# Efficiently Quantifying Individual Agent Importance in Cooperative MARL

**Omayma Mahjoub**[*, 1], **Ruan de Kock**[*,1], **Siddarth Singh**[*,1], **Wiem Khlifi**[1, 2] **Abidine Vall**[3], **Kale-ab Tessera**[4] **Rihab Gorsane**[1], **Arnu Pretorius**[1]

[1]InstaDeep Ltd
[2]National School of Computer Science, Tunisia
[3]National School of Engineering of Tunis
[4]University of Edinburgh
[*]Equal Contribution

## Abstract

Measuring the contribution of individual agents is challenging in cooperative multi-agent reinforcement learning (MARL). In cooperative MARL, team performance is typically inferred from a single shared global reward. Arguably, among the best current approaches to effectively measure individual agent contributions is to use Shapley values. However, calculating these values is expensive as the computational complexity grows exponentially with respect to the number of agents. In this paper, we adapt difference rewards into an efficient method for quantifying the contribution of individual agents, referred to as Agent Importance, offering a linear computational complexity relative to the number of agents. We show empirically that the computed values are strongly correlated with the true Shapley values, as well as the true underlying individual agent rewards, used as the ground truth in environments where these are available. We demonstrate how Agent Importance can be used to help study MARL systems by diagnosing algorithmic failures discovered in prior MARL benchmarking work. Our analysis illustrates Agent Importance as a valuable explainability component for future MARL benchmarks.

## 1 Introduction

In recent years, multi-agent reinforcement learning (MARL) has achieved significant progress, with agents being able to perform similar or better than human players and develop complex coordinated strategies in difficult games such as *Starcraft* (Samvelyan et al. 2019; Vinyals et al. 2019), *Hanabi* (Foerster et al. 2019; Bard et al. 2020; Hu and Foerster 2021; Du et al. 2021) and *Diplomacy* (Bakhtin et al. 2022). Furthermore, MARL has also shown promising results in solving real-world problems such as resource allocation, management and sharing, network routing, and traffic signal controls (Vidhate and Kulkarni 2017; Brittain and Wei 2019; Nasir and Guo 2019; Spatharis et al. 2019; Liu et al. 2020; Zhao, Liu, and Cheng 2020; Pretorius et al. 2020; Gu et al. 2021). These real-world settings are naturally formulated as cooperative MARL systems, where agents need to coordinate to optimise the same global reward.

One of the critical challenges in cooperative MARL is multi-agent credit assignment (Chang, Ho, and Kaelbling 2003). Since agents typically receive a global reward for their joint actions, this makes determining individual agent contributions challenging. This need for correct attribution becomes especially important as more autonomous systems are deployed in the real world. The inherent complexity of these MARL systems impedes our understanding of decision-making processes and the motivations behind actions, hindering progress in this field. Improved credit assignment could play a vital role in comprehending agent behaviour and system-level decision-making, aiding in accountability, trust, fairness, and facilitating the detection of potential issues such as coordination failures, or unethical behaviour.

Credit assignment can be considered from a core algorithmic perspective, where components of reinforcement learning (RL) algorithms, such as the value function, are adapted to better decouple the impact of the actions of individual agents. Methods such as VDN (Sunehag et al. 2017a), COMA (Foerster et al. 2018), and QMIX (Rashid et al. 2018a) fall into this domain. However, since these algorithms are trained end-to-end through the use of function approximators, explainability is difficult, i.e. it is challenging to correlate specific agent actions to reward outcomes over time. Furthermore, since these notions of agent impact are part of the RL algorithms themselves, it is not easy to transfer these between different algorithms.

Accurate credit assignment within a team of agents can also be seen as a form of explainable AI (XAI). XAI consists of machine learning (ML) techniques that are used to provide insights into the workings of models (Arrieta et al. 2020). It has been used across various domains in ML, and more recently in single-agent RL[1] (Glanois et al. 2021) and multi-agent systems (Heuillet, Couthouis, and Díaz-Rodríguez 2022). Following from (Arrieta et al. 2020; Glanois et al. 2021), we use the notion of explainability to refer to any external post-hoc methodology that is used to gain insights into a trained model. These techniques have the notable advantage of being able to be used across algorithms, often irrespective of their design or formulation.

Efforts to enhance explainability in RL have resulted in the development of various techniques (Juozapaitis et al.



[1]In this paper, we use the term "RL" to exclusively refer to *single-agent* RL, as opposed to RL as a field of study, of which MARL is a subfield.

2019; Madumal et al. 2020; Puiutta and Veith 2020; Glanois et al. 2021; Heuillet, Couthouis, and Díaz-Rodríguez 2021; Vouros 2022; Dazeley, Vamplew, and Cruz 2023). In contrast, MARL lacks dedicated explainability tools, with only a limited number of works addressing this topic (Kraus et al. 2019; Boggess, Kraus, and Feng 2022; Heuillet, Couthouis, and Díaz-Rodríguez 2022). One notable approach involves leveraging the Shapley value (Shapley 1953), a metric derived from game theory, and adapting it to MARL to quantify agent contributions to the global reward (Heuillet, Couthouis, and Díaz-Rodríguez 2022). Although Shapley values have shown promise in MARL explainability, calculating these values is expensive as the computational complexity grows exponentially with respect to the number of agents.

In this paper, we highlight the need for employing explainable tools to help quantify credit assignment in cooperative MARL systems. We show that an averaged calculation of the difference reward (Wolpert and Tumer 2001) across evaluation episodes, can be used as an effective metric for measuring an agent's contribution, which we refer to as the *Agent Importance*. Unlike Shapley values, the Agent Importance has a linear computational complexity (w.r.t. the number of agents) making it more efficient to compute. Through empirical analysis, we demonstrate a strong correlation between the Agent Importance values and the true Shapley values, while also empirically validating the scalability and computational advantage of this approach.

To showcase the practical use of Agent Importance, we revisit a previous benchmark in cooperative MARL (Papoudakis et al. 2021) and follow the standardised evaluation guideline proposed by (Gorsane et al. 2022) to reproduce key results from this benchmark under a sound protocol. We then proceed by applying Agent Importance to specific scenarios of interest as highlighted by the authors of this benchmark. This includes investigating: (1) why Multi-Agent Advantage Actor-Critic (MAA2C) (Mnih et al. 2016a; Papoudakis et al. 2021) outperforms Multi-Agent Proximal Policy Optimisation (MAPPO) (Yu et al. 2022) in the Level-Based Foraging (LBF) environment [2] (Albrecht and Ramamoorthy 2015; Albrecht and Stone 2019; Christianos, Schäfer, and Albrecht 2020); and (2) why parameter sharing between agents leads to improved performance (3) analyse agents' behaviour in case of heterogeneous settings. Using agent importance, we uncover that for (1) MAA2C achieves a more equal contribution among agents when compared to MAPPO, i.e. agents have a more similar importance to the overall team and therefore have a higher degree of cooperation; and that for (2) architectures without parameter sharing exhibit a higher variance in agent importance, leading to credit assignment issues and lower performance compared to architectures with parameter sharing. The source code to reproduce our analysis and compute the agent importance, as well as our raw experiment data is publicly available [3].

[2] A somewhat surprising result since MAPPO uses importance sampling for off-policy correction and is expected to perform at least as well as MAA2C as it incorporates a clipping function based on importance sampling allowing data retraining without divergent policies.

[3] Data and code are accessible at the following links:

Data- https://sites.google.com/view/agent-importance/home
Code- https://tinyurl.com/ycx47jz6

## 2 Related Work

**Explainability in RL** With the surging popularity of Deep RL, which relies on black-box deep neural networks, there has been an increase in literature that attempts to enable human understanding of complex, intelligent RL systems (Juozapaitis et al. 2019; Madumal et al. 2020; Puiutta and Veith 2020; Glanois et al. 2021; Heuillet, Couthouis, and Díaz-Rodríguez 2021; Vouros 2022; Dazeley, Vamplew, and Cruz 2023). Additionally, frameworks like ShinRL (Kitamura and Yonetani 2021) and environment suites like bsuite (Osband et al. 2019) offer comprehensive debugging tools including state and action space visualizations and reward distributions, and carefully crafted environments for behavioural analysis in RL.

**Explainability in MARL** In contrast to explainable RL, there has been a limited amount of work focusing on explainability in MARL (Kraus et al. 2019; Boggess, Kraus, and Feng 2022; Heuillet, Couthouis, and Díaz-Rodríguez 2022). Specifically, we are interested in explainability in the context of cooperative MARL with a shared, global reward and the aim is to effectively quantify credit assignment.

The challenges associated with measuring credit assignment in MARL have motivated researchers to explore the use of the **Shapley value** (Shapley 1953). Originating from game theory, the Shapley value addresses the issue of payoff distribution within a "grand coalition" (i.e. a cooperative game) and quantifies the contribution of each coalition member toward completing a task. Specifically, consider a cooperative game $\Gamma = (\mathcal{N}, v)$, where $\mathcal{N}$ is a set of all players and $v$ is the payoff function used to measure the "profits" earned by a given coalition (or subset) $\mathcal{C} \subseteq \mathcal{N} \backslash \{i\}$, such that the marginal contribution of player $i$ is given by $\phi_i(\mathcal{C}) = v(\mathcal{C} \cup \{i\}) - v(\mathcal{C})$. The Shapley value of each player $i$ can then be computed as:

$$S_i(\Gamma) = \sum_{\mathcal{C} \subseteq \mathcal{N} \backslash \{i\}} \frac{|\mathcal{C}|!(|\mathcal{N}| - |\mathcal{C}| - 1)!}{|\mathcal{N}|!} \cdot \phi_i(\mathcal{C}). \quad (1)$$

Calculating Shapley values in the context of MARL presents two specific challenges: (1) it requires computing $2^{n-1}$ possible coalitions of a potential $n(2^{n-1})$ coalitions (with $|\mathcal{N}| = n$) which is computationally prohibitive and (2) it strictly requires the use of a simulator where agents can be removed from the coalition and the payoff of the same states can be evaluated for each coalition.

Despite its limitations, the Shapley value is able to alleviate the issue of credit assignment and help towards understanding individual agent contributions in MARL. As a result, numerous efforts have been undertaken to incorporate it as a component of an algorithm (Wang et al. 2020; Yang et al. 2020a; Han et al. 2022; Wang et al. 2022). However, in this work, we focus on the Shapley value as an explainability metric. One such approach is introduced in (Heuillet, Couthouis, and Díaz-Rodríguez 2022), where the authors utilise a Monte Carlo approximation of the Shapley value to estimate the contribution of each agent in a system, which we refer to here as **MC-Shapley**. This approximate Shapley value is computed

as:

$$\hat{S}_i^{MC}(\Gamma) = \frac{1}{M}\sum_{m=1}^{M}(r_{\mathcal{C}_m\cup\{i\}} - r_{\mathcal{C}_m}) \approx S_i(\Gamma), \quad (2)$$

where M is the number of samples (episodes), $C_m$ is a randomly sampled coalition out of all possible coalitions excluding agent $i$, and $r_{\mathcal{C}_m\cup\{i\}}$ and $r_{\mathcal{C}_m}$ are the episode returns obtained with and without agent $i$ included in the coalition.

