On Stateful Value Factorization in Multi-Agent Reinforcement Learning

Enrico Marchesini
Massachusetts Institute of Technology
Cambridge (MA), USA
emarche@mit.edu

Andrea Baisero
Northeastern University
Boston (MA), USA
baisero.a@northeastern.edu

Rupali Bhati
Northeastern University
Boston (MA), USA
bhati.r@northeastern.edu

Christopher Amato
Northeastern University
Boston (MA), USA
c.amato@northeastern.edu

ABSTRACT

Value factorization is a popular paradigm for designing scalable multi-agent reinforcement learning algorithms. However, current factorization methods make choices without full justification that may limit their performance. For example, the theory in prior work uses stateless (i.e., history) functions, while the practical implementations use state information—making the motivating theory a mismatch for the implementation. Also, methods have built offof previous approaches, inheriting their architectures without exploring other, potentially better ones. To address these concerns, we formally analyze the theory of using the state instead of the history in current methods—reconnecting theory and practice. We then introduce DuelMIX, a factorization algorithm that learns distinct per-agent utility estimators to improve performance and achieve full expressiveness. Experiments on StarCraft II micromanagement and Box Pushing tasks demonstrate the benefits of our intuitions.

KEYWORDS

Value factorization; Multi-agent reinforcement learning

ACM Reference Format:
Enrico Marchesini, Andrea Baisero, Rupali Bhati, and Christopher Amato. 2025. On Stateful Value Factorization in Multi-Agent Reinforcement Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 16 pages.

1 INTRODUCTION

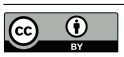
Recent advancements in multi-agent reinforcement learning (MARL) have led to impressive results in complex cooperative tasks [2, 19, 25]. Many of these methods use *centralised training with decentralised execution* (CTDE) [1, 3, 9, 42], which allows them to train in a centralized fashion but still execute in a decentralized manner.

The dominant form of CTDE in value-based MARL is value factorization [13, 14, 22–24, 28, 31, 33, 35, 36, 40]. These methods factor a (centralized) joint action value into (decentralized) per-agent utilities conditioned on local information. The resulting approaches ensure the greedy action selection from each agent's local utility is the same as greedy action selection over the centralized value function (i.e., the argmax over the local utilities is the same as a joint argmax over the centralized value function)—the *individual global max* (IGM) principle [30]. IGM provides decentralization and scalability since algorithms no longer need to perform costly joint maximization over all agents.

Earlier forms of factorization use strong constraints to ensure IGM (e.g., linearly or monotonically combining the local utilities in VDN [31] and QMIX [24]) which limited their expressiveness. More recent methods such as QPLEX [35] can, in theory, represent the full set of IGM factorizations. Despite the merits of value factorization, there is a mismatch between the theory and practice of these methods. In particular, the theory behind the methods assumes history information at the local and centralized levels while most practical implementations replace the history with the (ground truth) state in some places. While replacing history information with state information is tempting to exploit additional information during centralized training, it can be unsound in partially observable settings, as recently shown in the actor-critic CTDE case [10].

To address the gap between theory and practice in value factorization, we extend the theory to the *stateful* case that combines state and history information. We show that IGM (or Advantage-IGM—its formalization over advantage functions) still holds for most of the methods (VDN, QMIX, and QPLEX) but not necessarily for the weighted version of QMIX (WQMIX) [23]. We also show that QPLEX's practical implementation can not represent the full IGM function class due to the use of state information instead of history information—losing one of its main benefits.

In practice, while there are many architectures that would satisfy IGM, previous approaches made choices based on earlier work without exploring other alternatives that could improve performance. For example, unlike dueling networks, which typically learn separate history and advantage value functions at the agent level, QPLEX learns a Q-function and assumes the V-function is a max of it. For this reason, we introduce DuelMIX. DuelMIX maintains separate estimators at the agent level—instead of computing them from the agents' Q-functions. Such a separation has been shown to learn better value approximations, which enhance performance and sample efficiency in single-agent scenarios [6, 37, 41].



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).
* Work performed while at Carnegie Mellon University.

Our work makes the following contributions; we:

- Formalize IGM over the centralized stateful functions used in popular factorization algorithms.
- Analyze that the state does not introduce bias into QMIX [24] for IGM, and QPLEX [35] for Advantage-IGM.
- Empirically show that using other sources of information during factorization (i.e., constant and random vectors) could lead to performances comparable or better than using the state, contrasting the common belief that the state allows higher performance.
- Present DuelMIX, a factorization scheme integrating dueling networks at a per-agent level, and combining joint history-state values in a weighted fashion to achieve full expressiveness. This learning stream separation leads to significant benefits in cooperative scenarios where optimal joint policies often hinge on specific actions.

We evaluate these methods in the highly partially observable Box Pushing (BP) scenario, where the optimal behavior is contingent on a specific agent's action [38], and StarCraft II Lite (SMACLite) tasks [17]. Our results on BP show the benefits of separate value learning, allowing DuelMIX to achieve good performance where previous approaches fail. Moreover, SMACLite experiments show that DuelMIX outperforms previous factorization methods and significantly improves sample efficiency.

2 PRELIMINARIES

We model our tasks as *decentralized partially observable Markov decision processes* (Dec-POMDPs)[20]—a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{U}, T_{\mathcal{S}}, r, \mathcal{O}, T_{\mathcal{O}}, \gamma \rangle$. $\mathcal{N}, \mathcal{S}$ arefi nite set of agents and states; $\mathcal{U} \equiv \langle U_i \rangle_{i \in \mathcal{N}}, \mathcal{O} \equiv \langle O_i \rangle_{i \in \mathcal{N}}$ are thefi nite sets of joint actions and observations, while U_i, O_i are the individual ones. At each step, every agent i chooses an action, forming the joint action $\boldsymbol{u} \equiv \langle u_i \rangle_{i \in \mathcal{N}} \in \mathcal{U}$. After performing $\boldsymbol{u}$, the environment transitions from a state s to a new s', following $T_{\mathcal{S}} \colon \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to [0, 1]$ ($T_{\mathcal{S}}(s, \boldsymbol{u}, s') = \Pr(s' \mid s, \boldsymbol{u})$), and returning a joint reward $r \colon \mathcal{S} \times \mathcal{U} \to \mathbb{R}$. In a partially-observable setting, agents receive an observation $\boldsymbol{o} \equiv \langle o_i \rangle_{i \in \mathcal{N}} \in \mathcal{O}$ according to $T_{\mathcal{O}} \colon \mathcal{S} \times \mathcal{U} \times \mathcal{O} \to [0, 1]$ ($T_{\mathcal{O}}(\boldsymbol{u}, s', \boldsymbol{o}) = \Pr(\boldsymbol{o} \mid \boldsymbol{u}, s')$), and each agent maintains a policy $\pi_i(u_i \mid h_i)$ mapping local action-observation histories $h_i = (o_{i,0}, u_{i,0}, o_{i,1}, \ldots, o_{i,t}) \in H_i$ to actions. In finite-horizon tasks, we aim tofi nd a joint policy $\pi(\boldsymbol{u} \mid \boldsymbol{h})$ maximizing the expected discounted episodic return $\mathbb{E}_{\pi}\left[\sum_t \gamma^t r_t\right]$, where $\gamma \in [0, 1)$ is the discount and $\boldsymbol{h}_t = \langle \boldsymbol{o}_0, \boldsymbol{u}_0, \ldots, \boldsymbol{o}_t \rangle \in \mathcal{H}$ is the joint action-observation history.

