Decentralized Asymmetric DQN: Decentralization without Factorization in Multi-Agent Reinforcement Learning

Rupali Bhati, Anurag Kadkol, Andrea Baisero, Christopher Amato

Keywords: multi-agent reinforcement learning, cooperation, Dec-POMDP

Summary

Existing value-factorization-based centralized training with decentralized execution (CTDE) approaches used to solve cooperative multi-agent tasks often enforce architectural constraints—such as additivity or monotonicity—to ensure consistency between centralized training and decentralized execution. The resulting per-agent utilities end up being structural components of a mixing network and lack a clear interpretation as meaningful value functions. Moreover, the convergence properties of these learning dynamics remain largely uncharacterized. We propose Decentralized Asymmetric Deep Q-Networks (Dec-ADQN), a cooperative multi-agent reinforcement learning method that decouples decentralization from structural value factorization. Dec-ADQN learns a centralized, stateful action-value function during training and projects it onto decentralized per-agent Q-functions via regression. Each per-agent utility corresponds to the conditional expectation of the centralized evaluator given the agent's local information, providing a principled interpretation as an information-constrained value approximation under partial observability. We show that, under exact updates in a finite Dec-POMDP, Dec-ADQN implements decentralized asymmetric policy iteration and converges to person-by-person optimal policies. Empirically, Dec-ADQN strikes a favorable balance between independent and value factorization methods, remaining robust whether tasks are independently decomposable or demand tight inter-agent coordination in the OvercookedV2 and SMAX benchmarks. This work reframes decentralized value learning as projection under information constraints rather than structural decomposition, offering both interpretability and theoretical grounding for CTDE methods.

Contribution(s)

1. **A decentralization method without value factorization.** We introduce Dec-ADQN, a cooperative multi-agent reinforcement learning algorithm that learns decentralized policies without imposing IGM-based structural constraints on the joint value function.
   Context: Prior centralized training with decentralized execution (CTDE) value-based methods, rely on explicit value factorization and structural constraints to ensure consistency between centralized evaluation and decentralized execution.
2. **A decentralized asymmetric policy iteration view with convergence to person-by-person optimality.** Under exact centralized evaluation and projection, we show that Dec-ADQN performs coordinate-wise policy improvement and converges to a person-by-person optimal policy in finite policy spaces.
   Context: While stated under idealized assumptions, we provide a formal characterization of Dec-ADQN's improvement dynamics, showing that the method implements coordinate-wise ascent toward person-by-person optimality rather than structural value factorization.
3. **A projection-based interpretation of decentralized utility (Q_i) functions.** We show that, under fixed targets and exact optimization, each per-agent Q_i learned by Dec-ADQN is the conditional expectation of the centralized value given local information.
   Context: Existing value-factorization methods treat per-agent utilities as structural components of a decomposed joint value; in contrast, our analysis shows that Dec-ADQN utilities are policy-dependent projections rather than algebraic factors.

Decentralized Asymmetric DQN: Decentralization without Factorization in Multi-Agent Reinforcement Learning

Rupali Bhati[1†], **Anurag Kadkol**[1,†], **Andrea Baisero**[2], **Christopher Amato**[1]
{bhati.r, kadkol.a}@northeastern.edu, abaisero@uci.edu,
c.amato@northeastern.edu

[1]**Northeastern University**
[2]**UC Irvine**
[†]indicating equal contribution

Abstract

Centralized training with decentralized execution (CTDE) is a dominant paradigm for cooperative multi-agent reinforcement learning. Most successful value-based CTDE methods rely on value factorization, decomposing a joint value function into per-agent utilities under structural constraints such as additivity or monotonicity. While effective, these utilities are architectural components of a mixing network and lack a principled interpretation as standalone value functions. Moreover, existing approaches provide limited insight into the convergence properties of decentralized learning dynamics. We propose **Decentralized Asymmetric Deep Q-Networks (Dec-ADQN)**, which decouples decentralization from structural value factorization. During training, a centralized, stateful action-value function is learned and used as a regression target for independently trained per-agent Q-functions that are executed in a decentralized manner. We show that these per-agent Q-functions approximate the conditional expectation of the centralized evaluator given each agent's local information. Under exact optimization in a finite Dec-POMDP, we prove that Dec-ADQN implements a form of decentralized asymmetric policy iteration and converges to person-by-person optimal policies. Empirically, Dec-ADQN performs well across both independently decomposable tasks and tasks requiring tight coordination in the OvercookedV2 and SMAX benchmarks. These results suggest that projection-based decentralized learning offers an interpretable and theoretically grounded alternative to structural value factorization in cooperative MARL.

1 Introduction

Multi-agent reinforcement learning (MARL) problems requiring cooperation among agents are commonly observed in several real-world scenarios, including autonomous vehicles (Cao et al., 2012), coordination of robot swarms (Hüttenrauch et al., 2017), etc. A common way to solve such MARL problems is using Centralized Training with Decentralized Execution (CTDE) (Oliehoek et al., 2008; Kraemer & Banerjee, 2016), where the agents have access to privileged information during training, while they do not have such information during execution. Having privileged information during training can help the agents learn behaviors that will lead to better cooperation. Removing this privileged information at execution allows these agents to be decentralized, alleviating the need for a centralized controller, thus making them suitable for a wide range of applications.

A broad range of value-based and actor-critic MARL methods have explored how privileged information can be incorporated during centralized training. For example, centralized critic approaches

such as COMA (Foerster et al., 2018) and MADDPG (Lowe et al., 2017) leverage global state information and joint actions during training, while decentralized execution is maintained through agent-specific policies. In value-based MARL, methods focus on learning centralized action-value functions while enabling decentralized action selection. Value factorization methods, including VDN (Sunehag et al., 2017), QMIX (Rashid et al., 2018), QTRAN (Son et al., 2019), and QPLEX (Wang et al., 2021), achieve this by decomposing a joint action-value function into per-agent utilities under structural constraints such as additivity, monotonicity, or more general mixing architectures. These constraints enforce the Individual-Global-Max (IGM) condition (Son et al., 2019), ensuring that decentralized greedy actions recover the joint maximizer, and ensuring that the joint controller remains consistent with the individual decentralized controllers. However, this design choice leaves two conceptual gaps. First, the per-agent utilities in factorization methods are *structural components* of a mixing architecture rather than principled value functions. Second, existing value-based CTDE methods provide limited theoretical insight into their convergence properties. While IGM ensures consistency between joint and decentralized maximization under structural assumptions, it does not characterize the optimization dynamics or the type of equilibrium to which learning converges.