In essence, Heuillet, Couthouis, and Díaz-Rodríguez (2022) attempts to address the second limitation of the Shapley value, which involves removing agents from the environment. They propose three strategies for proxies of agent removal while computing the return $r_{\mathcal{C}_m}$. The first hypothesis is to provide the agent $i$ with a no-op (no-operation) action, the second is to assign the agent $i$ with a random action, and the third is to replace the action of agent $i$ with a randomly selected agent's action from the current coalition $C_m$. The paper's findings indicate that using the no-op approach yields the most accurate approximation of the true Shapley value. A primary limitation of this work is the dependence on a significant number of sampled coalitions, with each sample corresponding to a single episode. This characteristic has a notable impact on training speed, especially if the proposed approach is employed as an online metric for detecting the evolution of agents' contributions during system training.

**Difference Rewards.** Of central relevance to this work is difference rewards (Wolpert and Tumer 2001; Agogino and Tumer 2004, 2008; Devlin et al. 2014) which presents a method for estimating credit assignment within a system. It can be written as $D_i(z) = G(z) - G(z_{-i})$ where $D_i(z)$ is the difference reward for agent $i$, $z$ is a state or state-action pair depending on the application, $G(z)$ is the performance of the global system and $G(z_{-i})$ is the performance of a theoretical system that omits agent $i$. Any action taken that increases the difference reward $D_i(z)$ also increases $G(z)$ but will have a higher impact on the (typically unknown or hypothetical) individual reward for each agent compared to the global reward. It is from this property that we may determine the relative impact of each agent in a system.

## 3 Agent Importance

We compute the Agent Importance as an average of difference rewards and use it as an efficient estimate of the Shapley value. To ensure accuracy in our estimation, we emphasize the importance of utilizing an adequate number of samples. This is reminiscent of the MC-Shapley approach which uses Monte Carlo approximation over entire episodes (Heuillet, Couthouis, and Díaz-Rodríguez 2022). However, in this work, we show that such an approach to estimation is not necessary and instead, we compute difference returns over samples collected *per step*, rather than per episode, without the need to resample coalitions. We simply compute the difference reward for each agent at each timestep during evaluation and aggregate over all evaluation timesteps. This approach greatly improves the sample efficiency in estimation during online evaluation. Concretely, the **Agent Importance** is given by

$$\hat{S}_i^{AI}(\Gamma) = \frac{1}{T}\sum_{t=1}^{T} r^t - r^t_{-i}, \quad (3)$$

where $T$ is the number of timesteps in a full evaluation interval, $r^t$ is the team reward (i.e. the reward of the grand coalition), at timestep $t$ and $r^t_{-i}$ is the team reward when agent $i$ performs a no-op action.

Applying Equation 3 poses a technical challenge as it requires comparing rewards between agents based on the same exact environment state at a given timestep. In MARL, most simulators are not easily resettable and/or stateless, which makes measuring one reward and undoing that step and then measuring a second reward difficult [4]. To overcome this limitation, we adopt a simple solution outlined in Algorithm 1, where we create a copy of the environment for each agent to be able to compute the Agent Importance.

Algorithm 1: Per timestep difference reward contribution in Agent Importance

**Require:** $t$: evaluation timestep, $marginal\_contribution$: dictionary

$env\_copies \leftarrow$ deepcopy$(env, len(agents))$
$r^t \leftarrow$ env.step$(selected_actions)$
**for** $i = 0$ **to** $len(agents)$ **do**
  $actions\_with\_no\_op \leftarrow$ disable_actions$(selected_actions, i)$
  $r^t_{-i} \leftarrow$ env_copies$[i]$.step$(actions_with_no_op)$
  add_to_dict$(marginal_contribution, i, (r^t - r^t_{-i}))$
**end for**

## 4 Case Study: using Agent Importance to analyse a prior benchmark

Our case study setup is based on the work of (Papoudakis et al. 2021), which made a comparative benchmark of cooperative MARL algorithms. The study conducts evaluations and comparisons of multiple categories of MARL algorithms, covering Q-learning, and policy gradient (PG) methods, across two paradigms: independent learners (ILs), and centralised training with decentralised execution (CTDE). The findings of this study align with those of (Gorsane et al. 2022), concluding that current MARL algorithms are most performant on the popular Multi-Particle Environment (MPE) (Lowe et al. 2017) and Starcraft Multi-Agent Challenge (SMAC) (Samvelyan et al. 2019) environments–with most algorithms achieving comparable performance, in some cases seemingly to the point of overfitting. Consequently, our main analysis focuses on the remaining two environments from this benchmark: LBF, and RWARE.

**Environments.** The Multi-Robot Warehouse (RWARE) (Christianos, Schäfer, and Albrecht 2020; Papoudakis et al. 2021) is a multi-agent environment that is designed to represent a simplified setting where robots move goods around

[4] We however do note, that this could easily be achieved with simulators written using pure functions in JAX (Freeman et al. 2021; Lange 2022; Bonnet et al. 2023).

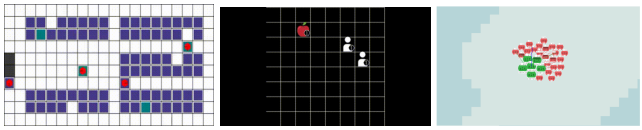


Figure 1: **Left**: Multi-Robot Warehouse (RWARE). **Middle**: Level-Based Foraging (LBF). **Right**: SMAClite

a warehouse. The environment requires agents (circles) to move requested shelves (colored squares) to the goal post (dark squares) and back to an empty square as illustrated at the top of Figure 1. Tasks are partially observable with a very sparse reward signal as agents have a limited field of view and are rewarded only upon a successful delivery.

Level-Based Foraging (LBF) (Albrecht and Ramamoorthy 2015; Albrecht and Stone 2019; Christianos, Schäfer, and Albrecht 2020) is a mixed cooperative-competitive game with a focus on inter-agent coordination illustrated at the bottom of Figure 1. Agents are assigned different levels and navigate a grid world where the goal is to consume food by cooperating with other agents if required. Agents can only consume food if the combined level of the agents adjacent to a given item of food exceeds the level of the food item. Agents are awarded points equal to the level of the collected food divided by their level. LBF has a particularly high level of stochasticity since the spawning position and level assigned to each agent and food are all randomly reset at the start of each episode.

In the original benchmarking work by (Papoudakis et al. 2021), the authors used the popular Starcraft Multi-Agent Challenge (SMAC) (Samvelyan et al. 2019) environment. In our case study, we instead use SMAClite (Michalski, Christianos, and Albrecht 2023), an environment designed to replicate SMAC faithfully, in Python. An illustration of SMAClite is given in Figure 1. SMAClite has similar system dynamics to SMAC but does not rely on the StarCraft 2 video game engine as a backend. Due to this SMAClite requires significantly less RAM making it more suitable for utilising parallel processing. This also means it can be used in conjunction with Python methods like `copy` which makes contribution analysis methods like simpler to implement.

**Algorithms.** As in the original benchmarking setup of (Papoudakis et al. 2021), we use the exact same collection of algorithms for our case study. Specifically, we use the value-based algorithms Independent Q-Learning (IQL) (Tan 1997), Value-Decomposition Network (VDN) (Sunehag et al. 2017a), and QMIX (Rashid et al. 2018a), alongside two policy-gradient (PG) algorithms, namely Multi-Agent Proximal Policy Optimisation (MAPPO) (Yu et al. 2022) and Multi-Agent Advantage Actor-Critic (MAA2C) (Foerster et al. 2018). To investigate the influence of parameter sharing, we conduct experiments with both parameter-sharing and non-parameter-sharing architectures. Further details about the algorithms can be found in the Appendix section A.

**Evaluation Protocol.** We follow the protocol outlined by (Gorsane et al. 2022), and apply the evaluation tools from (Agarwal et al. 2022) in the MARL setting as advocated in the protocol. We evaluate agents at 201 equally spaced evaluation intervals for 32 episodes each during training. Following from the recommendations of (Papoudakis et al. 2021) we train off-policy algorithms for a total of 2M timesteps and on-policy algorithms for a total of 20M timesteps summed across all parallel workers. This implies that evaluation occurs at fixed intervals of either 10k or 100k total environment steps for off- and on-policy algorithms respectively. For all our experiments, we use the EPyMARL framework (Papoudakis et al. 2021) which is opensourced under the Apache 2.0 licence. This is to ensure we are evaluating all algorithms on the same tasks, using the same codebase as was done by (Papoudakis et al. 2021) for maximal reproducibility. Furthermore, it allows us to use identical hyperparameters as used in their work, which are available in the Appendix section A. All results that are presented are aggregated over 10 independent experiment trials. In cases where aggregations are done over multiple tasks within an environment, as opposed to an individual task (e.g. for computing performance profiles), the interquartile mean is reported along with 95% stratified bootstrap confidence intervals. For all plots except for sample efficiency curves, the absolute metric (Colas, Sigaud, and Oudeyer 2018; Gorsane et al. 2022) for a given metric is computed. This metric is the average metric value of the best-performing policy found during training rolled out for 10 times the number of evaluation episodes.

**Computational resources**. All experiments were run on an internal cluster using either AMD EPYC 7452 or AMD EPYC 7742 CPUs. Each independent experiment run was assigned 5 CPUs and 5GB of RAM with the exception of the scalability experiments which were exclusively run using AMD EPYC 7742 CPUs and either 5, 15, 30, or 200 GB of memory depending on the number of agents and subsequently the number of environment copies that were required.

## 5 Results

We demonstrate the validity of Agent Importance by considering its correlation to the true Shapley value, its computational scalability and its reliability in quantifying individual agent contributions. We then proceed to illustrate how Agent Importance may be used as an explainability tool.

### Validating Agent Importance

**Correlation between Agent Importance and the Shapley value.** We note that the Agent Importance metric is not mathematically equivalent to the Shapley value. It focuses on the grand coalition rather than all possible agent coalitions. However, through empirical study, we argue that Agent Importance is sufficient for capturing agents' contributions in the context of cooperative MARL.

To validate our assertion, we conduct experiments on both LBF and RWARE to empirically assess the correlation between Agent Importance and the Shapley value. We generate a heatmap that describes the correlation between the metrics for the VDN algorithm. Furthermore, we assess the ability of a metric to maintain the relative agent rankings according to each agent's individual rewards (which are not seen by the agents). If a metric gives the same ranking to agents, we count this as a positive result–implying that a higher-ranking match is better. While only results on VDN are displayed,

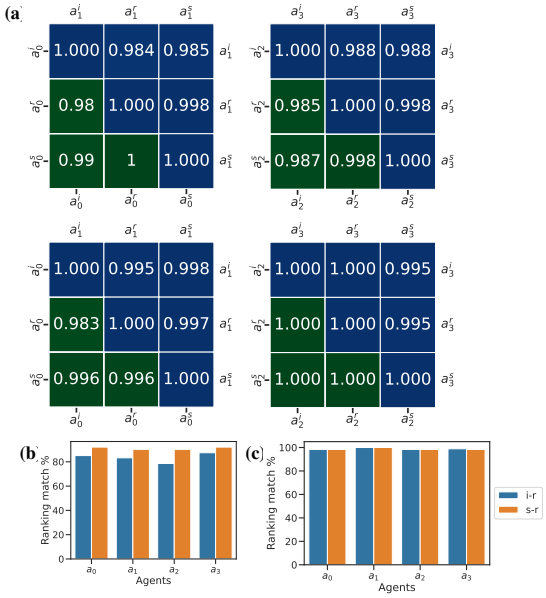



Figure 2: *Correlation analysis for agents* $\{a_0, a_1, a_2, a_3\}$*, for each metric: Agent Importance* $i$*, Shapley Value* $s$*, and Individual Reward* $r$ *using the VDN algorithm.* **(a)** Heatmap of Correlations among Metrics. **TOP:** LBF 15x15-4p-5f. **BOTTOM:** RWARE small-4ag. **(b)** Matching Rankings Comparison on LBF 15x15-4p-5f. **(c)** Matching Rankings Comparison on RWARE small-4ag. The legend refers to which metric is being compared to the individual agent rewards.

the trend is consistent for all algorithms across various tasks. Further results to this end are given in the Appendix section D.