2.1 Value Factorization

In this section, we summarize the stateless theoretical framework presented by seminal value factorization works. In Section 3, we will discuss how most algorithms use the state during factorization. As such, *we remark the following preliminaries do not correctly reflect most of the published literature.*

Factorization algorithms must satisfy the IGM principle (Equation (1)) [30], ensuring consistency between decentralized and centralized decision-making.

$$\arg\max_{\boldsymbol{u} \in \mathcal{U}} Q(\boldsymbol{h}, \boldsymbol{u}) \equiv \left(\arg\max_{u_i \in U_i} Q_i(h_i, u_i) \right)_{i \in \mathcal{N}}. \tag{1}$$

This consistency is key for scalability as it facilitates tractable joint action selection by deriving it from each agent's local utility. The individual history-action utilities $\langle Q_i : H_i \times U_i \to \mathbb{R} \rangle_{i \in \mathcal{N}}$ satisfy IGM for a joint history-action value function $Q : \mathcal{H} \times \mathcal{U} \to \mathbb{R}$ if the maximal actions over the centralized function and the local utilities align. Several methods have been proposed, each imposing different architectural constraints to guarantee IGM during the centralized training process. We provide concise descriptions of the main approaches considered in our work.

Additive Constraint. VDN is the foundational factorization method and expresses the joint history-action value as follows [31]:

$$Q(\boldsymbol{h}, \boldsymbol{u}) = \sum_{i=1}^{|\mathcal{N}|} Q_i(h_i, u_i). \tag{2}$$

We note that VDN is correctly formalized in terms of stateless functions and utilities, as it does not employ state information. However, VDN only represents a limited set of joint functions well.

Monotonic Constraint. QMIX uses a non-linear monotonic mixing network to combine agent utilities [24]:

$$\frac{\partial Q(\boldsymbol{h}, \boldsymbol{u})}{\partial Q_i(h_i, u_i)} \geq 0, \forall i \in \mathcal{N}. \tag{3}$$

By enforcing positive weights to satisfy the constraint in Equation (3), QMIX represents a broader class of functions compared to VDN. However, it is limited to functions that can be factored as non-linear monotonic combinations of the agents' utilities. Recently, Weighted-QMIX (WQMIX) extended QMIX with a weighting mechanism, placing more importance on better joint actions [23]. Despite framing the monotonic factored value in a stateless fashion, these methods use the state in the mixing network. This prompts our investigation of whether the state introduces possible learning biases, discussed later in Section 3.

Advantage-IGM. An advantage-based IGM principle equivalent to Equation (1) has been proposed by Wang et al. [35]. Given individual utilities $\langle Q_i \rangle_{i \in \mathcal{N}}$ and the joint Q defined as:

$$Q(\boldsymbol{h}, \boldsymbol{u}) = V(\boldsymbol{h}) + A(\boldsymbol{h}, \boldsymbol{u}), \;\; Q_i(h_i, u_i) = V_i(h_i) + A_i(h_i, u_i), \tag{4}$$

with V, A not strictly representing the mathematical values we typically associate with them, but their definition regardless of π:

$$\begin{aligned} V(\boldsymbol{h}) &= \max_{\boldsymbol{u'}} Q(\boldsymbol{h}, \boldsymbol{u'}), \\ A(\boldsymbol{h}, \boldsymbol{u}) &= Q(\boldsymbol{h}, \boldsymbol{u}) - \max_{\boldsymbol{u'}} Q(\boldsymbol{h}, \boldsymbol{u'}), \\ V_i(h_i) &= \max_{u_i'} Q_i(h_i, u_i'), \\ A_i(h_i, u_i) &= Q_i(h_i, u_i) - \max_{u_i'} Q_i(h_i, u_i'). \end{aligned} \tag{5}$$

Advantage-IGM is satisfied if the equivalence between centralized and decentralized action selections holds over A:

$$\arg\max_{\boldsymbol{u} \in \mathcal{U}} A(\boldsymbol{h}, \boldsymbol{u}) \equiv \left(\arg\max_{u_i \in U_i} A_i(h_i, u_i) \right)_{i \in \mathcal{N}}. \tag{6}$$

In its original formalization using Equation (5), Equation (6) has been shown equivalent to Equation (1) when the advantage values are non-positive—optimal actions' advantage must be zero, and

non-optimal actions must have negative advantages. QPLEX [35] builds on Advantage-IGM and decomposes the per-agent utilities $Q_i(h_i, u_i)$ into individual $V_i(h_i), A_i(h_i, u_i)$ according to Equation (5). In the practical implementation, such values are then conditioned on joint information using a non-linear transformation. This module outputs biases and positive weights $\langle b_i(\boldsymbol{h}), w_i(\boldsymbol{h}) > 0 \rangle_{i \in \mathcal{N}}$:

$$V_i(\boldsymbol{h}) = w_i(\boldsymbol{h}) V_i(h_i) + b_i(\boldsymbol{h}), \quad A_i(\boldsymbol{h}, u_i) = w_i(\boldsymbol{h}) A_i(h_i, u_i). \quad (7)$$

After this transformation, QPLEX factorizes the joint history-action value function as:

$$Q(\boldsymbol{h}, \boldsymbol{u}) = \sum_i V_i(\boldsymbol{h}) + \lambda_i(\boldsymbol{h}, \boldsymbol{u}) A_i(\boldsymbol{h}, u_i), \quad (8)$$

where $\langle \lambda_i(\boldsymbol{h}, \boldsymbol{u}) > 0 \rangle_{i \in \mathcal{N}}$ are weights computed with an attention module to enhance credit assignment.[1] While QPLEX achieves higher empirical performance than previous factorization methods, the authors claimed the transformations and $\lambda_i(\boldsymbol{h}, \boldsymbol{u})$ coefficients allow QPLEX to fully represent the functions satisfying Advantage-IGM [39]. This would be correct when using the joint history in the factorization modules, but the actual implementation uses the state. For this reason, in Section 3, we argue the QPLEX practical implementations are incorrect [7, 21, 26, 34] since they estimate weights and biases as functions of the state and not of the joint history (i.e., in practice we have $\langle b_i(s), w_i(s) > 0, \lambda_i(s, \boldsymbol{u}) \rangle_{i \in \mathcal{N}}$ rather than $\langle b_i(\boldsymbol{h}), w_i(\boldsymbol{h}) > 0, \lambda_i(\boldsymbol{h}, \boldsymbol{u}) \rangle_{i \in \mathcal{N}}$). Along these lines, QPLEX's theoretical framework also uses stateless functions, diverging from the actual stateful implementation. To address the theoretical mismatch, we formalize the stateful Advantage-IGM in Section 3, and formally analyze whether the state introduces biases also for QPLEX [35].