In this paper, we propose **Decentralized Asymmetric Deep Q-Networks (Dec-ADQN)**, which addresses both issues by decoupling decentralization from structural value factorization. Instead of decomposing a joint value into constrained components, we train a centralized, stateful action-value function purely as a training target and learn per-agent Q-functions via regression. Each per-agent value $Q_i(\tau_i, u_i)$ is not merely a structural factor of the joint value. Rather, it approximates the conditional expectation of the centralized evaluator given the agent's local information. This yields a principled interpretation: decentralized values arise through *double marginalization* over hidden state and other agents' behavior, rather than through architectural constraints. Secondly, we introduce decentralized asymmetric policy iteration and show that in the tabular setting without value function approximation, this process converges to an *person-by-person optimal* (Radner, 1962) policy where no single agent can unilaterally deviate from their policy and improve the joint return. Thus, beyond providing a representation interpretation, Dec-ADQN admits a convergence characterization absent in most value-factorization approaches.

We perform an empirical evaluation of Dec-ADQN on OvercookedV2 (Gessler et al., 2025) and SMAX (Rutherford et al., 2024). Dec-ADQN consistently outperforms strong value-factorization baselines on all OvercookedV2 layouts and remains competitive on most SMAX maps, with lower performance primarily in highly stochastic regimes. Overall, this work reframes decentralized value learning as projection under information constraints rather than structural decomposition, providing both an interpretable representation of per-agent utilities and a principled convergence perspective for cooperative MARL.

2 Background

2.1 Dec-POMDP

Cooperative multi-agent control problems are commonly modeled as Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) (Oliehoek & Amato, 2016). A Dec-POMDP is defined by a tuple $\langle \mathcal{N}, \mathcal{S}, O, \mathcal{U}, T, \mathcal{O}, r, \gamma \rangle$, where $\mathcal{N}$ is the set of $n = |\mathcal{N}|$ agents and $\mathcal{S}$ is the finite set of states with initial state distribution ρ_0. Each agent $i \in \mathcal{N}$ is associated with an action space $\mathcal{U}_i$ and an observation space $\mathcal{O}_i$, and $\mathcal{U} = \mathcal{U}_1 \times \cdots \times \mathcal{U}_n$ and $\mathcal{O} = \mathcal{O}_1 \times \cdots \times \mathcal{O}_n$ denote the joint action space and joint observation space, respectively. At each timestep, each agent $i \in N$ receives observation $o_i \in \mathcal{O}_i$ and chooses an action $u_i \in \mathcal{U}_i$, which forms the joint action $\mathbf{u} = \langle u_1, \ldots, u_n \rangle \in \mathcal{U}$. Then, the environment transitions from state s to state s' according to transition function $T(s, \mathbf{u}, s') = \Pr(s'|s, \mathbf{u})$ and outputs observations $\mathbf{o}' = \langle o_1, \ldots, o_n \rangle \in \mathcal{O}$ as per the observation function, $O(\mathbf{u}, s', \mathbf{o}') = \Pr(\mathbf{o}'|s', \mathbf{u})$ and the joint reward $r : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$. Each agent i learns its individual policy π_i conditioned on its action-observation history $\tau_i \in \mathcal{T}_i \equiv (\mathcal{O}_i \times \mathcal{U}_i)^* \times \mathcal{O}_i$ to jointly maximize the joint return. Thus, the goal is to find a joint policy $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ that

maximizes the expected discounted episodic return $\mathbb{E}_{s_0\sim\rho_0,\boldsymbol{\pi}}[\sum_t \gamma^t r(s_t, \mathbf{u}_t)]$ where γ is the discount factor and t is the time step.

2.2 Value-Based Methods for Decentralized MARL

One of the simplest value-based methods for decentralized MARL is Independent Q-Learning (IQL) (Tampuu et al., 2017), which applies Deep Q-Learning (Mnih et al., 2013) independently to each agent without centralized training or information sharing. Because these agents are learning simultaneously, the environment becomes non-stationary, which often leads to instability and poor coordination. CTDE approaches address this issue by leveraging additional information during learning. Most successful CTDE methods therefore rely on *value factorization*. These approaches learn a centralized joint action-value function during training and decompose it into per-agent utilities that can be used for decentralized execution. The decomposition is designed so that greedy action selection with respect to the per-agent utilities recovers the joint greedy action. This property is typically enforced through the Individual-Global-Max (IGM) condition, $\operatorname{argmax}_{\mathbf{u}\in\mathcal{U}} Q(\boldsymbol{\tau}, \mathbf{u}) = \big(\operatorname{argmax}_{u_i\in\mathcal{U}_i} Q_i(\tau_i, u_i)\big)_{i\in\mathcal{N}}$. To guarantee IGM, factorization methods impose structural constraints on how individual utilities combine to form the joint value. Value Decomposition Networks (VDN) (Sunehag et al., 2017) assume an additive decomposition, $Q(\boldsymbol{\tau}, \boldsymbol{u}) = \sum_{i=1}^{|\mathcal{N}|} Q_i(\tau_i, u_i)$. QMIX (Rashid et al., 2018) generalizes this idea using a nonlinear mixing network constrained to be monotonic in each Q_i, enforcing $\frac{\partial Q}{\partial Q_i} \geq 0$. QPLEX (Wang et al., 2021) further extends this approach using a duplex dueling architecture that separates utilities from a mixing advantage term while maintaining the monotonicity constraint. While these methods have achieved strong empirical performance, their per-agent utilities (Q_i functions) are defined as structural components of a mixing network. Moreover, the optimization dynamics of these approaches are not easily interpretable, as the learned utilities depend on the chosen factorization structure. Dec-ADQN takes a different approach. Instead of decomposing the joint value through structural constraints, we learn a centralized action-value function and train the per-agent Q-functions via regression to match its evaluation under the decentralized policy. This yields per-agent utilities with a clear interpretation: each Q_i approximates the conditional expectation of the centralized value given the agent's local information, providing a principled link between centralized evaluation and decentralized decision making.

2.3 Asymmetric Deep Q-Networks (ADQN).

In single-agent partially observable settings, agents are generally required to learn history-based value functions. In some training frameworks (e.g., training in simulation or in controlled environments), privileged information like the state is also available during the training, which can be exploited to achieve better learning performances. Asymmetric Deep Q-Networks (ADQN) (Baisero et al., 2022) is a single-agent method that leverages this privileged information by training an auxiliary privileged action-value model that provides targets for the standard history-based action-value model that is used at test time. Concretely, ADQN maintains two networks: a privileged evaluator $U(\tau, s, u)$ that has access to the global state s during training, and a Q-function $Q(\tau, u)$ that depends only on the agent's observation history τ. Let $\widehat{\pi}$ denote the greedy policy with respect to $\widehat{Q}$, i.e., $\widehat{\pi}(\tau) = \operatorname{argmax}_u \widehat{Q}(\tau, u)$. Then, the privileged network $\widehat{U}$ is trained via standard temporal-difference learning, and the Q-function is then trained to regress toward targets derived from the centralized evaluator:

$$\mathcal{L}_{\widehat{U}} = \frac{1}{2}\big(\operatorname{stopgrad}(y^{\widehat{U}}) - \widehat{U}(\tau, s, u)\big)^2. \tag{1}$$

$$\mathcal{L}_{\widehat{Q}} = \frac{1}{2}\big(\operatorname{stopgrad}(y^{\widehat{U}}) - \widehat{Q}(\tau, u)\big)^2. \tag{2}$$

where $y^{\widehat{U}} = r + \gamma\, \widehat{U}(\tau u o', s', \widehat{\pi}(\tau u o'))$. The notation $\tau u o'$ represents the updated trajectory at the next time step, defined as the concatenation of the current action-observation history τ, the executed action u, and the subsequent observation o'. This asymmetric setup allows the partially observ-

able Q-function to benefit from richer state information during training while remaining deployable without access to the global state.

