VAST: Value Function Factorization with Variable Agent Sub-Teams

Anonymous Author(s)
Affiliation
Address
email

Abstract

Value function factorization (VFF) is a popular approach to cooperative multi-agent reinforcement learning in order to learn local value functions from global rewards. However, state-of-the-art VFF is limited to a handful of agents in most domains. We hypothesize that this is due to the flat factorization scheme, where the VFF operator becomes a performance bottleneck with an increasing number of agents. Therefore, we propose VFF with *variable agent sub-teams (VAST)*. VAST approximates a factorization for sub-teams which can be defined in an arbitrary way and vary over time, e.g., to adapt to different situations. The sub-team values are then linearly decomposed for all sub-team members. Thus, VAST can learn on a more focused and compact input representation of the original VFF operator. We evaluate VAST in three multi-agent domains and show that VAST can significantly outperform state-of-the-art VFF, when the number of agents is sufficiently large.

1 Introduction

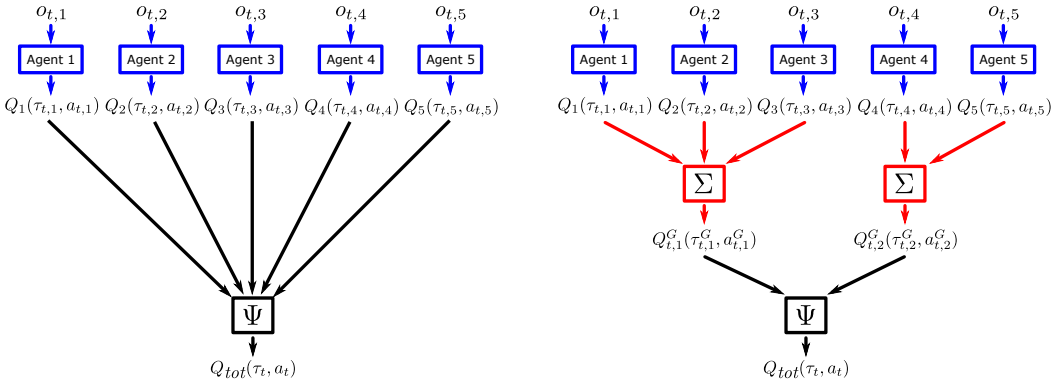
Many real-world problems can be defined as cooperative *multi-agent system (MAS)*, where multiple autonomous agents collaborate to achieve a common goal like fleet management [19, 20], industry 4.0 [9, 39], or communication networks [24, 45]. *Multi-agent reinforcement learning (MARL)* seems promising to realize such cooperative MAS by learning local policies for each autonomous agent [2, 25, 32, 35]. *Multi-agent credit assignment* is an important challenge, where all agents only observe a single global reward, which makes the deduction of individual agent contributions difficult, especially in large MAS with many agents. This can lead to poor policies, since it is unclear which agent policy needs to adapt to what extent in order to improve global MAS behavior [5, 10, 34].

Value function factorization (VFF) via end-to-end deep learning is a popular approach to MARL in order to address the credit assignment problem [28, 31, 34]. A centralized value function is learned from global rewards and factorized into local value functions, which can be used to realize coordinated local policies via multi-armed bandits [28, 35] or local actor-critic learning [27, 33].

Despite the popularity of VFF, most approaches have been only evaluated in domains with a handful of agents. We hypothesize that this is due to the flat factorization scheme of current VFF approaches (Fig. 1a). With an increasing number of agents, the centralized VFF operator becomes a *performance bottleneck*, where it gets difficult to provide sufficiently informative training signal for each agent.

To alleviate this performance bottleneck problem, we propose VFF with *variable agent sub-teams (VAST)*. Instead of directly factorizing a centralized value function for each agent, VAST approximates a factorization for agent sub-teams which can be defined in an arbitrary way and vary over time, e.g., to adapt to different situations. The sub-team values are then linearly decomposed for all sub-team members as illustrated in Fig. 1b. Therefore, VAST can learn on a more focused and compact input representation of the original VFF operator. Our contributions are as follows:

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Flat value function factorization for $N = 5$ agents (b) Factorization for $K = 2$ agent sub-teams

Figure 1: Illustration of different value function factorization schemes using a factorization operator Ψ. (a) Flat factorization directly based on local values Q_i per agent $i \in \mathcal{D}$. (b) Proposed factorization based on $K \leq N = |\mathcal{D}|$ sub-team values $Q^G_{t,k}$, which are linearly decomposed into local values Q_j per sub-team member $j \in G_{t,k} \subseteq \mathcal{D}$. Sub-team $G_{t,k}$ is defined by a assignment strategy (Section 4).

- We formulate VAST and show that VAST maintains decentralizability like state-of-the-art VFF given any sub-team assignment and depending on the sub-team based VFF operator.
- We propose a meta-gradient approach to optimize sub-team assignments in order to adapt and improve VAST. We also briefly discuss alternative sub-team assignment strategies.
- We empirically evaluate different variants of VAST in three multi-agent domains and show that VAST can significantly outperform flat state-of-the-art VFF approaches by alleviating the performance bottleneck problem, when the number of agents is sufficiently large.

2 Background

We model cooperative MAS as partially observable *Markov game* $M = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \Omega \rangle$, where $\mathcal{D} = \{1, ..., N\}$ is a set of agents i, $\mathcal{S}$ is a set of states s_t at time step t, $\mathcal{A} = \langle \mathcal{A}_i \rangle_{i \in \mathcal{D}}$ is the set of joint actions $a_t = \langle a_{t,i} \rangle_{i \in \mathcal{D}} = \langle a_{t,1}, ..., a_{t,N} \rangle$, $\mathcal{P}(s_{t+1}|s_t, a_t)$ is the transition probability, $r_t = \mathcal{R}(s_t, a_t) \in \mathbb{R}$ is the global reward, $\mathcal{Z}$ is a set of local observations $z_{t,i}$ for each agent i, and $\Omega(s_t, a_t) = z_{t+1} = \langle z_{t+1,i} \rangle_{i \in \mathcal{D}} \in \mathcal{Z}^N$ is the subsequent joint observation. Each agent i maintains a local *history* $\tau_{t,i} \in (\mathcal{Z} \times \mathcal{A}_i)^t$ and $\tau_t = \langle \tau_{t,i} \rangle_{i \in \mathcal{D}}$ is the *joint history*. $\pi(a_t|\tau_t) = \langle \pi_i(a_{t,i}|\tau_{t,i}) \rangle_{i \in \mathcal{D}}$ is the (joint) action probability of *joint policy* π, where π_i is the *local policy* of agent i. π can be evaluated with a *value function* $Q^\pi(s_t, a_t) = \mathbb{E}_\pi[R_t|s_t, a_t], \forall s_t \in \mathcal{S}, \forall a_t \in \mathcal{A}$, where $R_t = \sum_{x=0}^{\infty} \gamma^x r_{t+x}$ is the *return* with $\gamma \in [0, 1)$. The goal is to find an *optimal joint policy* $\pi^* = \langle \pi^*_i \rangle_{i \in \mathcal{D}}$ with $Q^{\pi^*} = Q^* = max_\pi Q^\pi$. If Q^* is known, π^* can be obtained by greedily maximizing Q^*.