Additionally, we believe that QPLEX's performance can be further improved since: (i) the left side of Equation (8) is factored in a simple additive fashion, and (ii) history and advantage utilities are decomposed from every $Q_i(h_i, u_i)$ by their definition. These values are not learned locally and separately by each agent as originally proposed by dueling networks [37].

A true dueling architecture is relevant when a small subset of actions are useful for the task, which we note commonly happens in highly cooperative scenarios. To address these issues, we introduce a dueling networks-based factorization architecture in Section 4.

3 STATEFUL VALUE FACTORIZATION

Practical implementations of QMIX, WQMIX, and QPLEX often use state values in their centralized models—usually through a mixing network—in ways that are neglected in their theoretical analysis. For thefi rst time (to our knowledge), we analyze if the theory of these methods extends correctly to the stateful case, or whether using state values introduces any learning bias. Such biases can easily happen by improper uses of state in single and multi-agent partially observable control problems as recently shown by Baisero and Amato [4], Lyu et al. [10]. We begin by adjusting the notation of centralized value models that use state, effectively resulting in history-state values $Q(\boldsymbol{h}, s, \boldsymbol{u})$. Given that consistent history and history-state values are related by:

$$Q(\boldsymbol{h}, \boldsymbol{u}) = \mathbb{E}_{s|\boldsymbol{h}} \left[Q(\boldsymbol{h}, s, \boldsymbol{u}) \right],$$

we correctly reformulate the IGM principle in a marginalized history-state form as follows:

Proposition 3.1 (History-State IGM). *For a joint $Q : \mathcal{H} \times \mathcal{S} \times \mathcal{U} \to \mathbb{R}$ and individuals $\langle Q_i : H_i \times U_i \to \mathbb{R} \rangle_{i \in \mathcal{N}}$ s.t. the following holds:*

$$\arg\max_{\boldsymbol{u} \in \mathcal{U}} \mathbb{E}_{s|\boldsymbol{h}} \left[Q(\boldsymbol{h}, s, \boldsymbol{u}) \right] \equiv \left(\arg\max_{u_i \in U_i} Q_i(h_i, u_i) \right)_{i \in \mathcal{N}}, \quad (9)$$

$\langle Q_i(h_i, u_i) \rangle_{i \in \mathcal{N}}$ are said to satisfy History-State IGM for $Q(\boldsymbol{h}, s, \boldsymbol{u})$.

3.1 Role of the State in QMIX

The question now becomes whether the mixing model of QMIX (*State-QMIX*) satisfies the History-State IGM principle. We categorize the QMIX variants in terms of which auxiliary input is provided to the mixing hyper-network in addition to the individual agent utilities. In that regard, QMIX [24] is originally formalized directly in a state variant, while the IGM principle is formalized without any sort of auxiliary information. This discrepancy is not directly addressed, and may potentially undermine the validity of the theoretical guarantees for the practical implementation. In the following, we formalize these two possible variants of QMIX.

3.1.1 Plain-QMIX. The Plain-QMIX variant is a direct implementation of the IGM principle, with no auxiliary information provided to set the hyper-network weights. This variant uses a mixing network to combine the per-agent utilities $\langle Q_i(h_i, u_i) \rangle_{i \in \mathcal{N}}$ into a joint history-action value function:

$$Q(\boldsymbol{h}, \boldsymbol{u}) = f(Q_1(h_1, u_i), \ldots, Q_n(h_n, u_n)),$$

that satisfies monotonic constraints between the joint Q and the individual Q_i. This variant trivially satisfies the IGM principle.

3.1.2 State-QMIX. The State-QMIX variant uses ground truth state information to set the hyper-network weights, and it is the main variant proposed in its respective paper (see Figure 6 in Appendix A [12]).[2] This variant uses a mixing *hyper*-network to combine the per-agent utilities $\langle Q_i(h_i, u_i) \rangle_{i \in \mathcal{N}}$ and the ground truth state into a joint history-action value function:

$$Q(\boldsymbol{h}, s, \boldsymbol{u}) = f(Q_1(h_1, u_i), \ldots, Q_n(h_n, u_n), s),$$

that satisfies monotonic constraints between the joint Q and the individual Q_i. The original work by Rashid et al. [24] performs an empirical evaluation concerning the role of the state. However, it does not formally demonstrate that a stateful mixing model still satisfies the stateless IGM principle.

Given QMIX's hyper-network architecture and the way the state is used, the monotonicity constraint of Equation (3) holds for every state even for history-state values, resulting in a *non-marginalized* version of History-State IGM:

$$\arg\max_{\boldsymbol{u} \in \mathcal{U}} Q(\boldsymbol{h}, s, \boldsymbol{u}) \equiv \left(\arg\max_{u_i \in U_i} Q_i(h_i, u_i) \right)_{i \in \mathcal{N}}. \quad (10)$$

Notably, Equation (10) is distinct from Equation (9) in ways that may have problematic or even catastrophic consequences in partially observable control, e.g., the maximal action of a centralized history-state value corresponds to the optimal joint action under full observability, and not necessarily the optimal joint action for a team

[1] Both the transformation and the attention module use positive weights to maintain action selection consistency over the advantage values. Naturally, this positivity enforces monotonicity in the advantage factorization.

[2] The supplementary material is available in Marchesini et al. [12]

of partially observable agents. Fortuitously, despite the misleading formalization of the stateless IGM in the QMIX work [24], we confirm that the architectural constraints of QMIX's architecture restrict the centralized model in such ways that fundamentally avoid this issue altogether; a formal proof is provided in Appendix B [12].

Proposition 3.2 (QMIX State Bias)**.** *A stateful implementation of QMIX does not introduce any additional state-induced bias compared to a stateless implementation.*

Role of the State in WQMIX. WQMIX [23] is an extension of QMIX that generalizes the representation space of QMIX models by applying a loss weighting scheme. The authors provide a theoretical analysis that proves correctness, unbiasedness, and the ability of their centralized models to act as universal function approximators. Their analysis is already framed in the context of stateful values that are relevant to our work. However, it is also limited by the further assumption that the agents have full observability of the state to begin with; a significant discrepancy compared to their proposed methods in practice. To the best of our knowledge, there is yet no extension of their analysis that holds without the full observability assumption. Further, the methods that we have used to prove the unbiasedness of both QMIX and, following, QPLEX do not similarly hold for WQMIX.

3.2 Role of the State in QPLEX

The analysis for QPLEX follows a similar structure to that for QMIX. We begin by adjusting the model notation by replacing the joint history in the weights and biases estimated by the transformation and mixing modules with the state as used in the actual implementations [7, 21, 34].