3 Decentralized Asymmetric DQN (Dec-ADQN)

Dec-ADQN generalizes ADQN to the multi-agent setting, employing training asymmetry to exploit privileged information available during training in order to train a team of decentralized agents. During training, a centralized privileged network $\widehat{U}(\boldsymbol{\tau}, s, \mathbf{u})$ evaluates the performance of the joint team starting from the given joint action. To simplify notation, let $\widehat{\boldsymbol{\pi}}$ denote the joint policy obtained by acting greedily with respect to the individual values $\widehat{Q}_i$, i.e., $\widehat{\boldsymbol{\pi}}(\boldsymbol{\tau}) \doteq (\mathrm{argmax}_u \widehat{Q}_i(\tau_i, u))_i$. The centralized values $\widehat{U}$ are trained using standard evaluation temporal-difference targets:

$$\mathcal{L}_{\widehat{U}} = \frac{1}{2}\big(\mathrm{stopgrad}(y^{\widehat{U}}) - \widehat{U}(\boldsymbol{\tau}, s, \mathbf{u})\big)^2. \tag{3}$$

where $y^{\widehat{U}} = r + \gamma\, \widehat{U}(\boldsymbol{\tau u o'}, s', \widehat{\boldsymbol{\pi}}(\boldsymbol{\tau u o'}))$. Intuitively, this step learns a centralized critic that estimates the value of joint behaviour using privileged state information. Similar to the notation in ADQN, $\boldsymbol{\tau u o'}$ denotes the joint trajectory at the next time step, formed by concatenating the joint action-observation history $\boldsymbol{\tau}$, the joint action $\mathbf{u}$, and the joint next observation $\mathbf{o}'$. Each agent is then trained to approximate the centralized evaluator. To do so, each per-agent Q-function $Q_i(\tau_i, u_i)$ is trained independently to regress toward the same centralized target:

$$\mathcal{L}_{\widehat{Q}_i} = \frac{1}{2}\big(\mathrm{stopgrad}(y^{\widehat{U}}) - \widehat{Q}_i(\tau_i, u_i)\big)^2. \tag{4}$$

In other words, every agent learns to estimate the centralized evaluation that has privileged information from its own limited information. Thus, Dec-ADQN extends asymmetric training to cooperative multi-agent settings: centralized state information stabilizes learning, while execution remains fully decentralized. Unlike value factorization approaches, the per-agent Q-functions are not constrained to form a structural decomposition of the joint value. Instead, they approximate the centralized evaluator under decentralized information constraints.

4 Idealized Asymmetric Policy Iteration for Dec-ADQN

This section clarifies two complementary aspects of Dec-ADQN: (i) its optimization structure under exact updates, and (ii) the precise object represented by the per-agent Q_i functions.

4.1 Optimization: Decentralized Asymmetric Policy Iteration

We first analyze an idealized variant of Dec-ADQN in which evaluation and projection are performed exactly and sequentially on finite policy spaces.

Theorem 1 (Decentralized Asymmetric Policy Iteration)**.** *Consider a finite-horizon Dec-POMDP with finite state, action, and observation spaces. Assume the following procedure:*

1. *Initialize a decentralized joint policy $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$.*
2. ***Joint evaluation:** Train U to convergence so that $U = U^{\boldsymbol{\pi}}$, the exact action-value of the decentralized policy $\boldsymbol{\pi}$.*
3. ***Decentralized projection:** For each agent i, train Q_i to convergence using*

$$\mathcal{L}_{Q_i} = \frac{1}{2}\big(\mathrm{stopgrad}(y^U) - Q_i(\tau_i, u_i)\big)^2,$$
$$y^U = r + \gamma\, U(\boldsymbol{\tau u o}, s', \boldsymbol{\pi}(\boldsymbol{\tau u o})).$$

4. ***Policy improvement:** Select an agent i randomly and keeping π_{-i} fixed, update $\pi_i'(\tau_i) \in \mathrm{argmax}_{u_i} Q_i(\tau_i, u_i)$.*

5. Repeat steps 2 to 4.

Then each unilateral policy update weakly improves the joint return: $J(\pi_i', \pi_{-i}) \geq J(\pi_i, \pi_{-i})$. *Since the policy space is finite, the procedure converges in finitely many steps to a* person-by-person optimal *policy (Radner, 1962)* $\boldsymbol{\pi}^*$ *satisfying* $\pi_i^* \in \operatorname{argmax}_{\pi_i} J(\pi_i, \pi_{-i}^*) \quad \forall i$.

(Proof in Appendix A.1)

Remark 1 (Link between Q-learning and Policy Iteration)**.** *Theorem 1 shows that, under exact updates, Dec-ADQN implements decentralized asymmetric policy iteration:* $U = U^\pi$ *gives exact policy evaluation, and* $Q_i = \mathbb{E}[U^\pi \mid \tau_i, u_i]$ *yields an exact projection. In practice, both are learned approximately due to the neural parameterization and interleaved optimization:* U *is learned via TD updates, while* Q_i *is learned by regression to targets derived from* U*, with policy improvement based on the learned* Q_i*. Thus, Theorem 1 characterizes the idealized optimization structure underlying Dec-ADQN. As in approximate policy iteration, evaluation is not solved exactly but learned jointly with policy improvement, so monotonic improvement and convergence guarantees do not necessarily hold under function approximation.*

4.2 Representation: What do the Q_i Functions Learn?

We now characterize the value represented by the decentralized Q_i networks when trained against a fixed centralized evaluator.

Proposition 1 (Projection under partial observability and decentralization)**.** *Consider Dec-ADQN with centralized evaluator* $U(\boldsymbol{\tau}, s, \mathbf{u})$ *and per-agent networks* $Q_i(\tau_i, u_i)$*. Assume that the decentralized joint policy* $\boldsymbol{\pi}$ *generating targets is fixed to a given value, and that the* U *model is trained until convergence to the privileged centralized policy values* $U^{\boldsymbol{\pi}}$*. Let* Q_i *be trained to minimize*

$$\mathcal{L}_i(Q_i) = \mathbb{E}\Big[\big(y^U - Q_i(\tau_i, u_i)\big)^2\Big], \tag{5}$$

$$y^U = r + \gamma U(\boldsymbol{\tau' u o}, s', \boldsymbol{\pi}(\boldsymbol{\tau' u o})). \tag{6}$$

Then, Q_i *converges to*

$$Q_i(\tau_i, u_i) = \mathbb{E}\left[y^U \mid \tau_i, u_i\right] \tag{7}$$

$$= \mathbb{E}\left[U^{\boldsymbol{\pi}}(\boldsymbol{\tau}, s, \boldsymbol{u}) \mid \tau_i, u_i\right] \tag{8}$$

$$= Q_i^{\boldsymbol{\pi}}(\tau_i, u_i). \tag{9}$$

(Proof in Appendix A.2).