Note: Expressions of the form $\langle e_i \rangle_{i \in \mathcal{I}}$ denote *unordered sets*, where e_i is mapped to exactly one identifier (e.g., an agent) $i \in \mathcal{I}$. Thus, we implicitly assume the order of agents to be irrelevant.

2.1 Independent Learning of Value Functions

Q^* can be approximated independently by each agent $i \in \mathcal{D}$ using naive decentralized MARL on $a_{t,i}$ and $\tau_{t,i}$ [10, 18, 35]. These local approximations $Q_i \sim Q^*$ can be used to realize local polices π_i for each agent i by using, e.g., multi-armed bandits on Q_i [28, 35] or actor-critic learning with Q_i as critic [10]. *Independent Learning (IL)* offers optimal scalability w.r.t. N but violates the Markov assumption due to non-stationarity caused by simultaneously learning agents [17, 31].

2.2 Value Function Factorization

For many problems, training usually takes place in a laboratory or in a simulated environment, where global information is available. State-of-the-art MARL exploits this fact to approximate a centralized

value function $Q_{tot} \sim Q^*$, which conditions on joint histories τ_t and joint actions a_t (and optionally on global states s_t). However, Q_{tot} is only required *during training* in order to realize local policies π_i, which can be used in a decentralized way because they only condition on the local history $\tau_{t,i}$. This paradigm is known as *centralized training and decentralized execution (CTDE)* [10, 28, 31].

Q_{tot} can be factorized into local value functions $\langle Q_i \rangle_{i \in \mathcal{D}}$ via a *VFF operator* Ψ as shown in Fig. 1a:

$$Q_{tot}(\tau_t, a_t) = \Psi(Q_1(\tau_{t,1}, a_{t,1}), ..., Q_N(\tau_{t,N}, a_{t,N})) \tag{1}$$

In practice, Ψ is realized with deep neural networks, such that $\langle Q_i \rangle_{i \in \mathcal{D}}$ can be learned end-to-end via backpropagation by minimizing the mean squared *TD*(λ) (*temporal difference*) error [28, 31, 34]. A VFF operator Ψ is *decentralizable* when satisfying the *IGM (Individual-Global-Max)* such that [31]:

$$argmax_{a_t \in \mathcal{A}} Q_{tot}(\tau_t, a_t) = \langle argmax_{a_{t,i} \in \mathcal{A}_i} Q_i(\tau_{t,i}, a_{t,i}) \rangle_{i \in \mathcal{D}} \tag{2}$$

VDN (Value Decomposition Networks) [34] formulates Ψ_{VDN} as linear sum such that $Q_{tot}(\tau_t, a_t) = \Psi_{VDN}(\cdot) = \sum_{i \in \mathcal{D}} Q_i(\tau_{t,i}, a_{t,i})$, which satisfies the IGM for Q_{tot} and $\langle Q_i \rangle_{i \in \mathcal{D}}$ [31].

QMIX [28] formulates Ψ_{QMIX} as a nonlinear monotonic combination of $\langle Q_i \rangle_{i \in \mathcal{D}}$ with a mixing network. The mixing network is generated by hypernetworks [12] and has nonnegative weights to satisfy the monotonicity condition $\frac{\delta Q_{tot}}{\delta Q_i} \geq 0, \forall i \in \mathcal{D}$ to maintain consistency w.r.t. the IGM [28, 31].

QTRAN [31] avoids the additivity and monotonicity constraints of VDN and QMIX respectively by formulating the more general Ψ_{QTRAN}, which aims to satisfy

$$\sum_{i \in \mathcal{D}} Q_i(\tau_{t,i}, a_{t,i}) - Q_{tot}(\tau_t, \mathbf{a_t}) + V_{tot}(\tau_t) = \begin{cases} = 0, \mathbf{a_t} = \overline{\mathbf{a_t}} \\ \geq 0, \mathbf{a_t} \neq \overline{\mathbf{a_t}} \end{cases} \tag{3}$$

where $\overline{\mathbf{a_t}} = \langle \overline{a_{t,i}} \rangle_{i \in \mathcal{D}} = \langle \overline{a_{t,1}}, ..., \overline{a_{t,N}} \rangle$ with $\overline{a_{t,i}} = argmax_{a_{t,i} \in \mathcal{A}_i} Q_i(\tau_{t,i}, a_{t,i})$ and $V_{tot}(\tau_t) = max_{a_t \in \mathcal{A}} Q_{tot}(\tau_t, a_t) - \sum_{i \in \mathcal{D}} Q_i(\tau_{t,i}, \overline{a_{t,i}})$, in order to be consistent w.r.t. the IGM.

3 Related Work

MARL is a long-standing research area with rapid progress towards complex domains [2, 11, 35, 38]. Most state-of-the-art approaches are based on CTDE to learn Q_{tot} for actor-critic learning [10, 21] or VFF [28, 31, 34]. VFF approaches like VDN, QMIX, and QTRAN use a flat factorization scheme, where Q_{tot} is directly factorized into $\langle Q_i \rangle_{i \in \mathcal{D}}$ (Fig. 1a). Recent work has used Q_i as critic for local actor-critic learning [27, 33]. We introduce a *hierarchical* VFF approach based on *agent sub-teams* which can vary over time, e.g., to adapt to different situations. With that, we can improve performance in large MAS, where flat VFF approaches could fail due to Ψ becoming a performance bottleneck.

Prior work on hierarchical MARL has mainly focused on temporal abstraction, where the MAS attempts to solve tasks based on temporal subgoals or roles [22, 36, 43]. We focus on VFF applied to agent sub-teams, which can be regarded as an *abstraction of agents*. These abstractions or sub-teams can *vary over time*, depending on the sub-team assignment strategy which may be chosen *arbitrarily*.

There is some prior work on sub-team assignments and agent-based hierarchization in MAS: The relationship between coordination, complexity, and performance depending on predefined organizational MAS structures was studied in [3, 7, 15, 29, 30]. [19] proposed a contextual MARL framework for fleet management, where the spatial environment is partitioned into fixed cells with locally assigned rewards and the number of agents per cell can vary over time. [16] proposed an attention-based mechanism for self-interested MAS to focus on different contextual information per agent in order to approximate Q_i. Mean field MARL was introduced in [44], where Q_i is learned based on the mean field approximation of the joint action of all neighbor agents, where the definition of "neighborhood" is domain dependent. Different clustering approaches for agents, communication messages, etc. w.r.t. some similarity criteria have been proposed in [4, 23, 40]. Our approach addresses the *performance bottleneck problem* of flat VFF approaches. It can be used with *any sub-team assignment strategy* like random assignments, clustering, or meta-learning to structure the MAS *dynamically* while satisfying the IGM for Q_{tot} and $\langle Q_i \rangle_{i \in \mathcal{D}}$ like flat state-of-the-art VFF approaches [28, 31, 34]. Unlike [16, 19, 44], our approach does not depend on predefined local rewards per agent or region but *automatically* approximates local value functions Q_i from global rewards via sub-team based VFF.