Figure 2 (a) shows that there exists a strong correlation between the Agent Importance, the Shapley value and the individual agent reward as calculated by the Pearson correlation coefficient. This indicates the effectiveness of both the Shapley value and Agent Importance in assessing agents' contributions, making them valuable substitutes for individual agent rewards in environments where such rewards are unavailable. Notably, Agent Importance showcases a promising ability to effectively replace both the Shapley value and individual rewards. While the Shapley value may provide greater consistency in ranking information when compared to the Agent Importance (as illustrated in Figures 2 (b,c)) where the frequency of ranking agreement between the individual reward and the contribution estimators is illustrated, it is important to note that Agent Importance is highly correlated with the individual reward and shows a minimal rate of non-matched rankings.

**Scalability of Agent Importance.** In order to validate the computational feasibility of the simplified Agent Importance against the full Shapley value we record the run time of both approaches on LBF tasks with $2, 4, 10, 20$, and $50$ agents. We run the algorithm without any training and compute the number of seconds it takes for agents to take a single environment step while computing each metric. The reported results here

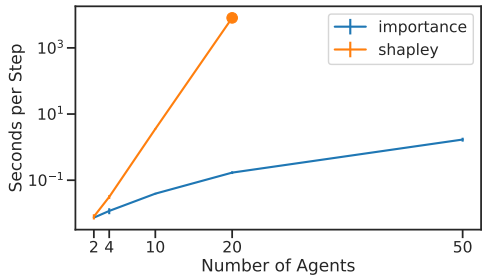



Figure 3: *Computational cost of computing the agent importance and the Shapley value.*

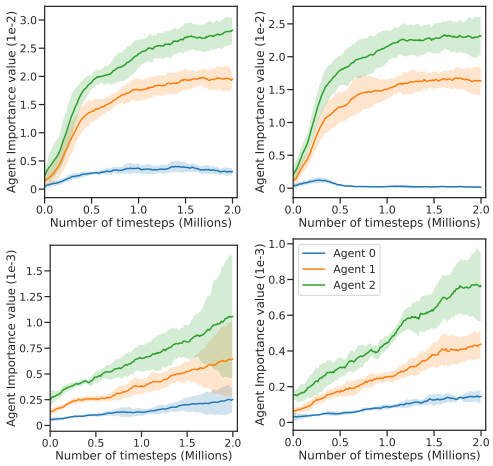



Figure 4: *Agent importance scores on the deterministic LBF scenario for MAA2C, MAPPO, VDN and QMIX.* Agents $0$, $1$ and $2$ are assigned fixed levels of $1$, $2$ and $3$ respectively– implying that their contributions should be weighted accordingly.

are the mean and standard deviation over 3 independent runs. The Shapley value became prohibitively slow as the agent number was increased and required approximately 2 hours to measure a single step within the environment with 20 agents. Nonetheless, Figure 3 clearly illustrates how the Agent Importance is significantly more computationally efficient than the Shapley value.

**Reliability of Agent Importance.** In order to validate the ability of the Agent Importance to effectively untangle agent contributions from a shared team reward, we create a deterministic version of LBF where agent levels are always fixed to be $1, 2$, and $3$ respectively, and the maximum level of each food is a random value between 1 and 6. Since agent 2 is assigned a fixed greater level than its counterparts we should expect it to contribute the most to the team return. Figure 4 illustrates the ability of Agent Importance to uncover the correct ordering and approximate level of contribution among agents towards the overall team goal.

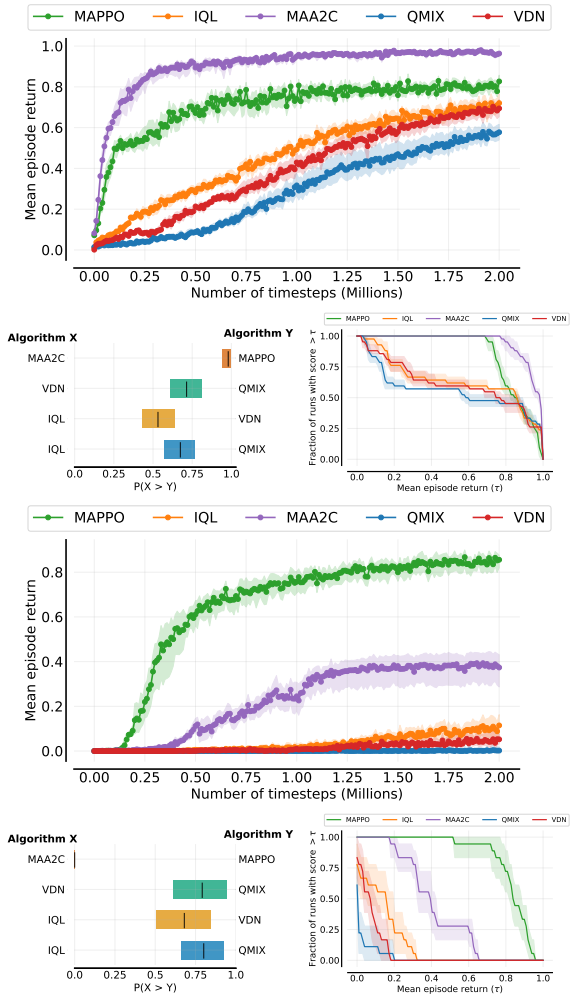



Figure 5: *Algorithm performance on LBF and RWARE including probability of improvement, performance profiles and sample efficiency curves*. **Top two rows:** Performance of algorithms on 7 LBF tasks. **Bottom two rows:** Performance of all algorithms on 3 RWARE tasks.

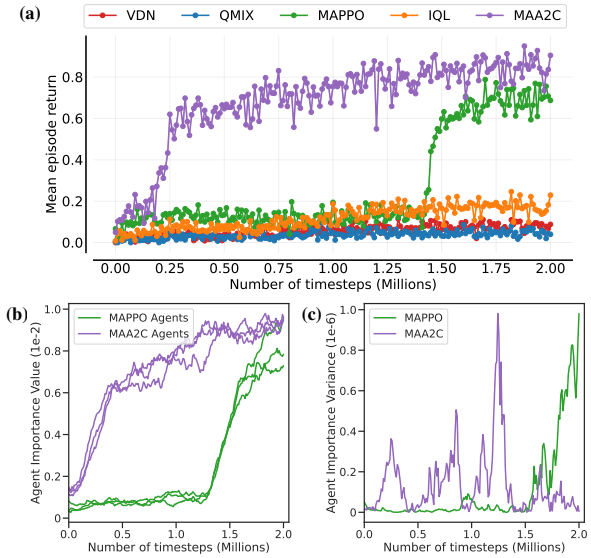



Figure 6: *MAA2C outperforms MAPPO on the LBF 15x15-3p-5f task.* **(a)** Sample efficiency curves (one seed). **(b)** Agent importance for all agents associated with a given algorithm. **(c)** Variance of Agent Importance. MAA2C has a lower variance in agent importance at convergence.

## Applications of Agent Importance

We replicated the experiments performed by (Papoudakis et al. 2021), obtaining similar results. However, our work adds value by following a strict protocol (Gorsane et al. 2022) which includes additional evaluation measurements such as examining the probability of improvement and providing performance profiles (Agarwal et al. 2022), as shown in Figure 5. Additional plots and tabular results for different scenarios and the performance of the algorithms without parameter sharing are included in the Appendix along with more detailed performance plots for SMAClite in Appendix section C.

**MAA2C vs MAPPO.** Empirical results in RL consistently demonstrate that PPO tends to outperform A2C (Heess et al. 2017; Schulman et al. 2017; Henderson, Romoff, and Pineau 2018). This trend naturally leads to the question of whether a similar pattern is observed in the multi-agent setting, i.e. between MAPPO and MAA2C. However, when examining the results in Figure 5, a conflicting observation arises. In the case of RWARE, we observe the expected behaviour with the probability of improvement aligning with our initial expectations. However, in the case of LBF, the opposite occurs as MAA2C outperforms MAPPO, presenting an unexpected outcome. Figure 6 (b) highlights a possible reason. By tracking Agent Importance, we may attribute this outcome to a narrowing in the spread of importance values between MAA2C agents at convergence, as compared to MAPPO agents. The assumption of lower variance in Agent Importance leading to improved performance in LBF is due to the stochasticity of the environment. It is reasonable to expect that an algorithm performing well in this environment should have the capability to adapt to the variability in agent and food levels across episodes. From the narrower spread in Agent Importance values in MAA2C we can see it has learnt treat all agents as equally important.For additional findings on RWARE see section B in the supplementary material.

**Parameter sharing vs non-parameter sharing.** Consistent with the findings of (Papoudakis et al. 2021), our experiments demonstrate that algorithms utilizing parameter sharing outperform those without it. As mentioned in the benchmark paper, this outcome is expected as parameter sharing enhances sample efficiency. Additionally, parameter sharing enhances the sharing of learned information across the system. The Agent Importance analysis for IQL, QMIX, and VDN provides clear evidence of the impact of parameter-sharing architectures, as illustrated in Figure 7. It is apparent that in the absence of parameter sharing, the agents contribute to varying degrees, leading to an uneven distribution of importance. And as mentioned previously, given LBF's

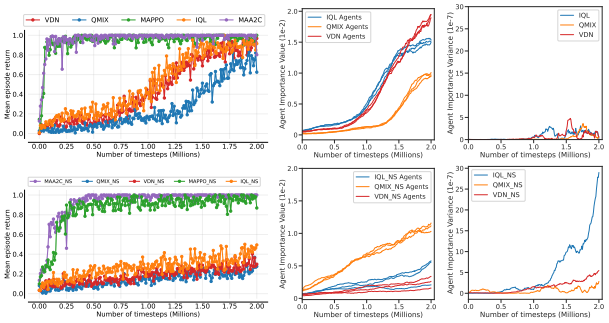



Figure 7: *Comparison of performance with and without parameter sharing on the LBF 10x10-3p-3f task for one seed including the sample efficiency, Agent Importance, and Agent Importance variance.* **Top row:** Performance with policy parameter sharing. **Bottom row:** Performance without policy parameter sharing. With parameter sharing the agent importance is more evenly distributed.

characteristics, requiring a high level of coordination in the presence of significant stochasticity, all agents should be expected (on average) to contribute equally. However, in the non-parameter sharing cases, especially for IQL and VDN, we observe that a small number of the agents dominate the contributions, resulting in lower performance compared to when parameter sharing is utilised.