Interpretation. Proposition 1 shows that each Q_i is not a structural factor of the centralized value U. Rather, it is a *policy-dependent projection*: the conditional expectation of the centralized evaluation given the information available to agent i. Under partial observability, this induces a double marginalization over (i) environmental uncertainty and (ii) other agents' behavior. This distinguishes Dec-ADQN from value-factorization methods which impose algebraic structure on the joint value function. Here, no IGM constraint or fixed mixing rule is required; instead, decentralization arises from projecting centralized evaluations onto local information spaces.

Discussion. Together, Theorem 1 and Proposition 1 provide a coherent interpretation of Dec-ADQN. The theorem characterizes the algorithm as coordinate-wise policy improvement under exact updates, while the proposition clarifies the semantic meaning of the learned Q_i: they are the best L^2 approximations of the centralized value compatible with each agent's information constraints. This characterization is useful for two reasons. First, it explains why the Dec-ADQN training procedure is sensible: the centralized evaluator defines the target value of joint behavior, while each agent learns the best locally computable approximation of that value. As a result, the algorithm can be viewed as learning decentralized surrogates of a centralized value function while preserving decentralized execution. Second, it provides a semantic interpretation of decentralized utilities

that is absent in most value factorization approaches, where per-agent values primarily arise from architectural constraints rather than an information-theoretic principle.

Remark 2 (Variance-Reduced Dec-ADQN)**.** *Inspired by the variance-reduced variant of ADQN proposed by Baisero et al. (2022), one may consider regressing each decentralized action-value function directly to the centralized utility estimate U rather than its Bellman target y^U. While this may reduce the variance of the regression target, using y^U is essential for Bellman consistency and ensures that each Q_i approximates $\mathbb{E}[U^\pi \mid \tau_i, u_i]$, supporting Proposition 1 and the policy iteration interpretation of Dec-ADQN. In contrast, regressing directly to U breaks Bellman consistency and leads to a different fixed point. Since our theoretical analysis is built around the Bellman-consistent formulation, we leave this alternative variant for future work.*

5 Experiment Setup

We evaluate Dec-ADQN on cooperative control tasks using two environments from the JaxMARL (Rutherford et al., 2024) suite: OvercookedV2 and SMAX, which provide complementary coordination challenges while enabling fast training through their JAX implementation.

5.1 OvercookedV2

The Overcooked environment (Carroll et al., 2019) is commonly used to evaluate fully observable cooperative tasks, in which every agent observes the entire state and cooperates to maximize a shared reward. The objective here is to prepare and deliver as many soups as possible within a fixed time horizon. JaxMARL (Rutherford et al., 2024) implements this environment in JAX and also introduces OvercookedV2 (Gessler et al., 2025), which extends previous versions by incorporating partial observability and stochasticity, resulting in more challenging coordination problems. We use OvercookedV2 because these characteristics better reflect the decentralized settings considered in this work. We evaluate all algorithms on the five original Overcooked layouts shown in Fig. 1, which span a range of coordination difficulty from weakly to tightly coupled tasks. We believe these layouts are sufficient to support our main claims.

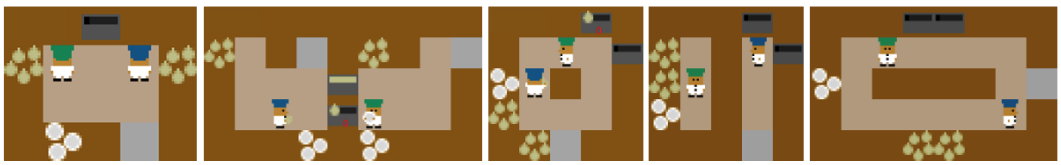


Figure 1: Experiment layouts of the Overcooked environment as introduced in Carroll et al. (2019). From left to right: (i) Cramped Room: introduces limited space, making agents prone to getting in the way of the other agent, (ii) Asymmetric Advantages: allows agents to complete the task independently or coordinate to come up with high level strategies allowing them to have role specialization, (iii) Coordination Ring: agents share a single path (a ring) to travel in the environment, (iv) Forced Coordination: demands cooperation, as no single agent can complete soup deliveries independently, (v) Counter Circuit: requires agents to pass ingredients over a counter, creating a challenging coordination problem.

5.2 SMAX

The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) and its extension SMACv2 (Ellis et al., 2023) have been widely used as benchmarks for evaluating MARL algorithms. These tasks focus on decentralized micromanagement challenges, where each of the units is controlled by an independent, learning agent that has to act based only on local observations, while the opponent's units are controlled by the built-in StarCraft II AI. SMAX (Rutherford et al., 2024) reimplements SMAC and SMACv2 in JAX, allowing faster training time and also includes a

stronger heuristic AI opponent. The SMAX maps are grouped into four categories: (i) Symmetric maps feature both allied and enemy teams with the same number and type of units, (ii) Asymmetric maps introduce imbalance by giving the enemy team more units than the allied team, (iii) the micro-trick maps are specifically designed to test targeted micromanagement strategies, and finally (iv) the SMACv2 maps extend the challenge by introducing randomized starting positions and unit types.