4 Value Function Factorization with Variable Agent Sub-Teams (VAST)

4.1 Variable Agent Sub-Teams

We now introduce VFF with *variable agent sub-teams (VAST)*. Given a *sub-team ratio* $\eta \in [\frac{1}{N}; 1]$, VAST divides the set of agents $\mathcal{D}$ into $K = \lceil \eta N \rceil \leq N$ agent *sub-teams* $G_{t,k} \in \mathcal{G}_t$ of *division* $\mathcal{G}_t = \langle G_{t,1}, ..., G_{t,K} \rangle$ at every time step t. Each agent $i \in \mathcal{D}$ is assigned to exactly one sub-team $G_{t,k}$ by a *sub-team assignment strategy* $\mathcal{X}$ with distribution $\mathcal{X}(k|i, \tau_{t,i}, s_t), k \in \{1, ..., K\}$ such that $G_{t,k} \subseteq \mathcal{D}$, $G_{t,k} \cap G_{t,k'} = \emptyset$ if $k \neq k'$, and $\mathcal{D} = \bigcup_{k=1}^{K} G_{t,k}$. Each sub-team $G_{t,k}$ can be regarded as temporary *agent abstraction* which selects *sub-team actions* $a^G_{t,k} = \langle a_{t,j} \rangle_{j \in G_{t,k}}$ based on all *sub-team members* $j \in G_{t,k}$. The value function $Q^G_{t,k}$ of $G_{t,k}$ is computed via Ψ_{VDN} on $\langle Q_j \rangle_{j \in G_{t,k}}$:

$$Q^G_{t,k}(\tau^G_{t,k}, a^G_{t,k}) = \Psi_{VDN}(\cdot) = \sum_{j \in G_{t,k}} Q_j(\tau_{t,j}, a_{t,j}) \tag{4}$$

Despite the simplified approximation of $Q^G_{t,k}$ in Eq. 4, Ψ_{VDN} has two important advantages over nonlinear variants like Ψ_{QMIX} and Ψ_{QTRAN}, which would also satisfy the IGM for $Q^G_{t,k}$ and $\langle Q_j \rangle_{j \in G_{t,k}}$: First, the sum of Ψ_{VDN} has no fixed input dimension, thus sub-team sizes may vary over time, e.g., to adapt to different situations. Second, Ψ_{VDN} does not introduce new tunable hyperparameters, thus being more efficient to use. Therefore, we defer nonlinear approximations of $Q^G_{t,k}$ to future work.

Q_{tot} is approximated from $\langle Q^G_{t,k} \rangle_{G_{t,k} \in \mathcal{G}_t}$ using a VFF operator Ψ according to Eq. 1:

$$Q_{tot}(\tau_t, a_t) = \Psi(Q^G_{t,1}(\tau^G_{t,1}, a^G_{t,1}), ..., Q^G_{t,K}(\tau^G_{t,K}, a^G_{t,K})) \tag{5}$$

where $K = \lceil \eta N \rceil$ specifies the input dimension of Ψ. The computation hierarchy of Q_{tot} based on VAST according to Eq. 4 and 5 is depicted in Fig. 1b. With that hierarchy, $\langle Q_i \rangle_{i \in \mathcal{D}}$ can be learned end-to-end, e.g., via backpropagation by updating Ψ w.r.t. the mean squared $TD(\lambda)$ error.

Algorithm 1 Variable Agent Sub-Teams (VAST)

procedure *VAST*$(M, \Psi, \mathcal{X}, \eta)$
  Initialize parameters of Ψ, $\mathcal{X}$, $\langle Q_i \rangle_{i \in \mathcal{D}}$
  $K \leftarrow \lceil \eta N \rceil$
  for episode $x \leftarrow 1, T$ **do**
    Sample s_1, oberve z_1
    for time step t **do**
      $a_t \sim \pi(a_t|\tau_t)$
      $r_t, z_{t+1} \leftarrow \mathcal{R}(s_t, a_t), \Omega(s_t, a_t)$
      $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$
      for sub-team $k \leftarrow 1, K$ **do**
        $G_{t,k} \leftarrow \{\}$
      $\mathcal{G}_t \leftarrow \langle G_{t,1}, ..., G_{t,K} \rangle$
      for agent $i \in \mathcal{D}$ **do**
        $k \sim \mathcal{X}(k|i, \tau_{t,i}, s_t)$
        $G_{t,k} \leftarrow G_{t,k} \cup \{i\}$
      $Q^G_{t,k}(\tau^G_{t,k}, a^G_{t,k}) \leftarrow Eq.4, \forall G_{t,k}$
      Update Ψ, $\langle Q_i \rangle_{i \in \mathcal{D}}$ with $TD(\lambda)$
    Update $\mathcal{X}$ (e.g., Eq. 6) ▷ optional

VAST is formulated in Algorithm 1, where M is the MAS, Ψ is an IGM preserving *VFF operator* like Ψ_{VDN}, Ψ_{QMIX}, or Ψ_{QTRAN} as listed in Section 2.2 to approximate Q_{tot} from $\langle Q^G_{t,k} \rangle_{G_{t,k} \in \mathcal{G}_t}$, $\mathcal{X}$ is a sub-team assignment strategy, and $\eta \in [\frac{1}{N}; 1]$ is the sub-team ratio.

η specifies the degree of input space compression of Ψ. The smaller η, the more compact the input representation of Ψ. In the extreme case of $\eta = \frac{1}{N} \Rightarrow K = 1$, the factorization reduces to $Q_{tot}(\tau_t, a_t) = \Psi(\Psi_{VDN}(\cdot)) = \Psi(\sum_{i \in \mathcal{D}} Q_i(\tau_{t,i}, a_{t,i}))$. Larger values of η enable more exploration of the input space of Ψ but at the cost of more compute, which increases linearly w.r.t. η. Furthermore, we suggest $\frac{1}{N} < \eta \ll N$ for large N to alleviate the original performance bottleneck problem of Ψ.

To show that VAST maintains decentralizability by satisfying the IGM for Q_{tot} and $\langle Q_i \rangle_{i \in \mathcal{D}}$ for an *arbitrary* sub-team assignment strategy $\mathcal{X}$, we formulate and prove Theorem 1:

Theorem 1. *Given a MAS $M = \langle \mathcal{D}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{Z}, \Omega \rangle$ at time step t, where each agent $i \in \mathcal{D}$ with local value function Q_i is assigned to exactly one sub-team $G_{t,k} \in \mathcal{G}_t$: If the IGM is satisfied for a factorization of the centralized value function Q_{tot} into sub-team value functions $\langle Q^G_{t,k} \rangle_{G_{t,k} \in \mathcal{G}_t}$ via a VFF operator Ψ according to Eq. 5, then the IGM is also satisfied for Q_{tot} and $\langle Q_i \rangle_{i \in \mathcal{D}}$ for each agent $i \in \mathcal{D} = G_{t,1} \cup ... \cup G_{t,K}$.*

Table 1: Characteristics of different sub-team assignment strategies $\mathcal{X}$. The worst case complexity indicates the computational overhead per time step t and agent i, when invoking $\mathcal{X}$ (line 15 in Algorithm 1) or updating $\mathcal{X}$ (line 18 in Algorithm 1) if all other parameters (e.g., η) are constant.