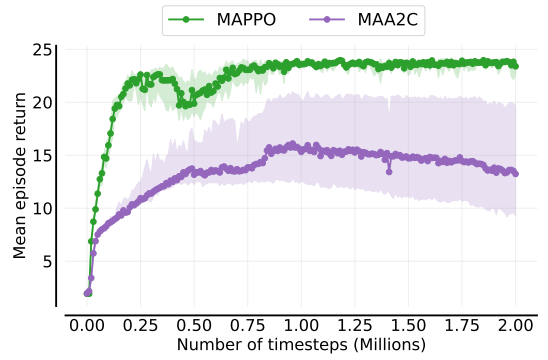
**Heterogeneous Agents.** In both LBF and RWARE the importance of each agent and the total reward are highly correlated as all agents have similar capabilities. In the heterogeneous setting of MMM2, rather than converging to similar importance levels over time, agents will instead converge to clear groups of importance levels as seen in figures 8b and 8c. Furthermore, note that agents of the same type can still fall into different levels of importance which is consistent with role decomposition analysis in ROMA (Yang et al. 2020b). As shown by (Yang et al. 2020b), the optimal policy in MMM2 requires a subset of marine agents to die early in the episode, who then cannot contribute to the team reward remaining timesteps, whereas a smaller number of marines survive until the end. This is clearly seen in figure 8c for MAPPO. In the case of MAA2C in figure 8b we can see that although clear clusters have formed, it has not learned to assign the correct importance to a subgroup of marines that are required to optimally solve the environment [5]

## 6 Discussion

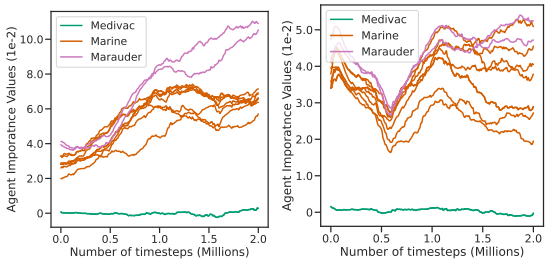
In this work, we illustrate that Agent Importance is an efficient and reliable measure for agent contributions towards the team reward in cooperative MARL. Aside from only quantifying the agent contributions we have also shown how the metric may be used as an explainability tool for uncovering failure modes in existing MARL results.

**Limitations.** Although Agent Importance is useful, using

[5]An optimal policy for MMM2 can be found in a video by the original SMAC authors in https://www.youtube.com/watch?v=VZ7zmQ_obZ0 and additional information in appendix C.




(a) MMM2 sample efficiency




(b) MAA2C (c) MAPPO

Figure 8: *Comparison between MAA2C and MAPPO in the MMM2 scenario from SMAClite.*8a: Mean episode returns. 8b: Agent Importance scores MAA2C. 8c: Agent Importance scores MAPPO.

simulators that allow for an agent's removal during runtime would be highly advantageous. Solely relying on no-op actions could still impact the coalition reward by obstructing other agents' presence and movement in their observations. Unfortunately, agent removal is uncommon in most simulators and some simulators also do not offer the option for a no-op action. Additionally, while popular MARL research environments are fairly low resource, creating multiple parallel instances of the environment during the Agent Importance calculation, makes using more resource-heavy simulators prohibitive from a memory perspective. However, with the growing popularity of the JAX framework, more stateless environments are becoming available where the parallel environments can be replaced with direct access to the environment state (Freeman et al. 2021; Lange 2022; Bonnet et al. 2023).

**Future Work.** It would be useful to investigate the rankings calculated by *agent importance* for simulators which do not have a no-op action. We could consider using random actions or the random actions of specific agents as a proxy for the no-op action or make use of function approximators to learn minimal impact actions for the marginalised agents.

## A Experimental details

### Environments

To ensure our experimentation setup is clear and easily reproducible, we make use of the same environment naming

conventions used in (Papoudakis et al. 2021). In this section, we provide an overview of the naming conventions employed. Primarily we break down how the naming conventions of each environment correspond to the features of each scenario in the Level-Based Foraging (LBF) and Multi-Robot Warehouse (RWARE) environments.

Figure 9 illustrates a collection of ten scenarios, each corresponding to a specific task in the LBF and RWARE environments. The LBF scenarios are described in detail in Section A, while the RWARE scenarios are explained in Section A.

**Level Based Foraging Naming Convention.** The scenarios in the LBF environment are named according to the following convention:

$$Foraging < obs > - < x\_size > x < y\_size > - < n\_agents > p - < food > f < force\_c > - v1$$

Each field in the naming convention has specific options:

- ¡obs¿: Denotes agent level of partial observability for all agents. If no value is given the agents can see as far as the grid is wide.
- $< x\_size >$: Size of the grid along the horizontal axis.
- $< y\_size >$: Size of the grid along the vertical axis.
- $< n\_agents >$: Number of agents in the environment.
- $< food >$: Number of food items in the environment. This is the total number of food that can spawn per episode.
- $< force\_c >$: Optional field indicating a forced cooperative task. It can be empty or set to "-coop" mode. In this mode, the levels of all the food items are intentionally set equal to the sum of the levels of all the agents involved. This implies that the successful acquisition of a food item requires a high degree of cooperation between the agents since no agent will be able to collect a food item by itself.

  As an example, an environment named "Foraging-2s-8x8-2p-2f-coop" has a sight range of "2s" implying that agents can view a 5x5 grid centred on themselves, a grid with horizontal and vertical size 8, contains 2 agents, 2 food objects and is set to a cooperative mode.

**Additional Level Based Foraging Scenarios** To gain insights into the interplay between individual agent levels, their impact on team performance, and individual contribution (Agent Importance value), we introduce additional LBF environments. These additional environments serve as a testing ground to study the reliability and scalability of the Agent Importance metric.

With the addition of these new scenarios, we focus on two distinct test features. Firstly, we assess the reliability of agent importance by making tasks that will always have three agents with levels of 1, 2, and 3, respectively. This allows us to compare the agent importance values to the predetermined agent levels to see whether they correspond. We also introduce two versions of the scenarios with fixed agent levels; one where the levels of food items are uniformly random values between 1 and 6 and another where the food levels are always 3. Secondly, we assess the scalability of agent importance w.r.t the number of agents in the setting. To do this we enlarge the grid size of the LBF scenarios to accommodate more agents up to a maximum tested number of 50.

**Used scenarios.** Our research experiments were carried out on a varied set of scenarios, and in all cases, agent positions and food positions as well as agent levels and food levels are randomly generated at each new environment episode.

- **Main scenarios**
  - Figure 9(a) **Foraging-2s-8x8-2p-2f-coop**: 8x8 grid, partial observability with sight=2, 2 agents, 2 food items, cooperative mode.
  - Figure 9(b) **Foraging-8x8-2p-2f-coop**: 8x8 grid, full observability, 2 agents, 2 food items, cooperative mode.
  - Figure 9(c) **Foraging-2s-10x10-3p-3f**: 10x10 grid, partial observability, 3 agents, 3 food items.
  - Figure 9(d) **Foraging-10x10-3p-3f**: 10x10 grid, full observability, 3 agents, 3 food items.
  - Figure 9(e) **Foraging-15x15-3p-5f**: 15x15 grid, full observability, 3 agents, 5 food items.
  - Figure 9(f) **Foraging-15x15-4p-3f: 15x15 grid**, full observability, 4 agents, 3 food items.
  - Figure 9(g) **Foraging-15x15-4p-5f: 15x15 grid**, full observability, 4 agents, 5 food items.
- **Reliability Scenarios**
  - **Foraging-15x15-3p-3f-det**: 15x15 grid, full observability, 3 agents, 3 food items. The food levels are fixed to always be 3.
  - **Foraging-15x15-3p-3f-det-max-food-sum**: 15x5 grid, full observability, 3 agents, 3 food items. The food levels are uniformly random values between 1 and 6.
- **Scalability Scenarios**
  - **Foraging-5x5-2p-2f**: 5x5 grid, full observability, 2 agents, 2 food items.
  - **Foraging-10x10-4p-4f**: 10x10 grid, full observability, 4 agents, 4 food items.
  - **Foraging-15x15-10p-10f**: 15x15 grid, full observability, 10 agents, 10 food items.
  - **Foraging-20x20-20p-20f**: 20x20 grid, full observability, 20 agents, 20 food items.
  - **Foraging-25x25-50p-50f**: 25x25 grid, full observability, 50 agents, 50 food items.

**Multi-Robot Warehouse Naming Convention.** The scenarios in the RWARE environment are named according to the following convention:

$$rware - < size > - < num\_agents > ag < diff > - v1$$

Each field in the naming convention has specific options:

- $< size >$: Represents the size of the Warehouse (e.g., "tiny", "small", "medium", "large").
- $< num\_agents >$: Indicates the number of agents (1-20).
- $< diff >$: Optional field indicating the difficulty of the task (default: N requests for each of the N agents).

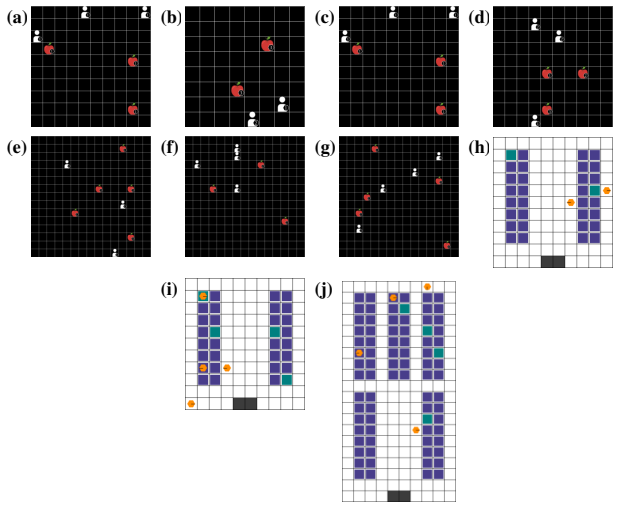



Figure 9: *Illustration of the seven LBF and three RWARE tasks used for the main experiments.*

**Used scenarios** In this situation, the experiments in our study were carried out using three different scenarios of the RWARE environment. In each of these scenarios, agents have a 3x3 observation grid centred on themselves, providing information on the location, rotation and surrounding configurations of other agents and shelves. By default, the number of requested shelves is equal to the number of agents.

- Figure 9(h) **rware-tiny-2ag**: The tiny map is a grid world of 11x11 squares, partial observability, 2 agents.
- Figure 9(i) **rware-tiny-4ag**: The tiny map is a grid world of 11x11 squares, partial observability, 4 agents.
- Figure 9(j) **rware-small-4ag**: The small map is a grid world of 11x20 squares, partial observability, 4 agents.

## Algorithms Details

**Algorithms Overview** In our analysis, we restrict ourselves to a limited set of algorithms from MARL literature. Our algorithm selection is done to cover Q-learning and policy gradient (PG) based methods in both the independent learner (IL) and centralised training decentralised execution (CTDE) paradigms. We also investigate the effect of parameter sharing and non-parameter sharing on the performance of the algorithms.

**Q-learning** For Q-learning-based methods we have selected, VDN and QMIX which fall into the paradigm of **CTDE** and IQL which is an **IL** method.

**IQL:** For Independent Q-Learning (IQL) (Tan 1993), each agent learns a policy based purely on their own egocentric experience in the training environment. This policy is parameterised by a Q-value network (Mnih et al. 2013).

**VDN:** In Value-Decomposition Network (VDN) (Sunehag et al. 2017b), IQL is extended through the use of value decomposition. Rather than learning purely from their own egocentric perspectives with each agent receiving the same reward, VDN formulates the joint Q value of the coalition as a linearly decomposed sum of the individual agent values. Each individual agent then updates its policy using the gradient flow based on a joint additive loss.