5.3 Algorithms Considered

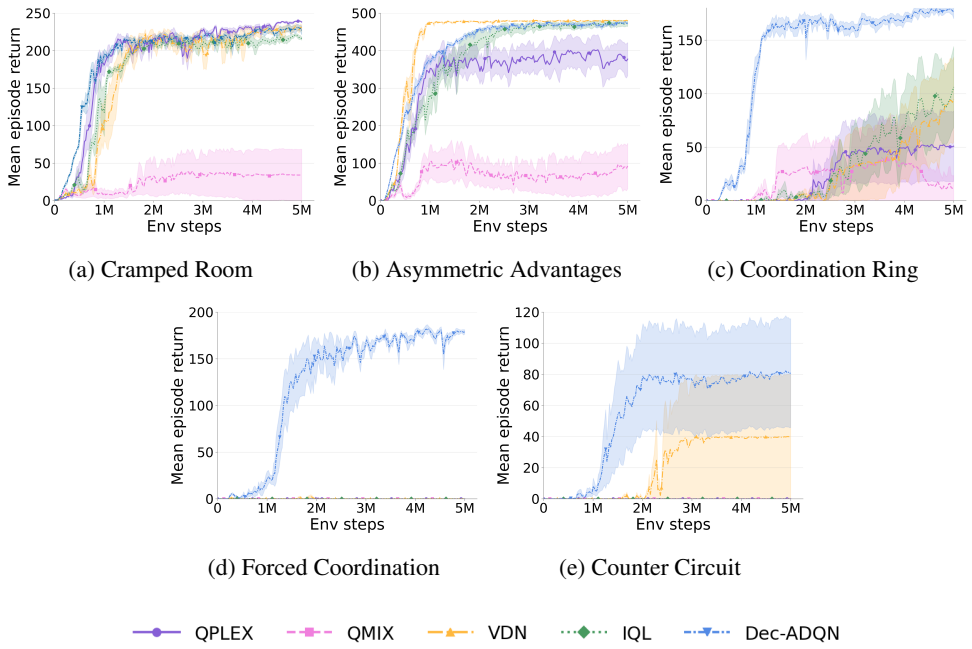
In both environments, we compare Dec-ADQN against four representative value-based MARL baselines spanning the main design spectrum of decentralized learning: IQL, which learns independently without centralized training; VDN, the simplest additive value factorization method; QMIX, which introduces a nonlinear monotonic mixing network; and QPLEX, the most expressive value-factorization approach that preserves the IGM condition. It is also worth noting here that we focus on value-based and not policy-gradient based baselines because Dec-ADQN is itself a value-based method and is most directly comparable to value-decomposition approaches. Policy-gradient methods optimize decentralized policies using actor–critic objectives rather than action-value learning. As a result, they differ substantially in optimization dynamics, credit assignment mechanisms, and sample efficiency. Our goal in this work is therefore to isolate the effect of removing structural value factorization within the value-based CTDE paradigm. All algorithms were evaluated across five random seeds to account for stochasticity in training and environment dynamics. We use standard error as a measure of statistical confidence, as it quantifies the uncertainty in the estimated mean performance across independent runs. The shaded regions/error bars in our plots, therefore, reflect the variability due to stochastic training and initialization. For OvercookedV2, we report the mean episodic return, which measures the total cooperative reward achieved by the agents. For SMAX, we report the mean win rate, the standard metric for StarCraft-style combat tasks that reflects the fraction of battles won. The full details of the architecture and hyperparameters used are provided in Appendix B and E, respectively.

6 Results

6.1 OvercookedV2

Fig. 2 shows the learning curves on the five OvercookedV2 environment layouts. OvercookedV2 introduces partial observability (agents observe only a small egocentric window) making coordination difficult without direct observation of teammates. We observe that Dec-ADQN (blue curve) consistently matches or outperforms all baselines across the five OvercookedV2 layouts.

(i) Cramped Room and Asymmetric Advantages: Compared to other layouts, these two provide ample movement space and require minimal ingredient passing. The task can largely be completed independently, reducing coordination demands and emphasizing spatial and task reasoning. As a result, coordination advantages between algorithms are less pronounced, and most methods—except QMIX—achieve high returns. *(ii) Coordination Ring, Forced Coordination, and Counter Circuit:* These layouts require tightly coupled coordination under partial observability. In Coordination Ring, narrow passages and shared paths require agents to implicitly coordinate movement with limited local observations. Forced Coordination imposes strong role asymmetry: one agent accesses ingredients while the other performs cooking and delivery, so neither can complete the task independently. Counter Circuit further introduces multi-step exchanges where agents must pass ingredients across a counter under strict timing and positioning constraints. Across these tasks, Dec-ADQN consistently outperforms the baselines and often converges faster. We attribute this to the centralized evaluator U, which observes the full state during training and can correctly evaluate complementary joint actions and multi-step coordination patterns (e.g., collision-free sequencing or successful ingredient exchanges). Because each Q_i regresses toward this global evaluation, coordinated behaviors receive positive learning signals even when their value is not locally observable. In contrast, structural factorization methods must represent these dependencies through additive or monotonic constraints,



(a) Cramped Room (b) Asymmetric Advantages (c) Coordination Ring

(d) Forced Coordination (e) Counter Circuit

Figure 2: Results for episodic returns on OvercookedV2 evaluated over 5 seeds. The solid/dashed lines indicate the mean episodic return, with the shaded region depicting the standard error.

which can struggle when coordination is sequential, asymmetric, or partially hidden. In Counter Circuit specifically, Dec-ADQN exhibits higher variance across seeds: some runs learn the higher-return strategy of passing ingredients across the counter, while others revert to walking around it. VDN, the only other method to achieve non-zero returns, consistently adopts the latter strategy.

Overall Pattern: (i) *No degradation in simple regimes:* When the coordination structure is weak (Cramped Room, Asymmetric Advantages), Dec-ADQN performs on par with the baselines, (ii) *Clear gains in tightly coupled regimes:* When optimal performance requires sequential or spatial coordination (Forced Coordination, Coordination Ring, Counter Circuit), Dec-ADQN converges faster and achieves higher returns. These results are consistent with the projection interpretation in Proposition 1. Each per-agent Q_i approximates the conditional expectation of a centralized value function given local information. When coordination dependencies are strong but not locally observable, this projection mechanism provides more informative credit assignment than structural value factorization, leading to improved decentralized policies.

6.2 SMAX

Fig. 3 shows the win rates for the baseline algorithms and Dec-ADQN on the SMAX maps. In SMAX, each agent receives an observation vector containing its own features (e.g., health) along with a partial, egocentric view of other unit features within its sight range. The state includes complete ground-truth information for all units, including positions, health, weapon cooldowns, unit types, and previous actions.

(i) Symmetric maps (2s3z, 3s5z): Symmetric maps feature mixed unit types, and the primary challenge here is learning which enemy types are most threatening to each allied unit type and prioritizing targets accordingly. IQL struggles to perform well on these tasks as they require coordinated focus fire and positioning. Dec-ADQN achieves high win rates, converging slightly more slowly than VDN in early training but reaching comparable asymptotic performance, indicating that Dec-ADQN is a suitable algorithm for such moderately complex coordination tasks. *(ii) Asymmetric maps (5m_vs_6m, 10m_vs_11m, 3s5z_vs_3s6z):* On asymmetric maps, the enemy has a numeri-