We categorize the QPLEX variants in terms of the inputs to the weight, bias, and attention models. In that regard, the derivation and implementation of QPLEX [35] differ in ways that undermine the validity of the theoretical guarantees for the practical implementation. In the following subsections, we formalize each variant clearly, highlighting the differences. The history variant corresponds to the theoretical formalization of QPLEX, the state variant corresponds to its practical implementation, and the history-state variant is our attempt at bridging the gap between theory and practice.

3.2.1 History-QPLEX. History-QPLEX corresponds to the theoretical stateless formulation derived in the corresponding paper. It decomposes the per-agent utilities $\langle Q_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ into individual values $\langle V_i(h_i)\rangle_{i\in\mathcal{N}}$ and advantages $\langle A_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ according to:

$$\begin{aligned} V_i(h_i) &= \max_{u_i'} Q_i(h_i, u_i'), \\ A_i(h_i, u_i) &= Q_i(h_i, u_i) - \max_{u_i'} Q_i(h_i, u_i'). \end{aligned} \tag{11}$$

Such values are then conditioned on the joint history using a monotonic transformation taking as input the joint history $\boldsymbol{h}$ and outputting biases and positive weights $\langle b_i(\boldsymbol{h}), w_i(\boldsymbol{h}) > 0\rangle_{i\in\mathcal{N}}$:

$$V_i(\boldsymbol{h}) = w_i(\boldsymbol{h})V_i(h_i) + b_i(\boldsymbol{h}), \quad A_i(\boldsymbol{h}, u_i) = w_i(\boldsymbol{h})A_i(h_i, u_i). \tag{12}$$

After this transformation, History-QPLEX factorizes the joint value function as:

$$Q(\boldsymbol{h}, \boldsymbol{u}) = \sum_i V_i(\boldsymbol{h}) + \lambda_i(\boldsymbol{h}, \boldsymbol{u})A_i(\boldsymbol{h}, u_i), \tag{13}$$

where $\langle \lambda_i(\boldsymbol{h}, \boldsymbol{u}) > 0\rangle_{i\in\mathcal{N}}$ are weights computed with an attention module to enhance credit assignment. Both the transformation and the attention module use positive weights to maintain action selection consistency over the advantage values. Naturally, this positivity enforces monotonicity in the advantage factorization.

3.2.2 State-QPLEX. State-QPLEX corresponds to the stateful practical implementation used in the empirical evaluation in the corresponding paper (see Figure 7 in Appendix A [12]). The main difference compared to History-QPLEX is that the transformation and attention modules replace the joint history for the underlying ground truth state.

State-QPLEX decomposes the per-agent utilities $\langle Q_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ into individual values $\langle V_i(h_i)\rangle_{i\in\mathcal{N}}$ and advantages $\langle A_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ as Equation (11). Such values are then conditioned on the ground truth state using a monotonic transformation. This module takes as input the state s and outputs biases and positive weights $\langle b_i(s), w_i(s) > 0\rangle_{i\in\mathcal{N}}$:

$$\begin{aligned} V_i(h_i, s) &= w_i(s)V_i(h_i) + b_i(s), \\ A_i(h_i, s, u_i) &= w_i(s)A_i(h_i, u_i). \end{aligned} \tag{14}$$

After this transformation, State-QPLEX factorizes the joint value function as:

$$Q(\boldsymbol{h}, s, \boldsymbol{u}) = \sum_i V_i(h_i, s) + \lambda_i(s, \boldsymbol{u})A_i(h_i, s, u_i), \tag{15}$$

where $\langle \lambda_i(s, \boldsymbol{u}) > 0\rangle_{i\in\mathcal{N}}$ are weights computed with an attention module to enhance credit assignment. Note that the joint value function is now also a function of state, due to the dependence on the stateful transformation and attention modules. We argue that these state-based implementations do not result in full IGM expressiveness. The advantage stream of QPLEX is composed by a monotonic combination of individual state-history advantage utilities (with the same expressiveness of monotonic factorization) and an additive combination of local state-history utilities. The two streams (i.e., V, A) would require feeding joint history information in a non-linear mixer to achieve full expressiveness.

3.2.3 History-State-QPLEX. History-State-QPLEX is our proposed attempt at unifying the stateless theoretical derivation with the stateful practical implementation. The main difference compared to previous variants is that the transformation and attention modules employ *both* the joint history and the underlying ground truth state.

This method decomposes the per-agent utilities $\langle Q_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ into individual values $\langle V_i(h_i)\rangle_{i\in\mathcal{N}}$ and advantages $\langle A_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ as Equation (11). Such values are then conditioned on the joint history and the ground truth state using a monotonic transformation. This module takes as input the joint history $\boldsymbol{h}$ and state s and outputs biases and positive weights $\langle b_i(\boldsymbol{h}, s), w_i(\boldsymbol{h}, s) > 0\rangle_{i\in\mathcal{N}}$:

$$V_i(\boldsymbol{h}, s) = w_i(\boldsymbol{h}, s)V_i(h_i) + b_i(\boldsymbol{h}, s), \tag{16}$$

$$A_i(\boldsymbol{h}, s, u_i) = w_i(\boldsymbol{h}, s)A_i(h_i, u_i). \tag{17}$$

After this transformation, History-State-QPLEX factorizes the joint value function as:

$$Q(\boldsymbol{h}, s, \boldsymbol{u}) = \sum_i V_i(\boldsymbol{h}, s) + \lambda_i(\boldsymbol{h}, s, \boldsymbol{u})A_i(\boldsymbol{h}, s, u_i), \tag{18}$$

where $\langle \lambda_i(\boldsymbol{h}, s, \boldsymbol{u}) > 0\rangle_{i\in\mathcal{N}}$ are weights computed with an attention module to enhance credit assignment. As in the previous case, the

joint value function is now also a function of state, due to the dependence on the stateful transformation and attention modules.

Regarding the potential learning bias introduced by using the state, the architectural constraints of QPLEX have to guarantee the History-State Advantage-IGM principle, which is again distinct from the respective stateless formulation of Equation (6) in ways that may negatively impact partially observable control.

Nonetheless, as in the case of QMIX, we can show that the QPLEX models prohibit the state from informing the action-selection process; a formal proof is provided in Appendix B [12].

Proposition 3.3 (QPLEX State Bias)**.** *A stateful implementation of QPLEX does not introduce any additional state-induced bias compared to a stateless implementation.*

4 DUELMIX

To overcome the implementation drawbacks of previous factorization algorithms, we present a novel factorization scheme, DuelMIX.

Our approach leverages the dueling networks estimator at a per-agent level and introduces a weighted mixing mechanism that estimates a joint history value. Following our intuitions on stateful factorization, we introduce DuelMIX using stateful functions.