**QMIX**: (Rashid et al. 2018b) then extends VDN by broadening the range of reward functions that can be decomposed. To create a more complex attribution of the Q values, it makes use of a parameterised mixing network to perform the attribution. This mixing network takes the individual agent Q values as input and then policy updates are performed in an end-to-end manner where attribution is done using backpropagation. Qmix also allows the use of data augmentation by accommodating additional data at training time.

**Policy Gradients (PG):** **IA2C:** Independent Advantage Actor-Critic (IA2C) is a variant of the A2C algorithm (Mnih et al. 2016b) applied to the multi-agent setting. IA2C trains agents using their own egocentric experiences in the training environment where each agent has their own critic and actor networks that approximate the optimal policy and state values.

**IPPO:** Independent Proximal Policy Optimisation (IPPO) is a variant of the PPO algorithm (Schulman et al. 2017) applied to the multi-agent setting. PPO can be thought of as an improvement to A2C. It uses a surrogate objective which limits the change in the policy at each update step which allows PPO to iterate over the same trajectory of data multiple times without policy divergence. Otherwise, its architecture is the same as A2C.

**MAPPO & MAA2C:** Multi-Agent Proximal Policy Optimisation (MAPPO) and Multi-Agent Advantage Actor-Critic (MAA2C) (Yu et al. 2022) extend IPPO and IA2C to make use of a joint state value function. Instead of multiple per-agent critics, there is a single critic that learns the value of the joint state representation rather than the egocentric individual agent observations. MAA2C is also sometimes referred to as Central-V (Foerster et al. 2018) because of this but, to prevent confusion, we use MAA2C. MAPPO also makes use of the same CTDE type architecture with a centralised critic.

**Parameter Sharing vs Non-Parameter Sharing:** To improve sample efficiency it is common to use **Parameter Sharing** (PS) in cooperative MARL. When PS is in use, all of the agents on a team share the same set of parameters for their neural networks (NN). In practice, this is equivalent to using a single neural network to represent all members of the team. Typically a one-hot agent ID is added to the local observation of each agent so that the NN can determine which agent to behave as. In some cases, using PS limits performance as agents tend to learn a smaller subset of roles. Alternatively, we can use **concurrent/non-parameter shared** learning where each agent is represented by a different set of parameters. Under this paradigm, we train each agent's parameters concurrently and maintain separate parameters for each individual agent.

## Evaluation Protocol

**Aggregation Metrics:** **Median**: The median is the 50th percentile, representing the central point of the sorted raw data. Counts of the datapoints on either side of the median will thus be the same.

**IQM**: The interquartile mean (IQM) or midmean is a measure of central tendency evaluated based on the truncated mean of the interquartile range. It involves computing the mean over the values that fall within the interquartile range, which is the range between the 25th and 75th percentiles of the data.

**Optimality Gap**: The optimality gap is the difference between the aggregated value and the optimal value. It provides insight into the performance by quantifying the deviation from the best achievable outcome.

**Absolute Metric**: The Absolute Metric represents the average performance achieved by the best policy obtained throughout the entire learning process. It's computed by evaluating the algorithm over a number of independent evaluation episodes that is 10 times greater than the original number used during training.

**Explanation of plots used:** **Sample efficiency** The concept of sample efficiency is used to evaluate how effectively an algorithm improves its performance on a specific measure in relation to the amount of data it samples during the training process. These curves are generated by calculating the normalised average performance at each evaluation interval.

**Performance Profiles**: Performance profiles plot the probability that the normalised return of an algorithm is greater than some fraction of a predetermined value. From these plots, we can see the likelihood of algorithms reaching an optimal score and compare their relative performance at different points.

**Probability of improvement**: The probability of improvement are plots that indicate the probability that algorithm X has superior performance than algorithm Y with a low score indicating that algorithm Y is likely to be better than algorithm X and vice versa for a high score.

**Experimental Hyperparameters** In our analysis, we sought to conduct comprehensive experiments in various environments using different algorithms. To ensure reliable and consistent results, it is important carefully select and optimize the hyperparameters for each algorithm in each environment.

To this end, we used the optimized hyperparameters from (Papoudakis et al. 2021), where the parameters of each algorithm were chosen based on a hyperparameter sweep for a single scenario of each environment and then reused across all other scenarios for the same settings. The choice of the set of hyperparameters is done by selecting the one with the highest evaluation score averaged over three seeds.

Tables 1 and 5 provide a summary of the shared hyperparameters used in the Q-learning and policy gradient algorithms, respectively. On the other hand, Tables 2, 3, and 4 specify the algorithm-specific hyperparameters for each Q-learning algorithm, namely IQL, VDN, and QMIX, respectively. Similarly, Tables 6 and 7 present the specific hyperparameter settings for the policy-gradient algorithms, namely MAPPO and MAA2C, respectively.

These aforementioned tables offer an overview of the hyperparameters utilized in each environment, encompassing the applicable algorithms in both parameter-sharing and non-parameter-sharing scenarios.

Table 1: Shared hyperparameters for Q-learning algorithms with and without parameter sharing

| | **Parameter Sharing** | | **Non-Parameter Sharing** | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Optimizer | Adam | Adam | Adam | Adam |
| Maximum gradient norm | 10 | 10 | 10 | 10 |
| Reward standardisation | True | True | True | True |
| Network type | GRU | FC | GRU | FC |
| Discount factor | 0.99 | 0.99 | 0.99 | 0.99 |
| $\epsilon$ schedule steps | 2e6 | 5e4 | 5e4 | 5e4 |
| $\epsilon$ schedule minimum | 0.05 | 0.05 | 0.05 | 0.05 |
| Batch size | 32 | 32 | 32 | 32 |
| Replay buffer size | 5000 | 5000 | 5000 | 5000 |
| Parallel workers | 1 | 1 | 1 | 1 |

| | **Parameter Sharing** | | **Non-Parameter Sharing** | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Hidden dimension | 128 | 64 | 64 | 64 |
| Learning rate | 0.0003 | 0.0005 | 0.0003 | 0.0005 |
| Reward standardisation | True | True | True | True |
| Network type | GRU | FC | GRU | FC |
| Evaluation epsilon | 0.05 | 0.05 | 0.05 | 0.05 |
| Target update | 200(hard) | 0.01(soft) | 200(hard) | 0.01 (soft) |

Table 2: Shared hyperparameters for IQL with and without parameter sharing

| | **Parameter Sharing** | | **Non-Parameter Sharing** | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Hidden dimension | 128 | 64 | 64 | 64 |
| Learning rate | 0.0003 | 0.0005 | 0.0001 | 0.0005 |
| Reward standardisation | True | True | True | True |
| Network type | GRU | FC | GRU | FC |
| Evaluation epsilon | 0.0 | 0.05 | 0.05 | 0.05 |
| Target update | 0.01(**soft**) | 0.01(**soft**) | 200(**hard**) | 0.01 (**soft**) |

Table 3: Hyperparameters for VDN with and without parameter sharing

| | **Parameter Sharing** | | **Non-Parameter Sharing** | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Hidden dimension | 64 | 64 | 64 | 64 |
| Network type | GRU | FC | GRU | FC |
| Mixing network size | 32 | 32 | 32 | 32 |
| Mixing network type | FC | FC | FC | FC |
| Mixing network activation | ReLU | ReLU | ReLU | ReLU |
| Hypernetwork size | 64 | 64 | 64 | 64 |
| Hypernetwork activation | ReLU | ReLU | ReLU | ReLU |
| Hypernetworks layers | 2 | 2 | 2 | 2 |
| Learning rate | 0.0003 | 0.0005 | 0.0001 | 0.0003 |
| Reward standardisation | True | True | True | True |
| Evaluation epsilon | 0.05 | 0.05 | 0.05 | 0.05 |
| Target update | 0.01(**soft**) | 0.01(**soft**) | 0.01 (**soft**) | 0.01 (**soft**) |

Table 4: Hyperparameters for QMIX with and without parameter sharing

Table 5: Shared hyperparameters for Policy-based algorithms with and without parameter sharing

| | Parameter Sharing | | Non-Parameter Sharing | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Optimizer | Adam | Adam | Adam | Adam |
| Maximum gradient norm | 10 | 10 | 10 | 10 |
| Discount factor | 0.99 | 0.99 | 0.99 | 0.99 |
| Entropy coefficient | 0.001 | 0.001 | 0.001 | 0.001 |
| Batch size | 10 | 10 | 10 | 10 |
| Replay buffer size | 10 | 10 | 10 | 10 |
| Parallel workers | 10 | 10 | 10 | 10 |

| | Parameter Sharing | | Non-Parameter Sharing | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Hidden dimension | 128 | 128 | 128 | 128 |
| Learning rate | 0.0003 | 0.0005 | 0.0001 | 0.0005 |
| Reward standardisation | False | False | False | False |
| Network type | FC | FC | FC | FC |
| Evaluation epsilon | 0.05 | 0.05 | 0.05 | 0.05 |
| Epsilon clip | 0.2 | 0.2 | 0.2 | 0.2 |
| Epochs | 4 | 4 | 4 | 4 |
| Target update | 0.01(soft) | 0.01(soft) | 200 (hard) | 0.01 (soft) |
| n-step | 5 | 10 | 10 | 10 |

Table 6: Hyperparameters for MAPPO with and without parameter sharing

| | Parameter Sharing | | Non-Parameter Sharing | |
|---|---|---|---|---|
| | **LBF** | **RWARE** | **LBF** | **RWARE** |
| Hidden dimension | 128 | 64 | 128 | 64 |
| Learning rate | 0.0005 | 0.0005 | 0.0005 | 0.0005 |
| Reward standardisation | True | True | True | True |
| Network type | GRU | FC | GRU | FC |
| Evaluation epsilon | 0.01 | 0.01 | 0.01 | 0.01 |
| Target update | 0.01(soft) | 0.01(soft) | 0.01 (soft) | 0.01 (soft) |
| n-step | 10 | 5 | 5 | 5 |

Table 7: Hyperparameters for MAA2C with and without parameter sharing

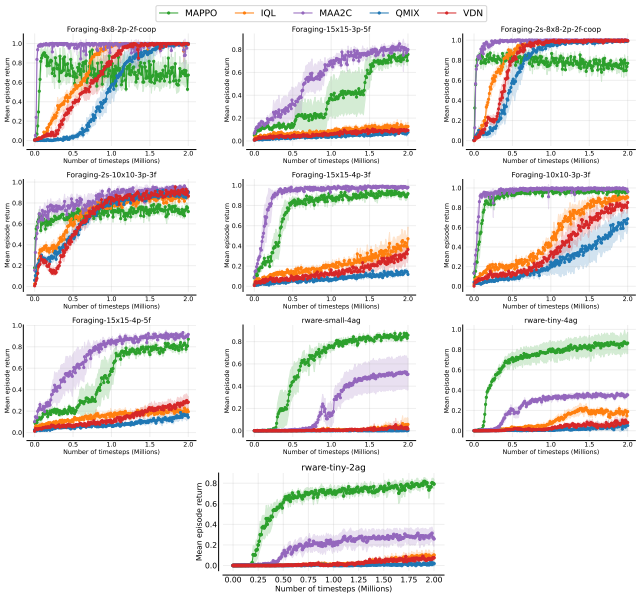



Figure 10: Mean episode returns for all algorithms with parameter sharing in seven LBF scenarios and three RWARE scenarios, with the mean and 95% confidence intervals over 10 distinct seeds.