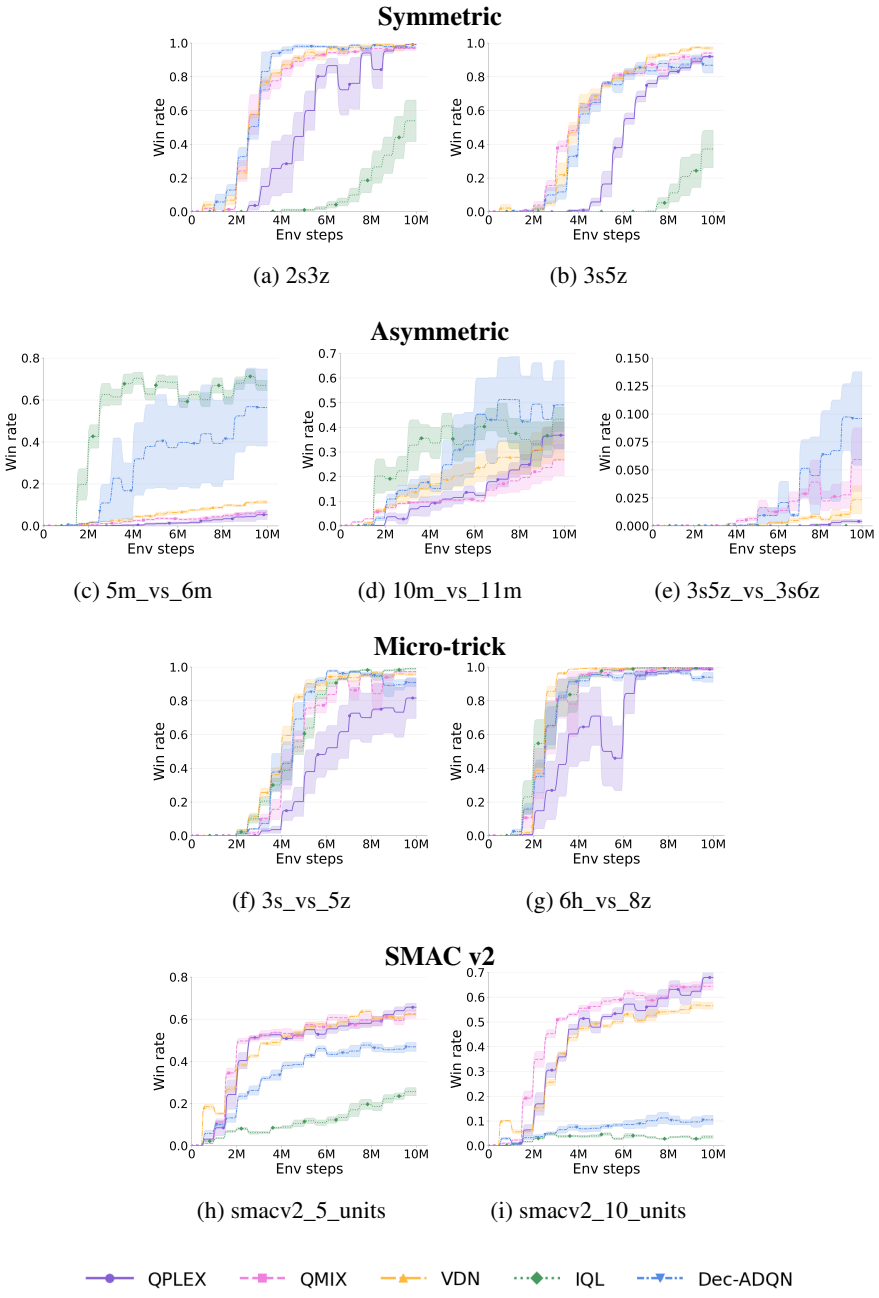


Figure 3: Results for win rates on SMAX maps averaged over 5 seeds per algorithm. The solid/dashed lines indicate the mean win rate, with the shaded region depicting the standard error.

cal advantage. Interestingly, IQL performs strongly on purely homogeneous maps (marines only), where optimal behavior reduces to locally greedy targeting. Nevertheless, on such maps, Dec-ADQN remains competitive, indicating that Dec-ADQN is robust to settings that require minimal coordination and is an improvement over value factorization methods that struggle on such tasks. The 3s5z_vs_3s6z map remains too challenging for any algorithm to solve. *(iii) Micro-trick maps (3s_vs_5z, 6h_vs_8z):* Micro-trick maps emphasize fine-grained micromanagement skills such as kiting and cooldown timing. The comparable performance across methods suggests that perhaps the allied team is too strong for the enemy team, such that succeeding at the task is almost algorithm agnostic (at least for the algorithms considered here). *(iv) SMACv2 maps (smacv2_5_units, smacv2_10_units):* In SMACv2, unit types and starting positions are randomized each episode. Dec-ADQN underperforms value-factorization methods on these maps. This behavior is consistent with Proposition 1 where each $Q_i(\tau_i, u_i)$ must approximate $\mathbb{E}\big[U^\pi(\boldsymbol{\tau}, s, \mathbf{u}) \mid \tau_i, u_i\big]$, which averages over both hidden global state and other agents' private observations. The stochastic initialization in SMACv2 maps increases the diversity of joint configurations that each Q_i must account for, potentially diluting the learning signal. The ablation study in Appendix D supports this interpretation by showing that reducing stochasticity through fixed unit types or starting positions improves Dec-ADQN's performance. However, this ablation does not isolate variance as the sole factor affecting performance, since reduced stochasticity may benefit other methods as well. Rather, we hypothesize that the increased variability of joint configurations presents a greater challenge for Dec-ADQN because each Q_i must approximate a broader conditional expectation. In contrast, value factorization methods enforce structural consistency between joint and local utilities, which can provide an inductive bias that stabilizes learning under such conditions.

Overall Pattern: (i) Dec-ADQN performs strongly in structured coordination tasks where joint evaluation under partial observability is beneficial, (ii) It remains competitive when coordination is nearly decomposable, (iii) Its performance degrades in highly stochastic environments where the conditional variance of the centralized value given local information is large. These results reinforce the theoretical interpretation of Dec-ADQN as a projection-based decentralized control method: its effectiveness depends on how informative local observations are about the centralized value function. When that information is sufficient, asymmetric centralized training provides stable and expressive decentralized policies; when it is highly noisy, the projection becomes less precise.

7 Conclusion

We introduced Dec-ADQN, a decentralized multi-agent reinforcement learning method that decouples decentralization from explicit value factorization. Rather than decomposing a joint value function into structural components, Dec-ADQN learns per-agent value functions as projections of a centralized evaluator onto local information spaces, providing a principled interpretation of decentralized learning. We showed that, in an idealized setting, Dec-ADQN implements a form of decentralized asymmetric policy iteration and converges to person-by-person optimal policies. Empirically, the method performs strongly on coordination-intensive OvercookedV2 layouts and remains competitive on most SMAX maps. We discuss limitations of our approach in Appendix G. Overall, our results suggest that asymmetric centralized training combined with decentralized projection offers a viable alternative to structural value decomposition. We hope this work encourages further investigation into projection-based approaches to decentralized control and deeper analysis of how conditional variance and information constraints shape multi-agent learning.

References

Andrea Baisero, Brett Daley, and Christopher Amato. Asymmetric dqn for partially observable reinforcement learning. In *Uncertainty in Artificial Intelligence*, pp. 107–117. PMLR, 2022.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination, 2012. URL https://arxiv.org/abs/1207.3231.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

Jakob 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.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination, 2025. URL https://arxiv.org/abs/2503.17821.

Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Guided deep reinforcement learning for swarm systems, 2017. URL https://arxiv.org/abs/1709.06011.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016. ISSN 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2016.01.031. URL https://www.sciencedirect.com/science/article/pii/S0925231216000783.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013. URL https://arxiv.org/abs/1312.5602.

F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, May 2008. ISSN 1076-9757. DOI: 10.1613/jair.2447. URL http://dx.doi.org/10.1613/jair.2447.

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

Roy Radner. Team decision problems. *The Annals of Mathematical Statistics*, 33(3):857–881, 1962.

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, 2018. URL https://arxiv.org/abs/1803.11485.

Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Garðar Ingvarsson Juto, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, et al. Jaxmarl: Multi-agent rl environments and algorithms in jax. *Advances in Neural Information Processing Systems*, 37:50925–50951, 2024.

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, 2019. URL https://arxiv.org/abs/1902.04043.

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, 2019. URL https://arxiv.org/abs/1905.05408.

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

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning, 2021. URL https://arxiv.org/abs/2008.01062.

Supplementary Materials

The following content was not necessarily subject to peer review.

Appendix

A Proofs

A.1 Proof of Theorem 1

Proof. Fix a joint policy $\boldsymbol{\pi}$ and perform exact centralized evaluation so that $U = U^{\boldsymbol{\pi}}$. By exact projection,

$$Q_i(\tau_i, u_i) = \mathbb{E}[U^{\boldsymbol{\pi}}(\boldsymbol{\tau}, s, \mathbf{u}) \mid \tau_i, u_i] \,.$$

Updating agent i greedily with respect to Q_i therefore maximizes, for every local history τ_i, the conditional expectation of the centralized value. Taking expectations over trajectories induced by (π_i', π_{-i}) yields

$$J(\pi_i', \pi_{-i}) \geq J(\pi_i, \pi_{-i}).$$

Thus, each update performs coordinate-wise ascent on the shared return. Because the joint policy space is finite and the return is bounded, the process must terminate at a policy where no single agent can improve the joint return unilaterally, i.e., a person-by-person optimal policy. □

A.2 Proof of Proposition 1

Proof. For fixed U and $\boldsymbol{\pi}$, the regression problem is stationary. The global minimizer of mean squared error over functions measurable with respect to (τ_i, u_i) is the conditional expectation

$$Q_i^*(\tau_i, u_i) = \mathbb{E}[y^U \mid \tau_i, u_i],$$

which is the L^2-projection of y^U onto the subspace of functions depending only on (τ_i, u_i). Conditioning integrates out the hidden state, other agents' histories, and their actions under $\boldsymbol{\pi}$. □

B Architecture

We implement Dec-ADQN as two networks that learn simultaneously, i.e., the joint network and the individual agent network as shown in Fig. 4. **Individual agent network:** Its architecture is shared by Dec-ADQN and the baselines to ensure fair comparison. The input to this network is the observation o_i and the output is $Q_i(\tau_i, u_i)$. It consists of an environment-specific *observation embedder* that processes o_i and generates an observation embedding. For SMAX, the observation embedder is a single fully connected (FC) layer, while for OvercookedV2, it consists of two convolution layers followed by an FC layer. These design choices are made based on the representational capacity required by the tasks. The observation embedding is then passed through an RNN layer, followed by an FC layer that outputs the Q-values. **Joint Network:** It computes $\hat{U}(\boldsymbol{\tau}, s, \mathbf{u})$ using the observations and actions of all agents along with the global state s. o_i from each agent is passed through an *observation embedder* (same as the one in the individual agent network), and the resulting observation embeddings from each agent are concatenated together. u_i of each agent are first concatenated and then passed through an *action embedder*, which is an FC layer. Lastly, s is passed through an environment-specific *state embedder* to get a state embedding. For SMAX, the state embedder is an FC layer, and for OvercookedV2, it is a convolutional neural network (the same as the observation embedder). The concatenated observation and action embeddings are then further concatenated together and passed through an RNN, whose output is concatenated with the state embedding. Lastly, the RNN output is passed through the *mixer* network that outputs $\hat{U}(\boldsymbol{\tau}, s, \mathbf{u})$. The mixer is a one-layer MLP for OvercookedV2 and a two-layer MLP for SMAX.

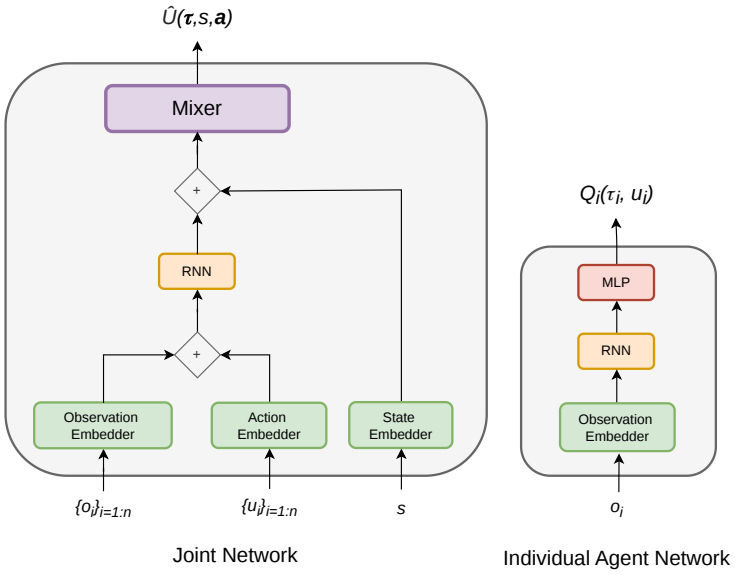
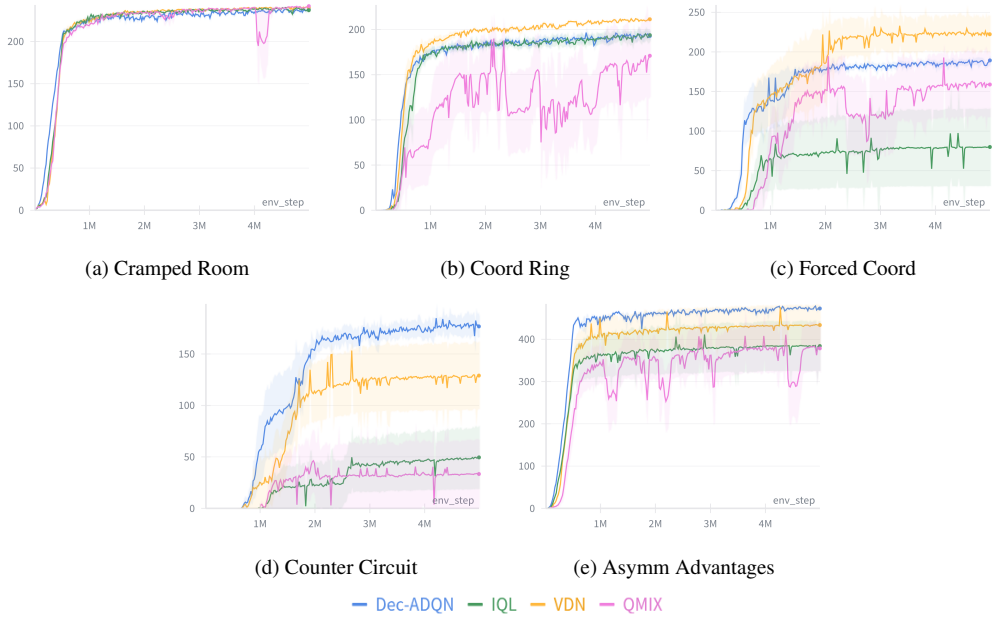


Figure 4: Dec-ADQN Architecture

C Overcooked Results

Fig. 5 shows the results for IQL, VDN, QMIX, and Dec-ADQN on Overcooked from the JaxMARL suite. The Overcooked environment is fully observable. The results demonstrate the superior performance of Dec-ADQN as compared to the baselines.