4.1 Algorithm

The overall architecture of DuelMIX is detailed in the following sections. In particular, Figure 1 shows the overall architecture of DuelMIX, which is composed of the following modules:

Agent Dueling Utility (yellow). Each agent $i \in \mathcal{N}$ employs a recurrent Q-network taking its previous hidden state h_i^{t-1}, previous action u_i^{t-1}, and current observation o_i^t as input to ensure decentralized execution. In contrast to previous factorization methods that produce a local utility $Q_i(h_i, u_i)$ through a single-stream estimator, DuelMIX utilizes two separate streams and outputs history and advantage utilities, denoted as $V_i(h_i)$ and $A_i(h_i, u_i)$, respectively. Crucially, building upon the insights of Wang et al. [37], each centralized update influences the V_i stream, enhancing the approximation of the history value. In single-agent scenarios, enhancing the approximation of value functions has led to higher performance and sample efficiency [11, 15, 16, 37], and also proves to be particularly advantageous in cooperative tasks. In these tasks, the optimal joint policy often hinges on specific actions taken in particular histories (as demonstrated in the Box Pushing experiments of Section 5). Simultaneously, advantage utilities are employed for decentralized action selection, such as with an ϵ-greedy policy. In particular, the advantage stream outputs:

$$A_i(h_i, \cdot) - \max_{u_i} A_i(h_i, u_i),$$

forcing advantages to be zero for the chosen action and non-positive (≤ 0) for the others (which becomes necessary when transforming the advantage values with positive weights).

Transformation (green). DuelMIX incorporates the transformation network used by Qatten and QPLEX. In detail, the transformation network combines local utilities $\langle V_i(h_i), A_i(h_i, u_i)\rangle_{i\in\mathcal{N}}$ with state information $\langle V_i(h_i, s), A_i(h_i, s, \boldsymbol{u})\rangle_{i\in\mathcal{N}}$. This module consists of a multi-layer perceptron (MLP) taking the state s as input and outputs a set of biases and positive weights $\langle b_i(s), w_i(s) > 0\rangle_{i\in\mathcal{N}}$. These weights and biases transform the local utilities as in Equation (11). Unlike QPLEX, the DuelMIX transformation module transforms the $V_i(h_i), A_i(h_i, u_i)$ learned by the agent, instead of being obtained by decomposition from $Q_i(h_i, u_i)$ following their optimal definition as in Equation (4).

Mixing (blue). The centralized mixing network employed by DuelMIX uses a similar multi-head attention mechanism as in QPLEX to estimate positive importance weights $\langle \lambda_i(\boldsymbol{h}, s, \boldsymbol{u}) > 0\rangle_{i\in\mathcal{N}}$. In contrast to QPLEX, we feed joint history information in the attention module, allowing DuelMIX to have full expressiveness over the class of functions satisfying IGM. In particular, these λ_i weights are positive to maintain action-selection consistency, and are combined with the per-agent transformed advantages, yielding the stateful joint advantage value:

$$A(\boldsymbol{h}, s, \boldsymbol{u}) = \sum_{i}^{|\mathcal{N}|} \lambda_i(\boldsymbol{h}, s, \boldsymbol{u}) A_i(h_i, s, u_i). \tag{19}$$

In contrast to prior works, we use an MLP taking as input the transformed history utilities and the state to estimate weights $\langle w_i'(\boldsymbol{h}, s)\rangle_{i\in\mathcal{N}}$ that are used to factorize the joint history value as:

$$V(\boldsymbol{h}, s) = \sum_{i}^{|\mathcal{N}|} w_i'(\boldsymbol{h}, s) V_i(h_i, s). \tag{20}$$

Unlike the positive weights in the joint advantage factorization, our design of $w_i'(\boldsymbol{h}, s)$ can assume arbitrary values since $V(\boldsymbol{h}, s)$ does not influence the action-selection process.[3]

The joint history-state-action value driving the centralized learning process then follows by definition:

$$Q(\boldsymbol{h}, s, \boldsymbol{u}) = V(\boldsymbol{h}, s) + A(\boldsymbol{h}, s, \boldsymbol{u}). \tag{21}$$

Given the nature of factorization methods based on Deep Q-Networks [18], DuelMIX is trained end-to-end to minimize the mean squared error loss:

$$\begin{aligned} L(\Theta) &= \frac{1}{|b|} \sum_{i=1}^{|b|} \left[\left(y - Q_\Theta(\boldsymbol{h}, s, \boldsymbol{u}) \right)^2 \right], \\ y &= r + \gamma Q_{\Theta'}(\boldsymbol{h}', s', \boldsymbol{u}'), \end{aligned} \tag{22}$$

where $\boldsymbol{u}' = \langle u_i' = \arg\max_{u_i'} Q_i(h_i', u_i')\rangle_{i\in\mathcal{N}}$, Θ represents the weights of the entire DuelMIX network (Θ' are the parameters of a target network [32]), and b is a batch of transitions sampled from a replay buffer. Given the nature of our architecture, DuelMIX shares the same limitations as related value factorization works (e.g., having a simulator for centralized training, where it is possible to get the state of the environment and communicate between agents).

4.2 Representational Complexity

The positive weights in DuelMIX, the non-positive advantage utilities, and the joint history used to estimate $\lambda_i(\boldsymbol{h}, s, \boldsymbol{u}), w_i'(\boldsymbol{h}, s)$, enable our method to satisfy IGM and achieve full expressiveness. A formal proof is provided in Appendix C [12].

[3] We explored different variations in designing the centralized history value stream of DuelMIX, with Equation (20) yielding the best overall performance.

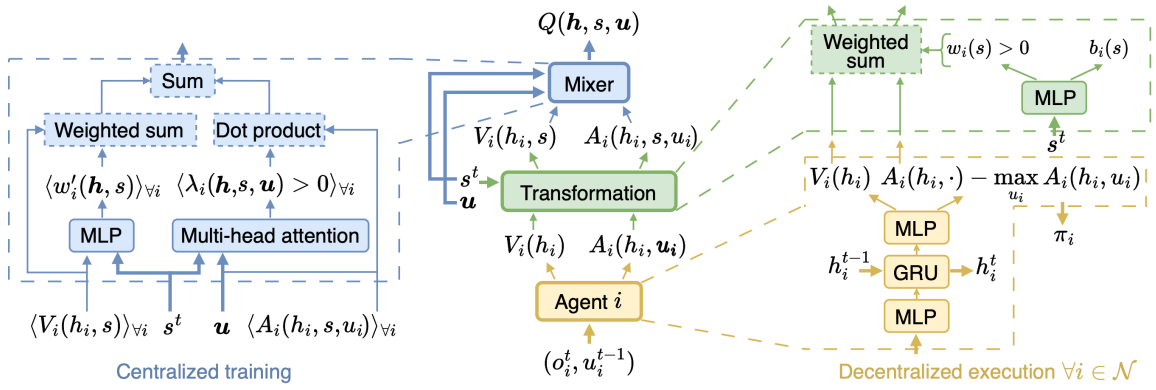


Figure 1: DuelMIX architecture: (i) agent dueling utility network structure (yellow); (ii) transformation module (green); (iii) mixing network architecture.