Approach	Description	Worst case complexity	Domain knowledge
$\mathcal{X}_{Random}$	Random assignment with $\mathcal{X}(k\|i,\tau_{t,i},s_t)=\frac{1}{K}$	$\mathcal{O}(1)$	None
$\mathcal{X}_{Fixed}$	Fixed assignment with deterministic $\mathcal{X}$	$\mathcal{O}(1)$	Agent IDs
$\mathcal{X}_{Spatial}$	Spatial clustering of agents to specify $\mathcal{X}$	$\mathcal{O}(N^C), C>1$	Spatial information
$\mathcal{X}_{MetaGrad}$	Meta-gradient learning of $\mathcal{X}(k\|i,\tau_{t,i},s_t)$ (Eq. 6)	$\mathcal{O}(N)$	Optional

Proof. The factorizations learned via Ψ and Ψ_{VDN} (Eq. 4) satisfy the IGM such that $\bar{\mathbf{a}}_\mathbf{t} = \langle \bar{\mathbf{a}}^\mathbf{G}_{\mathbf{t},\mathbf{k}} \rangle_{G_{t,k}\in\mathcal{G}_t}$ and $\bar{\mathbf{a}}^\mathbf{G}_{\mathbf{t},\mathbf{k}} = \langle \bar{a}_{t,i} \rangle_{i\in G_{t,k}}$ respectively, where $\bar{\mathbf{a}}_\mathbf{t} = argmax_{a_t\in\mathcal{A}} Q_{tot}(\tau_t, a_t)$, $\bar{\mathbf{a}}^\mathbf{G}_{\mathbf{t},\mathbf{k}} = argmax_{a^G_{t,k}\in\langle\mathcal{A}_i\rangle_{i\in G_{t,k}}} Q^G_{t,k}(\tau^G_{t,k}, a^G_{t,k})$, and $\bar{a}_{t,i} = argmax_{a_{t,i}\in\mathcal{A}_i} Q_i(\tau_{t,i}, a_{t,i})$:

$$\bar{\mathbf{a}}_\mathbf{t} \overset{\Psi}{=} \langle \bar{\mathbf{a}}^\mathbf{G}_{\mathbf{t},\mathbf{k}} \rangle_{G_{t,k}\in\mathcal{G}_t} \overset{\Psi_{VDN},Eq.4}{=} \langle \langle \bar{a}_{t,i} \rangle_{i\in G_{t,k}} \rangle_{G_{t,k}\in\mathcal{G}_t} \overset{\mathcal{D}=G_{t,1}\cup\ldots\cup G_{t,K}}{=} \langle \bar{a}_{t,i} \rangle_{i\in\mathcal{D}}$$

Therefore, the set of greedy local actions of all agents $\langle \bar{a}_{t,i} \rangle_{i\in\mathcal{D}} = \langle \bar{a}_{t,1}, ..., \bar{a}_{t,N} \rangle = \bar{\mathbf{a}}_\mathbf{t}$ maximizes Q_{tot} for *any* sub-team assignment according to the IGM in Eq. 2. □

4.2 Sub-Team Assignment Strategies

We propose the meta-gradient based sub-team assignment strategy $\mathcal{X}_{MetaGrad}$ similar to [42] in order to learn an adaptive assignment distribution for different situations and further improvement of VAST. $\mathcal{X}_{MetaGrad}$ is approximated with parameter vector θ, which is optimized via gradient ascent (line 18 in Algorithm 1) on some (high-level) objective $J(\theta)$ w.r.t. to the following estimated gradient:

$$g = \hat{A}(k,i,\tau_{t,i},s_t)\nabla_\theta log\mathcal{X}_{MetaGrad}(k|i,\tau_{t,i},s_t) \tag{6}$$

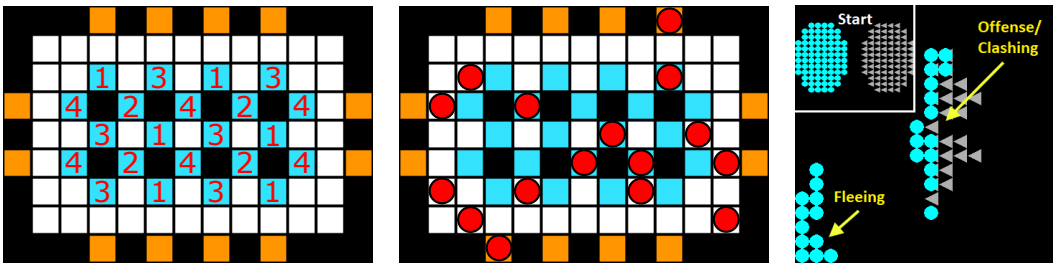
where $\hat{A}(k,i,\tau_{t,i},s_t) = \hat{Q}(k,i,\tau_{t,i},s_t) - \hat{V}(i,\tau_{t,i},s_t)$ is the advantage of k for sub-team $G_{t,k}\in\mathcal{G}_t$ given i, $\tau_{t,i}$, and s_t. $\hat{Q}$ estimates the (expected) performance when selecting k given i, $\tau_{t,i}$, and s_t. $\hat{V}$ represents a baseline function, which can depend on i, $\tau_{t,i}$, and s_t, for variance reduction. The concrete definitions of $\hat{Q}$ and $\hat{V}$ are based on $J(\theta)$, which should correlate with the original target of Q_{tot} and can optionally integrate domain knowledge. In this paper, we estimate $\hat{A}(k,i,\tau_{t,i},s_t)$ by setting $\hat{Q}(k,i,\tau_{t,i},s_t) = R_t$ to the return as defined in Section 2 and $\hat{V}(i,\tau_{t,i},s_t) = \sum_{a_{t,i}\in\mathcal{A}_i}\pi_i(a_{t,i}|\tau_{t,i})Q_i(\tau_{t,i},a_{t,i})$ to the expected local value of agent i with value function Q_i and local policy π_i to avoid any additional domain dependencies.

Table 1 lists some alternative sub-team assignment strategies for comparison: $\mathcal{X}_{Random}$ assigns each agent to a random sub-team at every time step, while $\mathcal{X}_{Fixed}$ permanently assigns each agent to a particular sub-team based on its ID. $\mathcal{X}_{Spatial}$ uses spatial information like coordinates to cluster agents in order to form sub-teams. $\mathcal{X}_{MetaGrad}$ optimizes sub-team assignments w.r.t. some (high-level) objective to adapt to different situations and to further improve VAST. $\mathcal{X}_{Spatial}$ and $\mathcal{X}_{MetaGrad}$ introduce additional computational overhead per time step and agent depending on N.

