input parameters; then apply their separate optimizer updates.
  end for
end while
Deploy only the local CTA policies, with each agent using its own observations, previous actions, memory, masks, and reset signals.

E HYPERPARAMETERS AND SEARCH SETTINGS

E.1 DEFAULT DSM CONFIGURATION

Table 4 gives DSM's architecture and CTD defaults; §3 defines the actor and projector, and Algorithm 1 specifies training.

Table 4: **Default architecture and CTD settings.** Only the local actor is deployed; the centralized teacher and projector are used during training.

Component	Parameter	Default
Teacher	Representation dimension	32
	Retention layers / heads	2 / 1
	Decay scaling factor	1.0
	Timestep positional encoding	Enabled
Local actor	Coordination tokens K	4
	Token dimension d	24
	Retention layers / heads	1 / 1
	Observation encoder / policy torso width	128 / 128
	Torso activation	ReLU
	Retention-state RMS limit M	50
	Initial-token parameters	Initialized from $\mathcal{N}(0, 0.02^2)$
Projector	Hidden-layer width	128
	Output normalization	Nonaffine LayerNorm
	Output LayerNorm ε	10^{-4}
CTD objective	Alignment weight λ_a	0.10
	Shared standard-deviation weight λ_v	0.05
	Standard-deviation target σ_0	0.20
	Variance stabilizer ε_v	10^{-4}

E.2 SENSITIVITY STUDY SETTINGS

We vary K, λ_a, d, M on CoordSum-3x30-50, LBF `15x15-4p-3f`, SMAX `3s5z_vs_3s6z`, and RWARE `large-8ag-hard`, using five seeds and Appendix C.7's evaluation protocol. Table 5 gives the sweep values for Figure 6; Table 4 lists the defaults.

Table 5: **Hyperparameter sweep values and reference settings.** Each sweep varies one parameter; the reference settings identify the highlighted curves in Figure 6.

Parameter	Values	Reference
Token count K	$\{1, 2, 4, 8, 16\}$	4
Alignment weight λ_a	$\{0.1, 0.25, 0.5, 1.0, 2.0\}$	1.0
Token dimension d	$\{16, 24, 32, 48, 64\}$	24
Retention RMS limit M	$\{25, 50, 100, 200\}$	50

E.3 PUBLISHED BASELINE SEARCH SPACES

Tables 6–8 reproduce Sable's and MAGPO's algorithm-specific search spaces (Mahjoub et al., 2025; Li et al., 2026); DSM's defaults and sweeps appear in Tables 4 and 5.

Table 6: **Search ranges for MAT and Sable** (Mahjoub et al., 2025; Li et al., 2026).

Parameter	Candidate values
PPO epochs	$\{2, 5, 10, 15\}$
Number of minibatches	$\{1, 2, 4, 8\}$
Entropy coefficient	$\{0.1, 0.01, 0.001, 1\}$
PPO clip ϵ	$\{0.05, 0.1, 0.2\}$
Maximum gradient norm	$\{0.5, 5, 10\}$
Learning rate	$\{10^{-3}, 5\times10^{-4}, 2.5\times10^{-4}, 10^{-4}, 10^{-5}\}$
Model embedding dimension	$\{32, 64, 128\}$
Number of heads	$\{1, 2, 4\}$
Number of blocks	$\{1, 2, 3\}$
Retention-decay scaling (Sable)	$\{0.3, 0.5, 0.8, 1\}$

Table 7: **Search ranges for MAPPO, HAPPO, and MAGPO** (Li et al., 2026).

Parameter	Candidate values
PPO epochs	$\{2, 4, 8\}$
Number of minibatches	$\{2, 4, 8\}$
Entropy coefficient	$\{0, 0.01, 0.00001\}$
PPO clip ϵ	$\{0.05, 0.1, 0.2\}$
Maximum gradient norm	$\{0.5, 5, 10\}$
Value learning rate	$\{10^{-4}, 2.5\times10^{-4}, 5\times10^{-4}\}$
Policy learning rate	$\{10^{-4}, 2.5\times10^{-4}, 5\times10^{-4}\}$
Recurrent chunk size	$\{8, 16, 32, 64, 128\}$
Double clip c (MAGPO)	$\{1.1, 1.2, 1.3, 1.5, 2, 3\}$
RL auxiliary coefficient α (MAGPO)	$\{0, 1, 2, 4, 8\}$

Table 8: **QMIX search ranges reported by Sable** (Mahjoub et al., 2025).

Parameter	Candidate values
Batch size	$\{16, 32, 64, 128\}$
Q-network learning rate	$\{3\times10^{-3}, 3\times10^{-4}, 3\times10^{-5}, 3\times10^{-6}\}$
Replay buffer size	$\{2000, 4000, 8000\}$
Target network update period	$\{100, 200, 400, 800\}$
Mixer network embedding dimension	$\{32, 64\}$