Proposition 4.1. *The function class that DuelMIX can realize, projected on* $\mathbb{E}_{s|\boldsymbol{h}}\,[Q(\boldsymbol{h}, s, \boldsymbol{u})]$*, is equivalent to what is induced by History-State IGM (Proposition 3.1).*

Role of the State in DuelMIX. We conclude our analysis of DuelMIX by including a result analogous to those in Section 3, on the bias that may be introduced by potentially improper use of stateful models. As in the case of both QMIX and QPLEX, we are able to determine the following proposition, whose formal proof is provided in Appendix B [12].

Proposition 4.2 (DuelMIX State Bias)**.** *A stateful implementation of DuelMIX does not introduce any additional state-induced bias compared to a stateless implementation.*

5 EXPERIMENTS

This section presents a comprehensive evaluation of the performance of DuelMIX in comparison to existing factorization methods, namely VDN [31], QMIX [24], Qatten [39], and QPLEX [35].[4]

Our experiments address the following key questions: (i) *Does the agent's dueling utility network learn a more effective representation of the state value?* (ii) *Does DuelMIX exhibit better performance over previous algorithms?* (iii) *Is the state crucial for performance in value factorization?* To answer these questions, we conduct experiments using the well-known Box Pushing task [27] and standard micromanagement tasks based on Starcraft II [17].

5.1 Implementation Details

Data collection is performed on Xeon E5-2650 CPU nodes with 64GB of RAM, using existing implementations for the baselines [24, 31, 35, 39]. Hyperparameters are in Appendix D [12] and we report the average return smoothed over the last ten episodes of ten runs per method. Shaded regions represent the standard error. This number of independent trials surpasses the typical 3-5 runs used in previous works [24, 31, 35, 39]. Considering the computational resources used for our experiments, Appendix E [12] addresses our strategy to offset estimated CO2 emissions.

[4]We use QMIX over WQMIX as it achieved comparable performance whenfine-tuned while being less computationally demanding [8].

5.2 Environments

To demonstrate the benefits of learning per-agent separate utilities, we consider the BP task [27] with a grid size of 30 (BP-30). In this Dec-POMDP task, two agents must collaborate to move a large box to the goal. Notably, agents can individually push small boxes, while moving the large box requires synchronized effort. Agents have very limited visibility observing only the cell in front of them, making high-dimensional scenarios considerably challenging.

We also test in the SMACLite decentralized micromanagement tasks [17]. SMAC is the standard benchmark for evaluating factorization-based MARL algorithms and SMACLite significantly reduces computational requirements while maintaining comparable performance to the original version [25].[5] We consider seven different setups, including two unsolved super-hard tasks proposed by Wang et al. [35] to show the superior performance of DuelMIX. We refer to Appendix F [12] for a more detailed discussion of these scenarios.

Figure 2 shows a representative game view of Box Pushing on the left, and SMACLite on the right.

5.3 Results

5.3.1 Box Pushing. This challenging Dec-POMDP task allows us to visualize how much importance the dueling utility network gives to the input features. Figure 3 shows results for stateful factorization algorithms. Overall, previous methods learn very sub-optimal

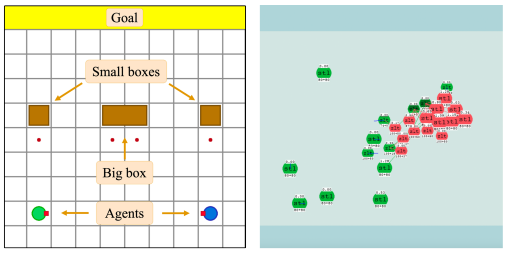


Figure 2: Representative overview of the SMACLite 7sz task (left) and Box Pushing (right).

[5]SMACLite trained policies have comparable results in the SMAC scenarios [17].

policies, where both agents roam around in the grid for a few steps, before one of them successfully pushes a small box to the goal. DuelMIX learns a more effective policy that enables the two agents to navigate directly to small boxes and push them simultaneously to the goal. This confirms DuelMIX's capability to achieve higher returns in challenging scenarios.

5.3.2 State Stream Representation. We show the saliency maps of Simonyan et al. [29], employing the Jacobian of the trained network, which allows us to visualize salient parts of the input as seen by the model's weights, showing which input features are more relevant for the weights. For clarity, this requires re-training our policies in a fully observable setup, where agents take as input their positions, their orientations as a one-hot encoding, and the position of the boxes.[6] It is thus possible to plot the "importance" that the state value stream of DuelMIX and QPLEX gives at each input. The idea is to show that DuelMIX's state value stream gives significantly less importance to entities that are not relevant to achieving a good payoff, while QPLEX's state value is not able to do so. Figure 4 overlaps the saliency map onto a reduced-size BP view. Blue cells indicate the features (cells) are not relevant to the agent so they will not affect its decisions relevantly. In contrast, green cells indicate the agent gives high importance to those features. So we would like to see our policies giving more importance (i.e., green) to the only cells leading to a positive reward. We show the average normalized

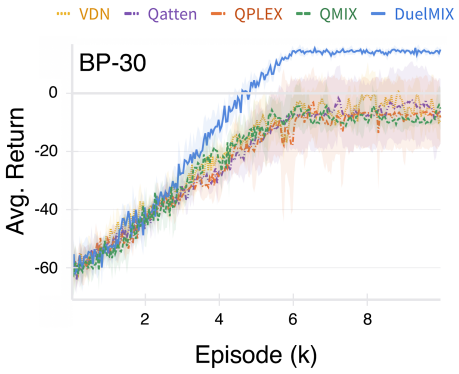


Figure 3: Learning curves for the stateful algorithms in BP.

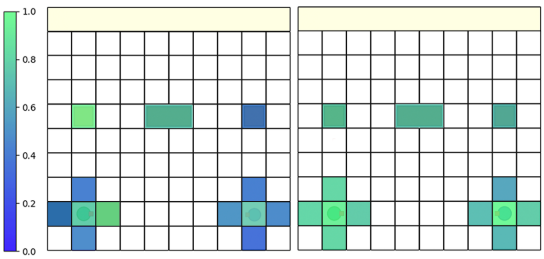


Figure 4: Saliency map of DuelMIX (left) and QPLEX (right) left agent's state value with respect to the initial state.

[6]Previous runs only observe the cell in front of the agents, which does not provide useful visual information.

importance that DuelMIX's value (left) and QPLEX's one (right) give to the input features at thefi rst step in the environment for the agent on the left. Crucially, we obtained similar results at different steps, showing DuelMIX's state value stream gives significantly higher importance to features that are relevant for achieving high returns (high-valued features in green are agents' positions and the nearby boxes while the others have inferior value and marked in blue). In contrast, QPLEX almost equally balances the importance of all the input features (i.e., all the values are similar), which could be detrimental to training.