5 Experimental Setup

5.1 Evaluation Domains

Warehouse[N] is a grid-world environment inspired by [7, 9, 39] and illustrated in Fig. 2a-b, where N agents or robots have to pick up randomly generated *orders* of 5 *items* $b_w \in \{1,2,3,4\}, w \in \{1,...,5\}$ at *work stations* (orange cells) and deliver each item b_w to its corresponding *drop off location* (cyan cells), where the drop off number according to Fig. 2a matches b_w. All agents start at random work stations. After delivering all items of an order, the agent can return to any work station for a new order. All agents have a 5×5 field of view and are able to drop off their items if possible, move north, south, west, east, or do nothing. Agents cannot share positions or occupy black obstacle cells. A successfully delivered item is rewarded with +1. Collisions with other agents are penalized with -0.5. At every time step, there is a time penalty of -0.01. An episode ends after 50 time steps.



(a) *Warehouse[N=16]* (layout) (b) *Warehouse[N=16]* (agents) (c) *Battle[N=80]*

Figure 2: Illustration of the *Warehouse[N]* and the *Battle[N]* domain. (a) Work stations (orange cells) and drop off locations (cyan cells) in *Warehouse[16]*. (b) All agents (red circles) need to pick up orders of 5 items $b_w \in \{1, 2, 3, 4\}$ at the work stations and deliver them to the corresponding drop off locations according to (a) while avoiding stalling and collisions with other agents. (c) An army of learning agents (cyan circles) has to fight another army of opponent agents (gray triangles).

Battle[N] is a grid-world environment inspired by [46] and illustrated in Fig. 2c, where an army of N learning agents (cyan circles) has to fight another army of N *opponent* agents (gray triangles), which randomly move towards and attack all learning agents within sight. Each agent i initially has 3 *health points* (HP_i), which are recovered by 0.01 at each time step when $0 < HP_i < 3$. An agent i is *dead* or *killed* when $HP_i = 0$. All agents have a 7×7 field of view and are able to move north, south, west, east, do nothing, or attack one opponent if occupying the same cell, resulting in the attacked opponent's loss of one health point. Successful attacks and kills are rewarded with +1. Attacking a cell without any opponent is penalized with -0.1 and being hit or killed by the opponent is penalized with -0.5. An episode ends after 100 time steps or when all agents of an army have been killed.

GaussianSqueeze[N] is a single-step multi-agent resource allocation problem introduced in [14], where N agents have to coordinate their actions $a_{t,i} \in \mathcal{A}_i = \{0, ..., 9\}$ to find the most efficient *allocation* $\zeta = \sum_{i=1}^{N} a_{t,i}$. The *system performance* is defined by $J(\zeta) = \zeta e^{\frac{-(\zeta-\mu)^2}{\sigma^2}}$ and μ and σ are domain dependent parameters, which we set to $\mu = 400$ and $\sigma = 200$.

5.2 Learning Algorithms and Training

We implemented IL, QMIX, and QTRAN as baselines. For VAST, we use the notation VAST($\Psi, \mathcal{X}, \eta$) with VFF operator $\Psi \in \{\Psi_{IL}, \Psi_{VDN}, \Psi_{QMIX}, \Psi_{QTRAN}\}$ (Ψ_{IL} approximates $Q^G_{t,k} = Q_{tot}$ independently), sub-team assignment strategy $\mathcal{X} \in \{\mathcal{X}_{Random}, \mathcal{X}_{Fixed}, \mathcal{X}_{Spatial}, \mathcal{X}_{MetaGrad}\}$, and sub-team ratio $\eta \in \{\frac{1}{4}, \frac{1}{2}\}$ (Algorithm 1). Since value-based algorithms are highly sensitive w.r.t. the exploration decay schedule, we completely omit exploration tuning and use Q_i as critic for local actor-critic learning to realize π_i instead [27, 33]. $\mathcal{X}_{Fixed}$ assigns each agent i to sub-team $G_{t,k}$ with $k = i \pmod K + 1$. $\mathcal{X}_{Spatial}$ uses k-means clustering on the agents' (x, y)-positions in *Warehouse[N]* and *Battle[N]* with $\frac{K}{2}$ centroids. If not stated otherwise, we assume the following defaults: $\Psi = \Psi_{QTRAN}$, $\mathcal{X} = \mathcal{X}_{MetaGrad}$.

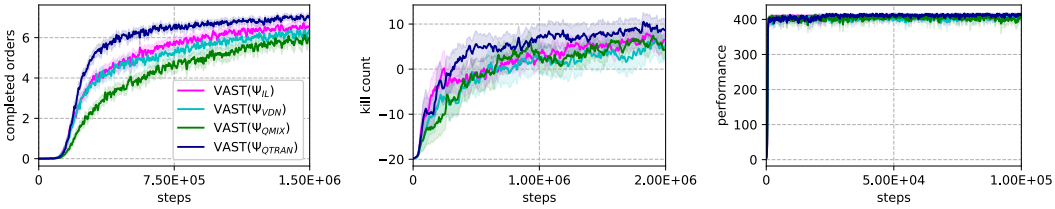
We performed 30 training runs for each MARL algorithm and report the domain-specific performance, i.e., the number of *completed orders* in *Warehouse[N]*, the *kill count* in *Battle[N]* (kills by opponent agents are counted negatively), and the *system performance* in *GaussianSqueeze[N]* respectively.

Further details on the training setup and the experiments are specified in Appendix A.1 and A.2.

6 Results

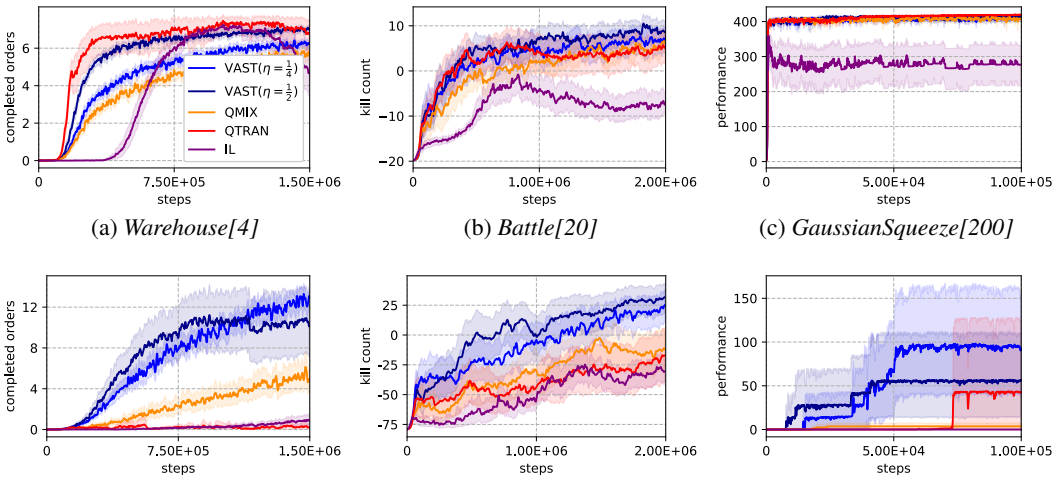
6.1 Comparison of Value Function Factorization Operators for VAST

We ran VAST with different VFF operators $\Psi \in \{\Psi_{IL}, \Psi_{VDN}, \Psi_{QMIX}, \Psi_{QTRAN}\}$, $\mathcal{X} = \mathcal{X}_{MetaGrad}$, and $\eta = \frac{1}{2}$ in *Warehouse[4]*, *Battle[20]*, and *GaussianSqueeze[200]*. The results are shown in Fig. 3. All variants show steady learning progress in all domains. VAST(Ψ_{QTRAN}) performs best in *Warehouse[4]* and *Battle[20]*. In *GaussianSqueeze[200]*, all variants perform equally well.