Table 8: Normalized Episode Return: Aggregated Scores with 95% Confidence Intervals in the LBF Environment with Parameter Sharing

| | **MAPPO** | **IQL** | **MAA2C** | **QMIX** | **VDN** |
|---|---|---|---|---|---|
| Median | 0.84 ±0.03 | 0.87 ±0.01 | 0.98 ±0.01 | 0.67 ±0.12 | 0.78 ±0.06 |
| IQM | 0.85 ±0.01 | 0.71 ±0.04 | 0.97 ±0.01 | 0.58 ±0.03 | 0.67 ±0.04 |
| Mean | 0.85 ±0.01 | 0.64 ±0.02 | 0.95 ±0.01 | 0.56 ±0.02 | 0.61 ±0.02 |
| Optimality Gap | 0.15 ±0.01 | 0.36 ±0.02 | 0.05 ±0.01 | 0.44 ±0.02 | 0.39 ±0.02 |

# B Main experiment results

In this section, we present additional plots that complement the figures presented in the main paper. These plots provide a more comprehensive visualization of the experimental results and support the analysis presented in the paper.

## Parameter sharing Experiments

In Figure 10, we can observe the performance of the aforementioned algorithms in the seven LBF tasks and the 3 RWARE tasks where we recorded the results of Mean episode returns with the mean and 95% confidence intervals over 10 distinct seeds in the 201 evaluations.

In contrast, Tables 8 and 9 present the comprehensive tabulated results of algorithms performance in the LBF and RWARE environments. These tables showcase the utilization of various aggregation metrics, including Median, IQM, Mean, and the Optimality gap, along with their corresponding 95% confidence intervals. The confidence intervals are estimated using the percentile bootstrap with stratified sampling.

Table 9: Normalized Episode Return: Aggregated Scores with 95% Confidence Intervals in the RWARE Environment with Parameter Sharing

| | MAPPO | IQL | MAA2C | QMIX | VDN |
|---|---|---|---|---|---|
| Median | 0.85 ±0.04 | 0.11 ±0.06 | 0.35 ±0.03 | 0.03 ±0.02 | 0.07 ±0.03 |
| IQM | 0.85 ±0.03 | 0.12 ±0.05 | 0.39 ±0.05 | 0.0 ±0.01 | 0.06 ±0.03 |
| Mean | 0.83 ±0.04 | 0.13 ±0.04 | 0.41 ±0.04 | 0.02 ±0.02 | 0.07 ±0.02 |
| Optimality Gap | 0.17 ±0.04 | 0.87 ±0.04 | 0.59 ±0.04 | 0.98 ±0.02 | 0.93 ±0.02 |

Table 10: Normalized Episode Return: Aggregated Scores with 95% Confidence Intervals in the LBF Environment without Parameter Sharing

| | MAPPO | IQL | MAA2C | QMIX | VDN |
|---|---|---|---|---|---|
| Median | 0.91 ±0.03 | 0.23 ±0.02 | 0.35 ±0.03 | 0.78 ±0.03 | 0.46 ±0.08 |
| IQM | 0.93 ±0.01 | 0.34 ±0.03 | 0.42 ±0.01 | 0.77 ±0.01 | 0.43 ±0.04 |
| Mean | 0.89 ±0.01 | 0.42 ±0.02 | 0.47 ±0.01 | 0.74 ±0.02 | 0.48 ±0.03 |
| Optimality Gap | 0.11 ±0.01 | 0.58 ±0.02 | 0.53 ±0.01 | 0.26 ±0.02 | 0.52 ±0.03 |

## Non-Parameter sharing Experiments

Similar to the replication of experiments conducted in the parameter sharing case, we also conducted a replication of the outcomes when agents do not share learning parameters. The outcomes of these experiments are illustrated in Figure 11.

By examining the results for both the LBF and RWARE environments, we aimed to compare and contrast the performance of algorithms under these distinct conditions. The replicated experiments serve to validate and complement the findings presented in Figure 11, offering an understanding of the impact of parameter sharing on algorithm performance.

In addition, in Figure 12 we also evaluated the performance of various algorithms without parameter sharing across a range of scenarios. For the LBF environment, we considered seven different scenarios, each presenting unique challenges and variations in agent and food levels. Similarly, for the RWARE environment, we explored three distinct scenarios, encompassing different warehouse sizes and numbers of agents.

Similarly to the parameter sharing case discussed in Section B, the results of algorithm performance in the LBF and RWARE environments are presented in Tables 8 and 9.

## Additional results on RWARE

We perform additional experimentation comparing agent importance and the Shapley values in RWARE which is a sparse setting. From figures 13 and 14 we can see that both method perform similarly in the sparse setting when performance

Table 11: Normalized Episode Return: Aggregated Scores with 95% Confidence Intervals in the LBF Environment without Parameter Sharing

| | MAPPO | IQL | MAA2C | QMIX | VDN |
|---|---|---|---|---|---|
| Median | 0.45 ±0.08 | 0.02 ±0.02 | 0.03 ±0.02 | 0.48 ±0.14 | 0.08 ±0.03 |
| IQM | 0.41 ±0.05 | 0.01 ±0.01 | 0.04 ±0.02 | 0.47 ±0.1 | 0.09 ±0.03 |
| Mean | 0.44 ±0.06 | 0.02 ±0.01 | 0.04 ±0.01 | 0.47 ±0.09 | 0.09 ±0.02 |
| Optimality Gap | 0.56 ±0.06 | 0.98 ±0.01 | 0.96 ±0.01 | 0.53 ±0.09 | 0.91 ±0.02 |

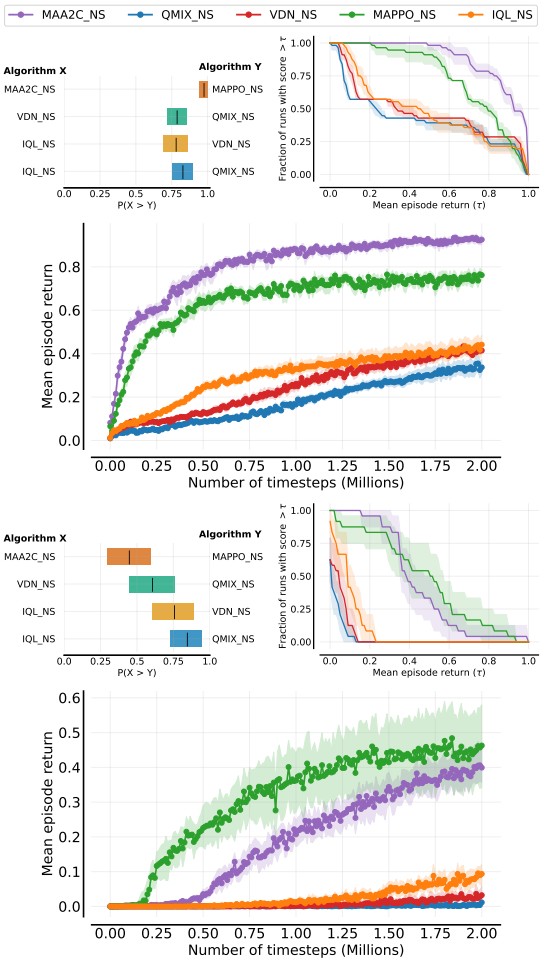



Figure 11: *Results from running the same experimental hyperparameters on the same tasks as (Papoudakis et al. 2021) including the probability of improvement, performance profiles and sample efficiency curves*. **Top row:** Performance of algorithms on 7 LBF tasks. **Bottom row:** Performance of all algorithms on 3 RWARE tasks.

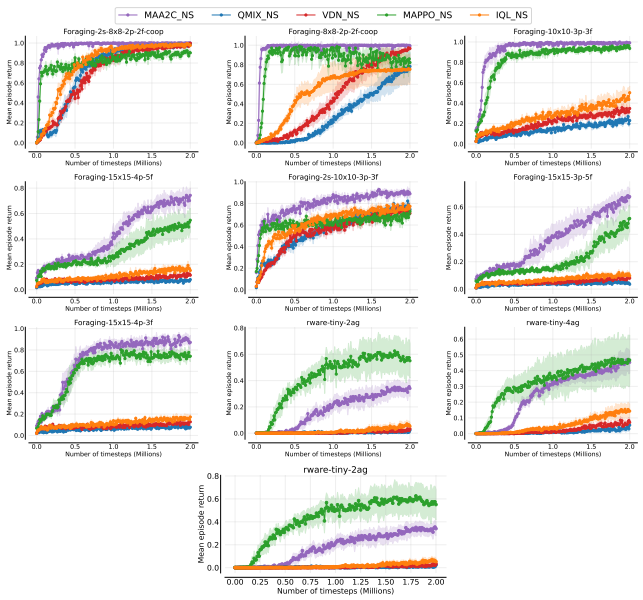



Figure 12: Mean episode returns for all algorithms without parameter sharing in seven LBF scenarios and three RWARE scenarios, with the mean and 95% confidence intervals over 10 distinct seeds.

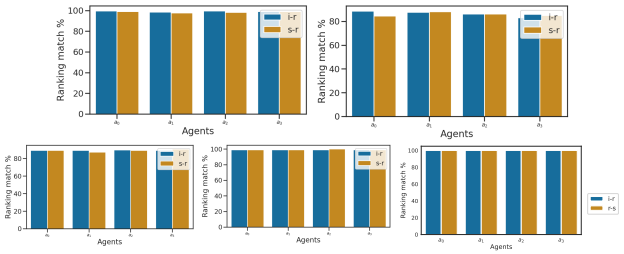



Figure 13: Comparison of Shapley value and agent importance rankings to individual rewards for IQL, MAA2C, MAPPO, VDN and QMIX on rware-small-4ag-v1.

is poor like for IQL, QMIX and VDN. However when the agents are able to achieve some level of success, accuracy drops for both methods. This is likely due to the sparse reward creating many zeros in the data and creating erroneous predictions. This is most noticeable for MAPPO and MAA2C where the Shapley Value estimations can drop below 90%. We can further verify this from figures 17 and 18 where the agent importance and Shapey values exhibit similar variance over time.