5.3.3 SMACLite. Figure 5 illustrates the results of our evaluation in SMACLite environments. Notably, DuelMIX achieves the highest average return across most tasks, demonstrating superior performance. Moreover, DuelMIX exhibits improved sample efficiency, especially in easy and hard environments, learning behaviors with higher payoffs in fewer steps. In contrast, the limited expressiveness of VDN is known to struggle in complex domains. Qatten also achieved low performance. This is potentially related to the discrepancy between the centralized joint history-state-action value function and the Qatten formalization, which has been addressed by QPLEX. The latter achieved the highest overall performance among the factorization baselines, due to the benefits of building upon Advantage-IGM. Moreover, it is interesting to note QMIX's competitive performance over the more advanced QPLEX. These results further confirm the intuitions of Hu et al. [8], which showed QMIX significantly benefits from appropriatefine-tuning.

5.4 Influence of the State on Performance

We explore the performance of using different centralized information during factorization. Injecting the state in the mixer has become a standard de facto, but its use can not be supported by the theory. We investigate the performance of factorization algorithms (except VDN, which does not use the state) on representative SMACLite tasks. Following our stateful analysis in Section 3, the state information gets marginalized so it is not clear that any information is actually being used from it. In both scenarios, the impact of weights and biases outputted by the factorization modules introduce noise in the joint estimation with the result of increasing exploration and, potentially, robustness. As such, we expect different sources of centralized information to work well with factorization algorithms.

Table 1 presents the results, employing two types of centralized information in place of the state (s): (i) uniform random noise $\in [0, 1.0]$ (r), and (ii) a constant vector (c).

In particular, uniform random noise has maximum entropy, while the constant vector carries no information. These experiments use the same hyperparameters of the previous evaluation. Interestingly, when using random noise, different methods result in different behaviors. QMIX is affected detrimentally, and its performance is significantly lower than that of its stateful version. The performance of Qatten remains similar in most scenarios. QPLEX achieves comparable performance in *5s10z* while being affected slightly negatively in the other two tasks. While behaving similarly to QPLEX in the latter tasks, DuelMIX with random noise achieves higher performance in *5s10z*. Moreover, the performance gap between the constant and the random cases is negligible for DuelMIX, which

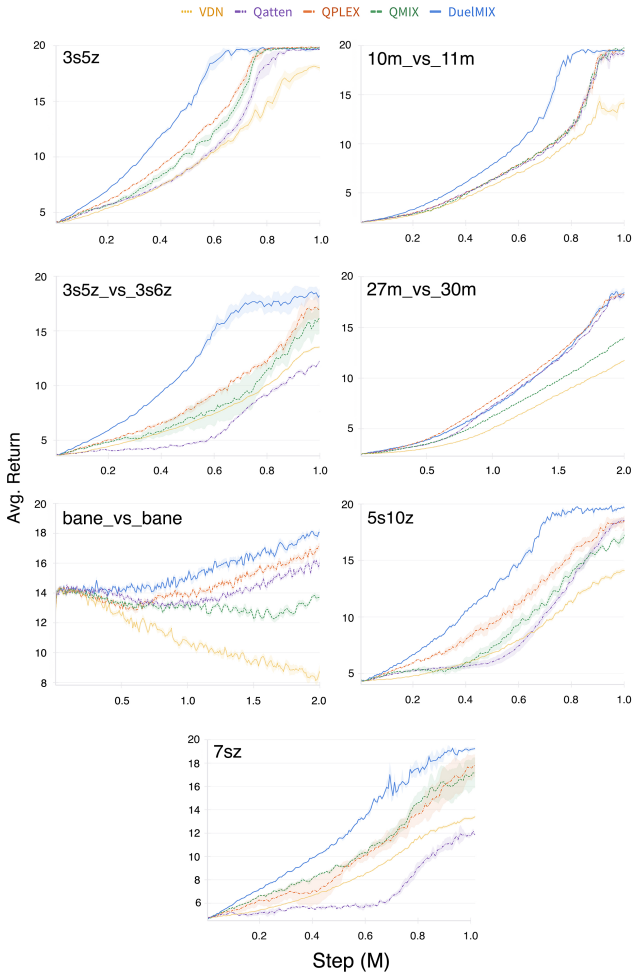


Figure 5: Average return during training for stateful factorization algorithms in SMACLite maps.

achieves superior performance in *5s10z*. The only major difference is that QPLEX outperforms its stateful version in the same task.

5.4.1 Fine-tuning. Additionalfi ne-tuning experiments in Table 2 reveal that using these different centralized information results in comparable or superior performance over the stateful choice, emphasizing the importance of our empirical investigation. We performed our initial grid search tofi ne-tune random noise and constant vector-based factorization methods. Table 2 shows the positive outcomes of these experiments in *5s10z*. Thefi ne-tuned QMIX achieves significantly higher performance than its non-fine-tuned counterpart. Moreover, thefi ne-tuned QPLEX with random noise and constant vector outperforms both its stateful version and its non-tuned runs. The average return during training for these additional experiments is in Appendix G [12].

6 CONCLUSIONS

This work addressed an important gap in the theoretical and practical underpinnings of value function factorization. We analyzed the relationship between theoretical frameworks and practical implementations and proposed a novel efficient factorization algorithm.

From a theoretical standpoint, we formally analyze the mismatch between the stateless theoretical framework presented in prior research and the actual stateful algorithms. Our experiments further questioned the conventional use of the state during the factorization process. Contrary to common practice, where the state is employed as centralized information, our results suggest there could be better solutions. In particular, even simplistic forms of centralized random noise or zero vectors, exhibit comparable or superior performance compared to stateful algorithms in certain scenarios.

On the practical front, we introduced DuelMIX, a factorization scheme designed to learn more general and distinct per-agent utility estimators. Furthermore, DuelMIX combines history values in a weighted manner to refine the estimation of the joint history value. Experiments on StarCraft II micromanagement and complex coordination tasks demonstrate the benefits of our intuitions.

Our empirical insights not only contribute significantly to the current understanding of MARL but also lay a principled foundation for future research in this domain.

Table 1: Average convergence return when using centralized state (s), random noise (r), or constant vector (c).

		3s5z_vs_3s6z	5s10z	7sz
QMIX	*s*	*16.0 ± 2.7*	*15.8 ± 0.4*	*16.7 ± 1.9*
	r	12.4 ± 0.9	9.2 ± 0.8	12.7 ± 0.3
	c	11.4 ± 0.9	8.0 ± 0.2	12.0 ± 0.6
Qatten	*s*	*12.2 ± 0.4*	*16.5 ± 1.7*	10.7 ± 0.8
	r	11.8 ± 1.1	12.0 ± 0.6	*11.3 ± 0.5*
	c	11.7 ± 1.1	11.9 ± 0.3	11.2 ± 0.5
QPLEX	*s*	*17.4 ± 2.7*	16.4 ± 2.4	*16.3 ± 3.2*
	r	13.9 ± 1.2	16.4 ± 0.6	14.1 ± 0.8
	c	14.4 ± 1.2	*17.1 ± 0.3*	12.5 ± 1.0
DuelMIX	*s*	***18.6 ± 0.8***	19.0 ± 0.7	***19.1 ± 0.4***
	r	17.2 ± 1.5	***19.3 ± 0.5***	17.5 ± 1.3
	c	17.0 ± 1.1	19.2 ± 0.7	17.9 ± 1.3

Table 2: Average return forfi ne-tuned QPLEX and QMIX with (r, c) vector information.