(a) *Warehouse[4]* (b) *Battle[20]* (c) *GaussianSqueeze[200]*

Figure 3: Average training progress of VAST with $\Psi \in \{\Psi_{IL}, \Psi_{VDN}, \Psi_{QMIX}, \Psi_{QTRAN}\}$, $\mathcal{X}_{MetaGrad}$, and $\eta = \frac{1}{2}$. Shaded areas show the 95% confidence interval. Legend in (a) applies to all plots.



(a) *Warehouse[4]* (b) *Battle[20]* (c) *GaussianSqueeze[200]*

(d) *Warehouse[16]* (e) *Battle[80]* (f) *GaussianSqueeze[800]*

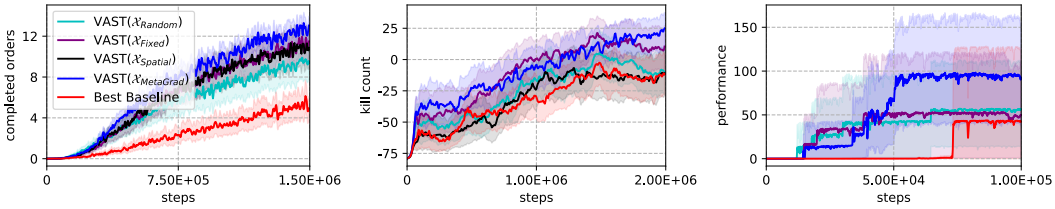
Figure 4: Average training progress of VAST with $\eta \in \{\frac{1}{4}, \frac{1}{2}\}$, Ψ_{QTRAN}, and $\mathcal{X}_{MetaGrad}$ as well as QMIX, QTRAN, and IL. Shaded areas show the 95% confidence interval. Legend in (a) applies to all plots. The full results of the state-of-the-art comparison are shown in Fig. 8 in Appendix A.3.1.

6.2 State-of-the-Art Comparison

We ran VAST with different sub-team ratios $\eta \in \{\frac{1}{4}, \frac{1}{2}\}$, $\Psi = \Psi_{QTRAN}$, and $\mathcal{X} = \mathcal{X}_{MetaGrad}$ in *Warehouse[4]*, *Battle[20]*, and *GaussianSqueeze[200]* as well as in larger instances, i.e., *Warehouse[16]*, *Battle[80]*, and *GaussianSqueeze[800]* (medium instances are shown in Appendix A.3.1) to compare the performance with QMIX, QTRAN, and IL as shown in Fig. 4. In *Warehouse[4]*, QTRAN makes slightly faster progress than VAST($\eta = \frac{1}{2}$). However in *Warehouse[16]*, both VAST variants outperform all baselines, which perform poorly. In *Battle[20]*, both VAST variants slightly outperform QMIX and QTRAN, but they perform significantly better in *Battle[80]*. In *GaussianSqueeze[200]*, all CTDE approaches perform equally well, but both VAST variants clearly outperform all baselines in *GaussianSqueeze[800]*. VAST($\eta = \frac{1}{2}$) initially improves faster than VAST($\eta = \frac{1}{4}$) in most domains but in *Warehouse[16]* and *GaussianSqueeze[800]*, VAST($\eta = \frac{1}{4}$) surpasses VAST($\eta = \frac{1}{2}$) over time.

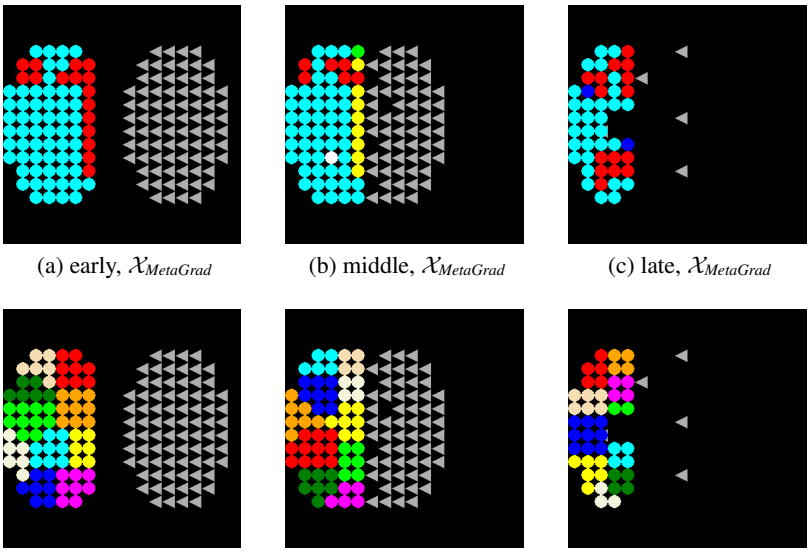
6.3 Comparison of Sub-Team Assignment Strategies for VAST

We ran VAST with different $\mathcal{X} \in \{\mathcal{X}_{Random}, \mathcal{X}_{Fixed}, \mathcal{X}_{Spatial}, \mathcal{X}_{MetaGrad}\}$, $\Psi = \Psi_{QTRAN}$, $\eta = \frac{1}{4}$ in *Warehouse[16]*, *Battle[80]*, and *GaussianSqueeze[800]* to compare the performance with the respective best baselines from Fig. 4 in Section 6.2. $\mathcal{X}_{Spatial}$ was not tested in *GaussianSqueeze[800]*, due to the lack of spatial information. The results are shown in Fig. 5. VAST($\mathcal{X}_{MetaGrad}$) performs best in all domains. In *Battle[80]*, VAST($\mathcal{X}_{Fixed}$) is competitive to VAST($\mathcal{X}_{MetaGrad}$) while VAST($\mathcal{X}_{Random}$) and VAST($\mathcal{X}_{Spatial}$) are competitive to the best baseline. In *Warehouse[16]* and *GaussianSqueeze[800]*, all VAST variants clearly outperform the respective best baselines.



(a) *Warehouse[16]* (b) *Battle[80]* (c) *GaussianSqueeze[800]*

Figure 5: Average training progress of VAST with $\mathcal{X} \in \{\mathcal{X}_{Random}, \mathcal{X}_{Fixed}, \mathcal{X}_{Spatial}, \mathcal{X}_{MetaGrad}\}$, Ψ_{QTRAN}, $\eta = \frac{1}{4}$. Shaded areas show the 95% confidence interval. Legend in (a) applies to all plots.