## Aggregation of agent importance

Throughout the paper we use a single seed to display agent importance over time. For homogeneous settings with parameter sharing this is required as agents can take different roles in each seed depending on the training conditions. Essentially agents of a similar type can fulfil multiple different sub-roles during training which makes aggregating agent contributions over multiple seeds inconsistent in the stochastic case as seen

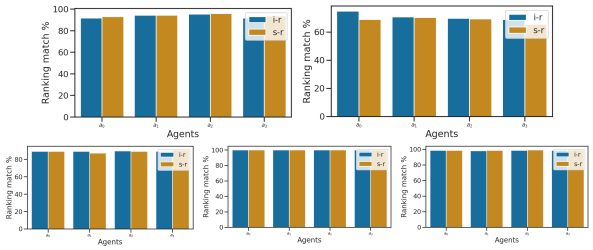
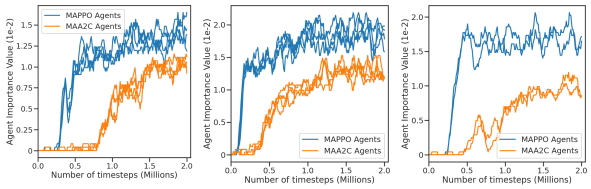
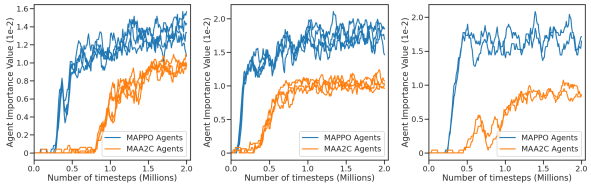



Figure 14: Comparison of Shapley value and agent importance rankings to individual rewards for IQL, MAA2C, MAPPO, QMIX and VDN on rware-tiny-4ag-v1



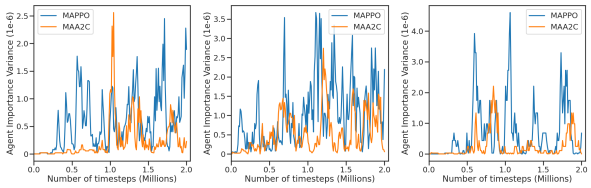

(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 15: Comparisons of the agent importance on rware-small-4ag-v1 for MAPPO and MAA2C



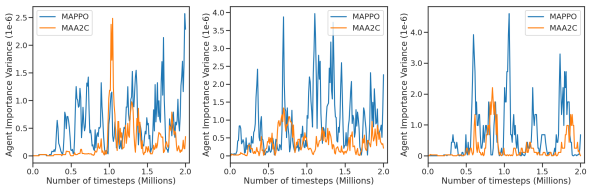

(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 16: Comparisons of the Shapley values on RWARE for MAPPO and MAA2C



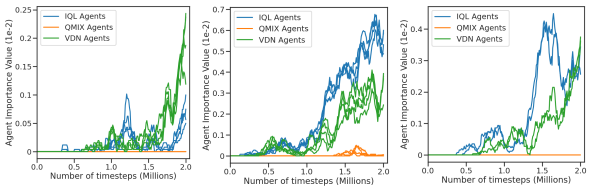

(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 17: Comparisons of the agent importance variance on RWARE for MAPPO and MAA2C



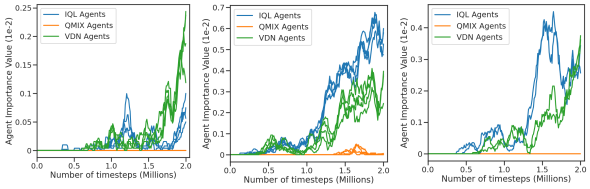

(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 18: Comparisons of the Shapley value variance RWARE for MAPPO and MAA2C



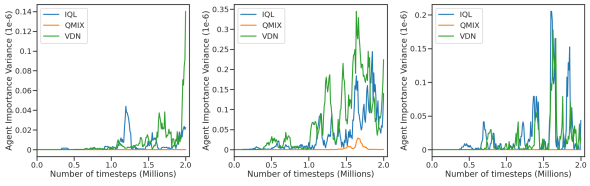

(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 19: Comparisons of the agent importance on RWARE for QMIX, VDN and IQL



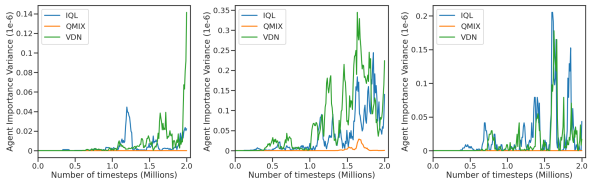

(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 20: Comparisons of the Shapley value on RWARE for QMIX, VDN and IQL




(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 21: Comparisons of the agent importance variance on RWARE for QMIX, VDN and IQL




(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

Figure 22: Comparisons of the Shapley value variance on RWARE for QMIX, VDN and IQL

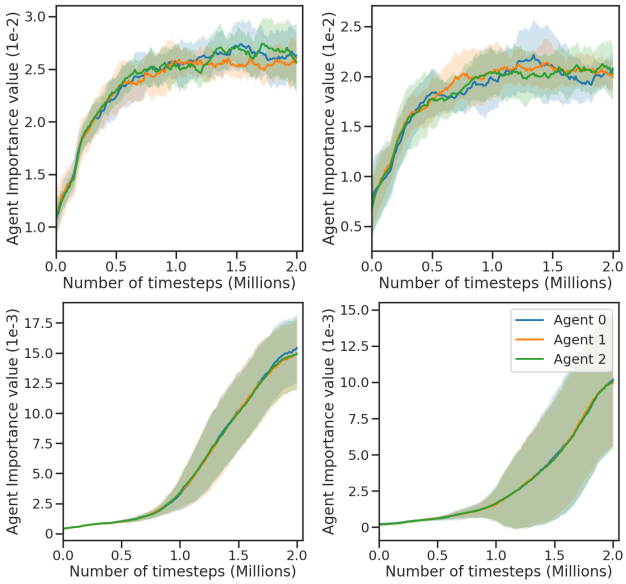



Figure 23: *Agent importance scores on the stochastic Foraging-10x10-3p-3f-v2 LBF scenario for MAA2C, MAPPO, VDN and QMIX.*

in 23. When compared to the parameter sharing explain in figure 7 we can see that determining individual agent contributions becomes difficult. Similar issues can be seen in figure 24 when compared to figure 6.

## C On heterogeneous settings

For most of our experiments, we make use of the LBF and RWARE settings. RWARE is completely homogeneous as all agents have the same capabilities as each other and their roles and importance within the coalition are developed during training time as the individual policies associated with each agent's IDs are learnt. This means that agent roles are inconsistent across seeds and parameterisation as depending on external factors across runs, different agent IDs can occupy different roles. For LBF, agents are homogeneous in their action space but their importance rankings are essentially preassigned due to their levels determining the extent of their contribution towards collecting food. Given the limitations of these settings, it is important to determine how effectively *agent importance* is able to determine the contributions of

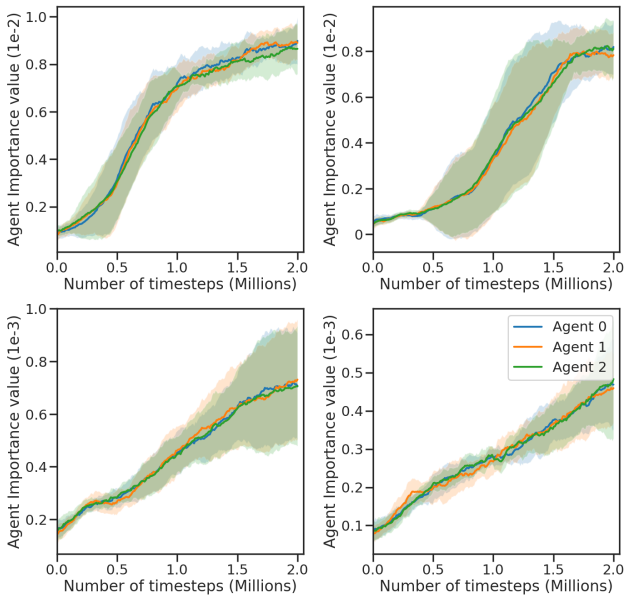
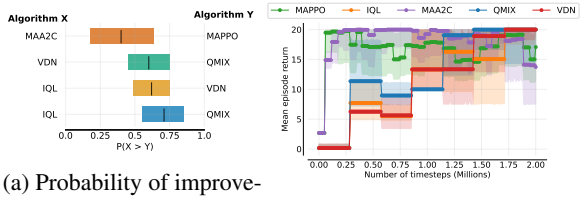



Figure 24: *Agent importance scores on the stochastic lbforaging:Foraging-15x15-3p-5f-v2 LBF scenario for MAA2C, MAPPO, VDN and QMIX.*

agents in complex heterogeneous settings where there are clear agent types with differing capabilities that compose the coalition.

A popular setting with heterogeneous agents in cooperative MARL is the Starcraft Multi-Agent Challenge (SMAC) (Samvelyan et al. 2019) however, this environment has 2 limitations which make applying agent importance difficult. Firstly, it uses the original game engine for the Starcraft 2 (SC2) video game which is coded in the C++ programming language as a back-end. This makes it unsuitable for creating multiple parallel copies of the setting using the built-in Python copy method which makes applying contribution calculation methods like the Shapley value and agent importance challenging. Secondly, the SC2 engine back-end is computationally expensive to run and the high resource requirements make using contribution calculation methods on top of the existing environment unappealing. Instead, we make use of SMAClite (Michalski, Christianos, and Albrecht 2023), which implements a setting similar to SMAC but in purely Python code which allows it to be copied and reduces computational requirements.




(a) Probability of improvement

(b) Sample efficiency

Figure 25: Algorithms performance on SMAClite without parameter sharing

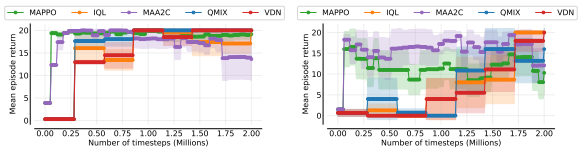



Figure 26: **Left:** Mean episode return on 2s3z. **Right:** Mean episode return on 3m.

We plot performance over training using the absolute metric in figure 26 for the 3m and 2s3z scenarios. We average results over 6 seeds rather than the 5 used in the SMAClite paper and run the policy gradient (PG) methods for 20 million timesteps and the Q-learning methods for 2 million as recommended in the EPymarl benchmark (Papoudakis et al. 2021). We found there to be high variance in the performance across seeds for the 3m setting especially for the PG methods which often experienced significant performance decay later into training however, for the more complex 2s3z setting performance was fairly stable and algorithms quickly converged to reasonable policies.

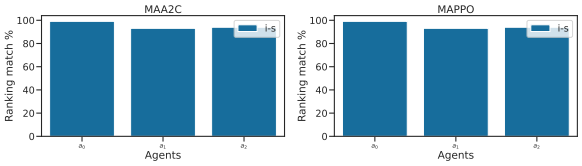



Figure 27: Ranking agreement percentages for MAPPO and MAA2C for 3m

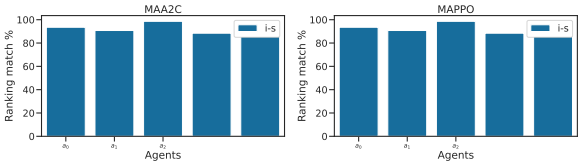
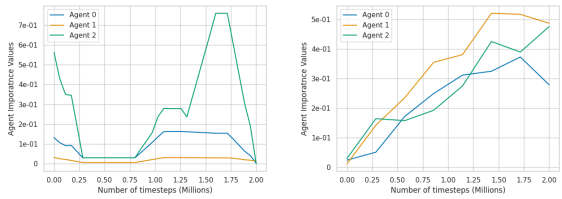


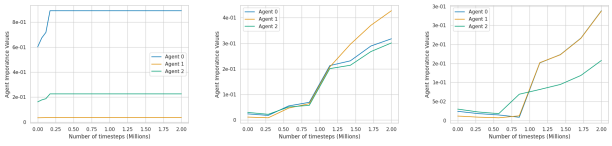

Figure 28: Ranking agreement percentages for MAPPO and MAA2C for 2s3z

Firstly, given the high variance between seeds we determine how accurately agent importance is able to capture the inddividual rewards that compose the joint reward in SMAClite. Unlike the RWARE and LBF setting, SMAC and SMAClite do not produce individual rewards which can be used as a ground truth value. Therefore instead of comparing contribution methods to the individual rewards we directly compare agent importance and the Shapley value were we take the shapley value to be an accurate approximation of the ground truth. We note that for 3m, despite the high variance in return across all methods, agent importance and the Shapley value have near 100 percent agreement w.r.t agent rankings. When moving to 2s3z agreement drops to 90 percent for agents 0 and 1 but remains near 100 percent for agents 2 to 4. As agent 0 and 1 are of the same type. Possibly this indicates that agent importance is effective in determining the relative contributions of each agent but as it does not

calculate the value of all possible coalitions it can produce erroneous values when multiple agents are closely related but have different importance.