(fine-tuned ↓)		**5s10z**
QMIX	*s*	**15.8 ± 0.4**
	r	14.5 ± 1.4
	c	14.7 ± 0.1
QPLEX	*s*	16.2 ± 2.1
	r	18.0 ± 0.6
	c	**18.3 ± 0.8**

ACKNOWLEDGMENTS

This work was partially funded by the NSF award number 2044993.

REFERENCES

[1] Christopher Amato. 2024. An Introduction to Centralized Training for Decentralized Execution in Cooperative Multi-Agent Reinforcement Learning. arXiv:2409.03052 [cs.LG] https://arxiv.org/abs/2409.03052

[2] Ayhan Alp Aydeniz, Enrico Marchesini, Christopher Amato, and Kagan Tumer. 2024. Entropy Seeking Constrained Multiagent Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2141–2143.

[3] Ayhan Alp Aydeniz, Enrico Marchesini, Robert Loftin, and Kagan Tumer. 2023. Entropy Maximization in High Dimensional Multiagent State Spaces. In *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. 92–99.

[4] Andrea Baisero and Christopher Amato. 2022. Unbiased Asymmetric Reinforcement Learning under Partial Observability. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[5] Balázs Csanád Csáji. 2001. Approximation with Artificial Neural Networks. In *MSc Thesis, Eötvös Loránd University (ELTE), Budapest, Hungary*.

[6] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 32 (2018).

[7] Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. 2021. https://github.com/hijkzzz/pymarl2.

[8] Jian Hu, Siying Wang, Siyang Jiang, and Musk Wang. 2023. Rethinking the Implementation Tricks and Monotonicity Constraint in Cooperative Multi-agent Reinforcement Learning. In *The Second Blogpost Track at ICLR 2023*.

[9] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[10] Xueguang Lyu, Andrea Baisero, Yuchen Xiao, Brett Daley, and Christopher Amato. 2023. On Centralized Critics in Multi-Agent Reinforcement Learning. *Journal of Artificial Intelligence Research (JAIR)* 77 (2023), 295–354.

[11] Enrico Marchesini and Christopher Amato. 2023. Improving Deep Policy Gradients with Value Function Search. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=6qZC7pfenQm

[12] Enrico Marchesini, Andrea Baisero, Rupali Bhati, and Christopher Amato. 2024. On Stateful Value Factorization in Multi-Agent Reinforcement Learning. arXiv:2408.15381 [cs.AI] https://arxiv.org/abs/2408.15381

[13] Enrico Marchesini and Alessandro Farinelli. 2021. Centralizing State-Values in Dueling Networks for Multi-Robot Reinforcement Learning Mapless Navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[14] Enrico Marchesini and Alessandro Farinelli. 2022. Enhancing Deep Reinforcement Learning Approaches for Multi-Robot Navigation via Single-Robot Evolutionary Policy Search. In *2022 International Conference on Robotics and Automation (ICRA)*.

[15] Enrico Marchesini, Luca Marzari, Alessandro Farinelli, and Christopher Amato. 2023. Safe Deep Reinforcement Learning by Verifying Task-Level Properties. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 1466–1475.

[16] Luca Marzari, Changliu Liu, Priya L. Donti, and Enrico Marchesini. 2024. Improving Policy Optimization via ε-Retrain. In *arXiv*.

[17] Adam Michalski, Filippos Christianos, and Stefano V. Albrecht. 2023. SMAClite: A Lightweight Environment for Multi-Agent Reinforcement Learning. In *arXiv*.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[19] Igor Mordatch and Pieter Abbeel. 2017. Emergence of Grounded Compositional Language in Multi-Agent Populations. In *arXiv*.

[20] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[21] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. 2021. https://github.com/uoe-agents/epymarl.

[22] Thomy Phan, Fabian Ritz, Lenz Belzner, Philipp Altmann, Thomas Gabor, and Claudia Linnhoff-Popien. 2021. VAST: Value Function Factorization with Variable Agent Sub-Teams. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.). 24018–24032.

[23] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schroder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[25] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. 2019. The StarCraft Multi-Agent Challenge. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[26] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. 2019. https://github.com/oxwhirl/pymarl.

[27] Sven Seuken and Shlomo Zilberstein. 2007. Improved memory-bounded dynamic programming for decentralized pomdps. In *Conference on Uncertainty in Artificial Intelligence (UAI)*.

[28] Siqi Shen, Chennan Ma, Chao Li, Weiquan Liu, Yongquan Fu, Songzhu Mei, Xinwang Liu, and Cheng Wang. 2023. RiskQ: Risk-sensitive Multi-Agent Reinforcement Learning Value Factorization. In *Advances in Neural Information Processing Systems*. 34791–34825.

[29] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *CVPR*.

[30] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[31] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2017. Value-Decomposition Networks For Cooperative Multi-Agent Learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[32] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-learning. In *Conference on Artificial Intelligence (AAAI)*.

[33] Jianhao Wang, Zhizhou Ren, Beining Han, Jianing Ye, and Chongjie Zhang. 2021. Towards Understanding Cooperative Multi-Agent Q-Learning with Value Factorization. In *Advances in Neural Information Processing Systems*.

[34] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. https://github.com/wjh720/QPLEX.

[35] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *International Conference on Learning Representations (ICLR)*.

[36] Xiangsen Wang and Xianyuan Zhan. 2023. Offline Multi-Agent Reinforcement Learning with Coupled Value Factorization. In *AAMAS*.

[37] Ziyu Wang, Matteo Hessel Tom Schaul, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. 2016. Dueling Network Architectures for Deep Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

[38] Yuchen Xiao, Joshua Hoffman, and Christopher Amato. 2020. Macro-Action-Based Deep Multi-Agent Reinforcement Learning. In *CoRL*.

[39] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A General Framework for Cooperative Multiagent Reinforcement Learning. In *arXiv*.

[40] Hanhan Zhou, Tian Lan, and Vaneet Aggarwal. 2022. PAC: Assisted Value Factorization with Counterfactual Predictions in Multi-Agent Reinforcement Learning. In *Advances in Neural Information Processing Systems*. 15757–15769.

[41] Tianchen Zhou, Yutaka Yakuwa, Natsuki Okamura, Hiroyuki Hochigai, Takayuki Kuroda, and Ikuko Eguchi Yairi. 2025. Dueling Network Architecture for GNN in the Deep Reinforcement Learning for the Automated ICT System Design. *IEEE Access* 13 (2025), 21870–21879.

[42] Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Yanhao Huang, Jie Song, and Mingli Song. 2023. Is Centralized Training with Decentralized Execution Framework Centralized Enough for MARL? arXiv:2305.17352 [cs.AI] https://arxiv.org/abs/2305.17352