(a) early, $\mathcal{X}_{MetaGrad}$ (b) middle, $\mathcal{X}_{MetaGrad}$ (c) late, $\mathcal{X}_{MetaGrad}$

(d) early, $\mathcal{X}_{Spatial}$ (e) middle, $\mathcal{X}_{Spatial}$ (f) late, $\mathcal{X}_{Spatial}$

Figure 6: Visualizations of the generated sub-teams of $\mathcal{X}_{MetaGrad}$ with $\eta = \frac{1}{4}$ and $\mathcal{X}_{Spatial}$ with k-means clustering using 10 centroids at different stages (early, middle, late) in *Battle[80]* after training. All learning agents (round circles) of the same sub-team have the same color.

We further examined the generated sub-teams of $\mathcal{X}_{MetaGrad}$ and $\mathcal{X}_{Spatial}$ at different stages in *Battle[80]* by visualizing all agents of the same sub-team with the same color in Fig. 6. In the early stage (Fig. 6a), $\mathcal{X}_{MetaGrad}$ generates a cyan sub-team for agents that are rather far away from the opponent army and a red sub-team which is rather close to it (with some prediction noise). In the middle stage (Fig. 6b), a yellow sub-team emerges, when both armies clash, which disappears later (Fig 6c), when the opponent army is significantly decimated, thus reverting back to the cyan and red sub-teams depending on the agent positions. $\mathcal{X}_{Spatial}$ simply groups agents depending on their spatial distances to each other with no obvious relation to the danger of the current situation as shown in Fig. 6d-f.

Since the opponent army follows an offensive strategy, most learned policies adopted a defensive strategy, where all agents group and defend themselves together like in Fig. 6. However, in some cases, VAST learned a "splitting" strategy, where the army splits into a *fleeing part* to reduce overall deaths and an *offensive part* that clashes with the opponent army to increase the kill count as shown in Fig. 2c. The generated sub-teams of the splitting strategy are shown in Fig. 10 in Appendix A.3.3.

7 Discussion

We proposed VAST to approximate value function factorizations for agent sub-teams which can be defined in an arbitrary way and vary over time, e.g., to adapt to different situations. The sub-team values are then linearly decomposed for all sub-team members. VAST learns on a more focused

and compact input representation of the original VFF operator, thus being able to better address the multi-agent credit assignment problem in larger MAS than flat state-of-the-art VFF approaches.

Our experiments show that VAST is able to learn with different VFF operators Ψ to improve performance in domains, where flat VFF approaches could fail to learn meaningful factorizations. This is clearly shown for QTRAN and VAST(Ψ_{QTRAN}) in the large MAS instances *Warehouse[16]*, *Battle[80]*, and *GaussianSqueeze[800]* in Fig. 4d-f, where VAST(Ψ_{QTRAN}) significantly outperforms QTRAN. The difference between IL and VAST(Ψ_{IL}) can already be seen in the small MAS instances *Warehouse[4]*, *Battle[20]* and *GaussianSqueeze[200]*, where IL lacks stability in Fig. 4a-c, while VAST(Ψ_{IL}) improves steadily in these domains as shown in Fig. 3a-c. VAST can significantly outperform flat state-of-the-art VFF approaches like QMIX and QTRAN by alleviating the performance bottleneck problem, when the number of agents is sufficiently large as shown in Fig. 4d-f and Fig. 5.

VAST achieves competitive or superior performance with arbitrary sub-team assignment strategies $\mathcal{X}$ as shown in Fig. 5, which is supported by Theorem 1. $\mathcal{X}_{MetaGrad}$ is an adaptive approach, which optimizes sub-team assignments to further improve VAST. In *Battle[80]*, $\mathcal{X}_{MetaGrad}$ is able to meaningfully structure the MAS according to different situations, which might be more beneficial for VAST than just relying on simple domain dependent features like agent IDs [1, 10] or spatial positions [19, 44] as shown in Fig. 6 and 10 in Appendix A.3.3. However, $\mathcal{X}_{MetaGrad}$ introduces additional computational overhead, which scales linearly per agent w.r.t. N as stated in Table 1. Furthermore, the learning quality strongly depends on the objective definition of $\mathcal{X}_{MetaGrad}$ [42].

In Fig. 4d and Fig. 4f, VAST($\eta = \frac{1}{2}$) itself suffers from the performance bottleneck (but to a much lesser extent than flat VFF), where VAST($\eta = \frac{1}{4}$) improves more stably and surpasses VAST($\eta = \frac{1}{2}$) over time. However, VAST($\eta = \frac{1}{2}$) is superior in *Battle[80]* and performs better in the early training stages in *Warehouse[16]*. In Fig. 9 in Appendix A.3.2, this is indicated by more exploration through a higher sub-team division diversity. Due to the potential performance bottleneck of Ψ and the computational scaling w.r.t. η, we recommend VAST for large MAS with $\frac{1}{N} < \eta \ll N$ for high efficiency and performance. A self-tuning mechanism for η would be interesting for future work.

The linear approximation of sub-team values ensures flexibility w.r.t. sub-teams and makes additional hierarchization of sub-teams obsolete, due to the associative property of additions (e.g., $(a + b) + (c + d) = a + b + c + d$). However, this might be too restrictive for some domains. Using nonlinear variants like recurrent neural networks [6, 8, 13] or transformers [26, 37] could further improve flexibility and performance but requires more compute and yields higher complexity due to more hyperparameters. Thus, we defer an investigation on more flexible VAST schemes to future work.

8 Potential Negative Societal Impacts

The goal of our work is to realize autonomous systems to solve complex tasks at large scale in a distributed way as motivated in Section 1. To focus completely on the potential effects of our work, we refer to [41] for a general overview regarding societal implications of deep reinforcement learning.

VAST is a CTDE approach with a centralized training regime to realize decentralized policies. These policies have a common objective which might include bias of a central authority and can cause harm to opposing parties, e.g., via discrimination or misleading information. Since we assume VAST to be trained in a laboratory or in a simulation, the trained system might exhibit unsafe behavior when being deployed into the real world due to poor generalization, e.g., by causing traffic accidents. Depending on the choice of Ψ, $\mathcal{X}$, and η, some computational overhead is added to the original VFF approach, which can be significant when scaling up. The generated sub-teams can potentially be used to evaluate and categorize living beings w.r.t. some assignment strategy $\mathcal{X}$ and objective as shown in Fig. 6 and 10 in Appendix A.3.3, which could lead to misuse or discrimination of particular groups.

As experimentally shown in the *Battle[N]* domain, VAST can be misused for real combat, e.g., in autonomous weapon systems to realize coordinated and distributed strategies as demonstrated in Fig. 6 and 10 in Appendix A.3.3. MAS trained with VAST can be assumed to be resilient w.r.t. single agent failures (e.g., agent deaths in *Battle[N]*), which can make human intervention (e.g., shutting down the MAS by disabling single agents) difficult. Behavioral changes of single agents due to updates, failures, or malicious attacks could lead to unexpected emergent effects like adaptive reorganizations of sub-teams, which can cause, e.g., traffic jams, outages of critical infrastructure, or directly harm to others, depending on the quality of the learned policies and the common objective.

References

[1] Craig Boutilier. Planning, Learning and Coordination in Multiagent Decision Processes. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210. Morgan Kaufmann Publishers Inc., 1996.