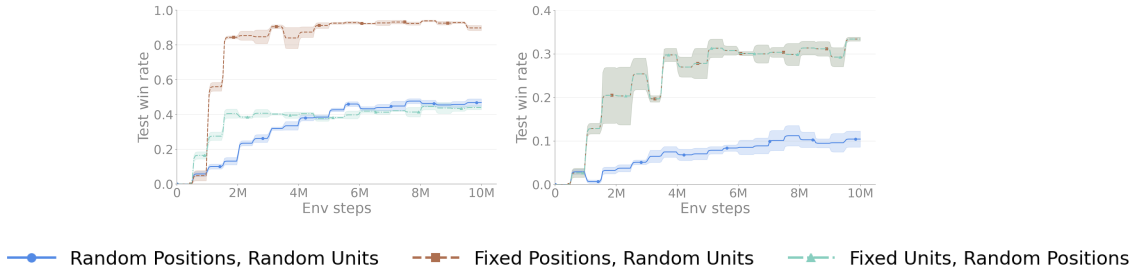
(a) Cramped Room (b) Coord Ring (c) Forced Coord

(d) Counter Circuit (e) Asymm Advantages

Figure 5: Results for episodic returns on Overcooked

D SMAX Ablation on SMACv2 Maps

Fig. 6 shows the results for Dec-ADQN on the two SMACv2 maps when either the unit type or the unit starting positions, or both are fixed. In the SMACv2_5_units map, fixing unit types had little effect, while fixing positions allowed Dec-ADQN to solve the task. In the SMACv2_10_units map, fixing the unit type or the unit starting positions had a very similar effect on the learning. This indicates that reducing the stochasticity in the environment significantly helps Dec-ADQN.



(a) smacv2_5_units (b) smacv2_10_units

Figure 6: SMACv2 ablation study

E Hyperparameters Used

For the SMAX maps, we adopt the default hyperparameters provided in the JaxMARL paper (Rutherford et al., 2024) to ensure consistency and comparability with prior work. Similarly, for OvercookedV2, we adopt the default Overcooked hyperparameters reported in the JaxMARL paper, as no standardized hyperparameter configuration is currently available for OvercookedV2. The details of the hyperparameters are shared in Table 1. These settings provide a consistent and reproducible baseline for cooperative MARL in the same environment family. For OvercookedV2, we use a slightly higher learning rate of 5×10^{-4} for Dec-ADQN, which empirically improves optimization stability for the centralized critic and the per-agent regressions. Increasing the learning rate for the baseline algorithms did not improve their performance, so we retain their standard hyperparameter settings for comparison.

F Fairness across algorithms

To ensure fair comparisons, we carefully controlled the network capacity and architectural complexity in the experiments. Dec-ADQN and all the baselines use identical individual agent networks. Keeping the individual agent consistent is important because IQL and VDN solely rely on these networks, while QMIX, QPLEX and Dec-ADQN have mixer networks in their architectures. For OvercookedV2, as shown in Table. 2, the size of the ADQN mixer is significantly smaller than that of the QMIX and the QPLEX Mixer. On the other hand, in Table 3, we see that the Dec-ADQN often mixer contains more parameters than the QMIX and QPLEX mixers. However, this increase stems from processing substantially larger inputs; Dec-ADQN takes the RNN embeddings and state embeddings as inputs compared to the QMIX mixer's input of Q values and the state. Despite the higher parameter count, the Dec-ADQN mixer (1-2 layer MLP) is structurally simpler than the QMIX (hypernetwork-based design) and QPLEX mixers (dueling mixing and attention-based mechanism). Thus, the difference in the number of parameters does not reflect unfair computational advantages but different approaches in processing the inputs.

Table 1: Hyperparameters

Hyperparameter	**SMAX**	**Overcooked**
Total Timesteps	1×10^7	5×10^6
Number of Environments	16	16
Number of Steps	128	32
Buffer Size	5,000	10,000
Buffer Batch Size	32	128
Hidden Size	512	512
ϵ-start	1.0	1.0
ϵ-finish	0.05	0.05
ϵ-decay	0.1	0.1
Learning Starts (timesteps)	10,000	1,000
Max Gradient Norm	10	1
Target Update Interval	10	10
τ	1.0	1.0
Number of Epochs	8	4
Learning Rate	5×10^{-5}	7×10^{-5}
LR Linear Decay	False	True
γ	0.99	0.99
Reward Scale	10.0	-
QMIX-specific parameters		
Mixer Embedding Dim	64	64
Mixer Hypernet Hidden Dim	256	256
Mixer Init Scale	0.001	0.001

Map	Dec-ADQN Mixer	QMIX Mixer	QPLEX Mixer	Individual Agent (All Alg)
cramped_room	3073	0.1M	0.4M	1.9M
asymm_advantages	3073	0.2M	0.4M	2.0M
coord_ring	3073	0.1M	0.4M	2.0M
forced_coord	3073	0.1M	0.4M	2.0M
counter_circuit	3073	0.1M	0.4M	2.0M

Table 2: Dec-ADQN Layer-wise parameter count for OvercookedV2

G Limitations

While Dec-ADQN is useful for many reasons, we acknowledge that it has certain limitations. First, Dec-ADQN provides convergence guarantees only in the idealized tabular setting and under exact optimization. In particular, Theorem 1 establishes convergence to a person-by-person optimal policy, not a globally optimal joint policy. Moreover, Dec-ADQN relies on projecting a centralized value function onto decentralized information spaces. When the conditional variance of the centralized value given local observations is large—such as in highly stochastic environments —the resulting regression targets can be noisy. In terms of computation, the centralized evaluator U operates over the joint action space during training. Although execution remains decentralized, the training-time complexity scales with the size of the joint action space. Finally, while Dec-ADQN avoids structural value factorization constraints, this flexibility may remove useful inductive biases. Therefore, in certain environments, factorization-based methods may perform equally well or better due to their stronger structural regularization.

Map	ADQN Mixer	QMIX Mixer	QPLEX Mixer	Individual Agent (All Alg)
3s_vs_5z	0.6M	0.1M	0.3M	1.6M
2s3z	1.3M	0.1M	0.4M	1.6M
3s5z_vs_3s6z	3.0M	0.1M	0.6M	1.6M
3s5z	3.0M	0.1M	0.6M	1.6M
6h_vs_8z	1.8M	0.1M	0.5M	1.6M
smacv2_5_units	1.3M	0.1M	0.4M	1.6M
smacv2_10_units	4.4M	0.1M	0.7M	1.6M
5m_vs_6m	0.1M	0.1M	0.4M	1.6M
10m_vs_11m	0.1M	0.1M	0.8M	1.6M

Table 3: ADQN Layer-wise parameter count for SMAX