[2] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-Agent Reinforcement Learning: An Overview. In *Innovations in Multi-Agent Systems and Applications-1*, pages 183–221. Springer, 2010.

[3] Kathleen M Carley and Les Gasser. Computational Organization Theory. *Multiagent systems: A modern approach to distributed artificial intelligence*, pages 299–330, 1999.

[4] Santhana Chaimontree, Katie Atkinson, and Frans Coenen. A Framework for Multi-Agent Based Clustering. *Autonomous Agents and Multi-Agent Systems*, 25(3):425–446, 2012.

[5] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. All Learning is Local: Multi-Agent Learning in Global Reward Games. In *Advances in Neural Information Processing Systems*, pages 807–814, 2004.

[6] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[7] Jen Jen Chung, Damjan Miklić, Lorenzo Sabattini, Kagan Tumer, and Roland Siegwart. The Impact of Agent Definitions and Interactions on Multiagent Learning for Coordination. In *AAMAS'19 Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1752–1760. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

[8] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.

[9] Daniel Claes, Frans Oliehoek, Hendrik Baier, and Karl Tuyls. Decentralised Online Planning for Multi-Robot Warehouse Commissioning. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, pages 492–500. IFAAMAS, 2017.

[10] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual Multi-Agent Policy Gradients. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[11] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative Multi-Agent Control using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[12] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Chris HolmesParker, M Taylor, Yusen Zhan, and Kagan Tumer. Exploiting Structure and Agent-Centric Rewards to Promote Coordination in Large Multiagent Systems. In *Adaptive and Learning Agents Workshop*, 2014.

[15] Bryan Horling and Victor Lesser. A Survey of Multi-Agent Organizational Paradigms. *Knowledge Engineering Review*, 19(4):281–316, 2004.

[16] Shariq Iqbal and Fei Sha. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2961–2970, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[17] Guillaume J Laurent, Laëtitia Matignon, Le Fort-Piat, et al. The World of Independent Learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(1):55–64, 2011.

[18] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-Agent Reinforcement Learning in Sequential Social Dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and Multiagent Systems*, AAMAS '17, page 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[19] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1774–1783. ACM, 2018.

[20] Jiajing Ling, Tarun Gupta, and Akshat Kumar. Reinforcement Learning for Zone Based Multiagent Pathfinding under Uncertainty. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 551–559, 2020.

[21] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[22] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. Hierarchical Multi-Agent Reinforcement Learning. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 246–253, 2001.

[23] Elth Ogston, Benno Overeinder, Maarten Van Steen, and Frances Brazier. A Method for Decentralized Clustering in Large Multi-Agent Systems. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 789–796, 2003.

[24] Frans A Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*, volume 1. Springer, 2016.

[25] Liviu Panait and Sean Luke. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multiagent Systems*, 11(3):387–434, 2005.

[26] Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphael Lopez Kaufman, Aidan Clark, Seb Noury, et al. Stabilizing Transformers for Reinforcement Learning. In *International Conference on Machine Learning*, pages 7487–7498. PMLR, 2020.

[27] Bei Peng, Tabish Rashid, Christian A. Schroeder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. FACMAC: Factored Multi-Agent Centralised Policy Gradients, 2021.

[28] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4295–4304. PMLR, 2018.

[29] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition Structure Generation with Worst Case Guarantees. *Artificial intelligence*, 111(1-2):209–238, 1999.

[30] Young-pa So and Edmund H Durfee. Designing Tree-Structured Organizations for Computational Agents. *Computational & Mathematical Organization Theory*, 2(3):219–245, 1996.

[31] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5887–5896. PMLR, 2019.

[32] Peter Stone and Manuela Veloso. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[33] Jianyu Su, Stephen Adams, and Peter A Beling. Value-Decomposition Multi-Agent Actor-Critics. In *Proceedings of the Twenty-Fourth AAAI*, 2021.

[34] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-Decomposition Networks for Cooperative Multi-Agent Learning based on Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (Extended Abstract)*, AAMAS '18, page 2085–2087. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[35] Ming Tan. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, pages 330–337. Morgan Kaufmann Publishers Inc., 1993.

[36] Hongyao Tang, Jianye Hao, Tangjie Lv, Yingfeng Chen, Zongzhang Zhang, Hangtian Jia, Chunxu Ren, Yan Zheng, Zhaopeng Meng, Changjie Fan, et al. Hierarchical Deep Multiagent Reinforcement Learning with Temporal Abstraction. *arXiv e-prints*, pages arXiv–1809, 2018.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[38] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature*, pages 1–5, 2019.

[39] Qian Wan, Chonglin Gu, Sankui Sun, Mengxia Chen, Hejiao Huang, and Xiaohua Jia. Lifelong Multi-Agent Path Finding in a Dynamic Environment. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 875–882. IEEE, 2018.

[40] Xin Wen, Zheng-Jun Zha, Zilei Wang, Liansheng Zhuang, and Houqiang Li. CCNet: Cluster-Coordinated Net for Learning Multi-Agent Communication Protocols with Reinforcement Learning. In *Proceedings of The 10th Asian Conference on Machine Learning*, volume 95 of *Proceedings of Machine Learning Research*, pages 582–597. PMLR, 2018.

[41] Jess Whittlestone, Kai Arulkumaran, and Matthew Crosby. The Societal Implications of Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, 70:1003–1030, 2021.

[42] Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-Gradient Reinforcement Learning. *Advances in Neural Information Processing Systems*, 31:2396–2407, 2018.

[43] Jiachen Yang, Igor Borovikov, and Hongyuan Zha. Hierarchical Cooperative Multi-Agent Reinforcement Learning with Skill Discovery. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1566–1574, 2020.

[44] Y Yang, R Luo, M Li, M Zhou, W Zhang, and J Wang. Mean Field Multi-Agent Reinforcement Learning. In *35th International Conference on Machine Learning, ICML 2018*, volume 80, pages 5571–5580. PMLR, 2018.

[45] Dayong Ye, Minjie Zhang, and Yun Yang. A Multi-Agent Framework for Packet Routing in Wireless Sensor Networks. *sensors*, 15(5):10026–10047, 2015.

[46] Lianmin Zheng, Jiacheng Yang, Han Cai, Ming Zhou, Weinan Zhang, Jun Wang, and Yong Yu. MAgent: A Many-Agent Reinforcement Learning Platform for Artificial Collective Intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4 and 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 8.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 2 and 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] See complete proof below Theorem 1 in Section 4.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and README are provided in the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5 and Appendix A.1.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All plots in Section 6 display the 95% confidence intervals of 30 independent random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.2 and Appendix A.1.3. We provide the average runtime of each training run of the results in Section 6 and Appendix A.3 including hyperparameter tuning to estimate the total amount of compute.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A] We (re)implemented all algorithms and environments by ourselves in order to fit our MARL interface. However, we cited all the original works.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A] Does not apply.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] Does not apply. Only simulated data from the domains described in Section 5 was used.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] Does not apply. Only simulated data from the domains described in Section 5 was used.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Does not apply.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Does not apply.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Does not apply.