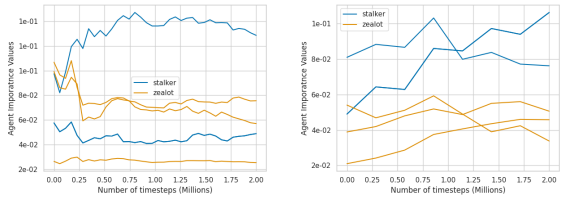

(a) MAPPO agent importance (b) IQL agent importance



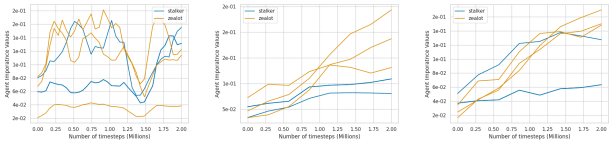
(c) MAA2C agent importance (d) QMIX agent importance (e) VDN agent importance

Figure 29: Agent importance plots for 3m from seed 0

In the homogeneous setting of 3m, we can see from figure 29 that agent importance follows a similar trend to the RWARE and LBF settings. As agents perform similar functions in the setting, their importance values are closely related. Agent importance also has a high percent ranking match rate with the Shapley values in this case as we can see in figure 27.




(a) MAPPO agent importance (b) IQL agent importance



(c) MAA2C agent importance (d) QMIX agent importance (e) VDN agent importance

Figure 30: Agent importance plots for 3m from seed 0

In the heterogeneous setting of 2s3z, we can see from figure 30 that the agents tend to naturally separate into very distinct importance ranges. This is more distinct when stable converge has been reached as we can see with MAPPO where after reaching an optimal solution, the agents no longer have highly similar importance values. Comparatively when convergence is unstable like with MAA2C agent importance will oscillate. It is also notable that even agents of the same type can have high variation in contribution score at the end to training. This is inline with existing literature like (Yang et al. 2020b; Singh and Rosman 2023) which have show that the importance of different agent agent types varies across the settings of the original SMAC and that importance cannot be assigned uniformly. Additionally as agents in SMAC can die during the episode rollout, the relative importance of the remaining agents increases as dead agents cannot contribute to the coalition which can result in agents of the same type have different assigned weighting based on how long they are able to survive.

## Parameter Sharing for MMM2

We perform additional experimentation on MMM2 to gain insight into how parameter sharing affects agent importance in the heterogeneous case. We can see from figure 31 that unlike in LBF and RWARE, parameter sharing seems to degrade performance. This is most noticeable for MAA2C which is unable to converge to a stable policy across 6 seeds. This is expected as (Wen et al. 2022) provide theoretical evidence towards parameter sharing reducing effectiveness in heterogeneous settings. We can also observe that like the findings on SMAC by (Wen et al. 2022) showing that it is not a challenging enough to compare parameter vs non-parameter sharing for SOTA algorithms like HATRPO, the MMM2 setting that has been ported to SMAClite does also not show a large change in performance for MAPPO in both the shared and non-shared cases.

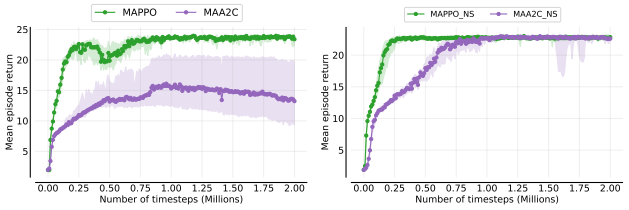



Figure 31: **Left:** Mean episode return for MMM2 with parameter sharing. **Righ:** Mean episode return for MMM2 with without parameter sharing.

From figures 32 and 33 we can see that the assigned agent importance rankings vary greatly between the parameter sharing (PS) and no-sharing (NS) results on MMM2. In the NS case the algorithm is able to learn clearer distinctions between the subgroups and doe not consistently over weight the value of the single marauder as it does in the PS case.

From figures 34 and 35 we can see that the assigned agent importance rankings are fairly similar for both the PS and NS case. In both cases from figure 31 we can see that MAPPO obtains a similar learning curve. Essentially this supports the claims made by (Wen et al. 2022) regarding evaluation of SOTA methods using PS. Although NS does improve performance and correct credit assignment in the heterogeneous case, the improvement is only noticeable if the PS variant of the algorithm is not already able to easily achieve optimal policies.

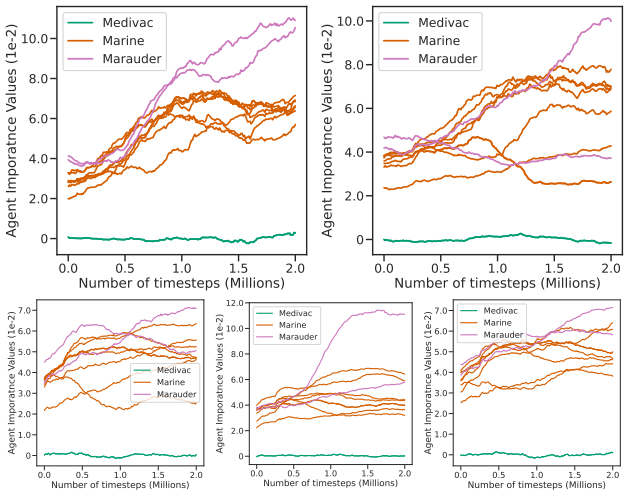



Figure 32: Agent importance plots for MAA2C with parameter sharing across 5 seeds in MMM2

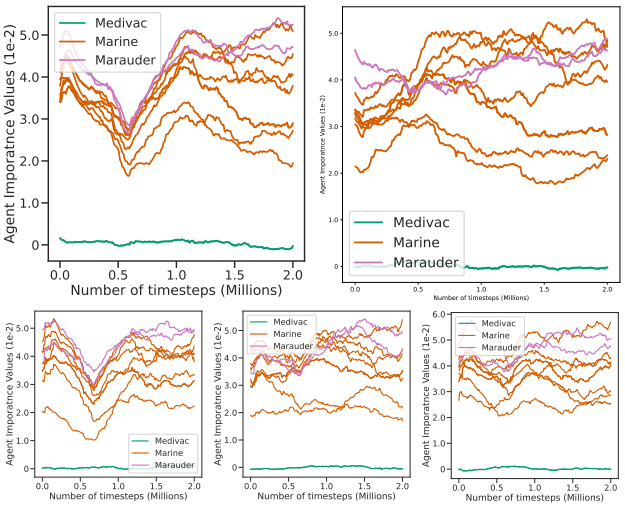



Figure 34: Agent importance plots for MAPPO with parameter sharing across 5 seeds in MMM2

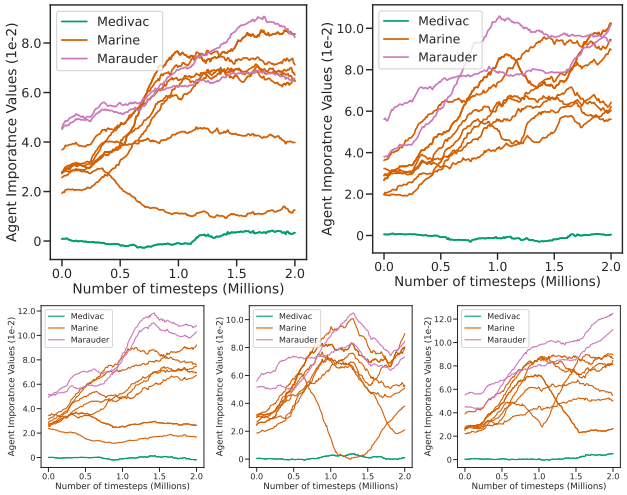



Figure 33: Agent importance plots for MAA2C without parameter sharing across 5 seeds in MMM2

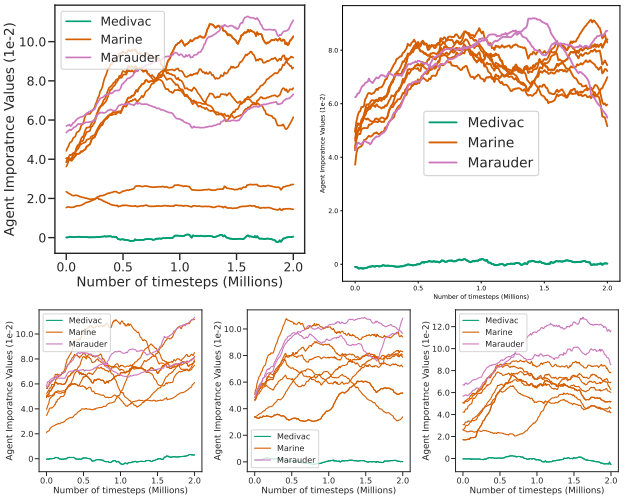



Figure 35: Agent importance plots for MAPPO without parameter sharing across 5 seeds in MMM2

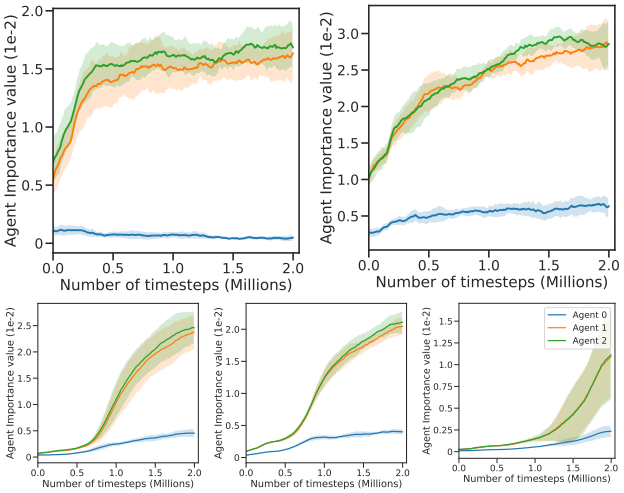



Figure 36: Agent importance scores on the **Foraging-15x15-3p-3f-det** scenario with parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

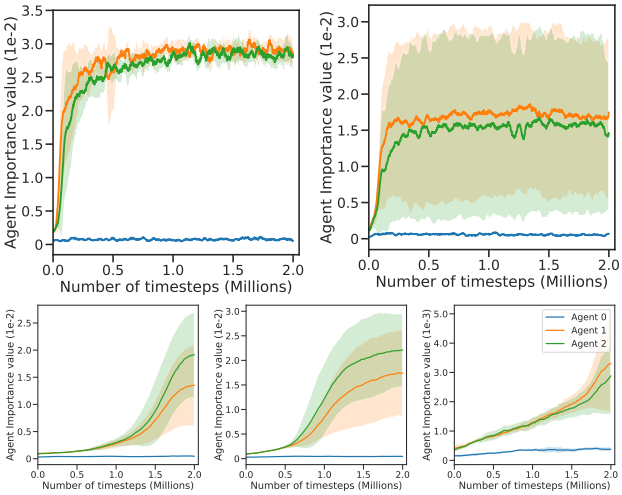



Figure 37: Agent importance scores on the **Foraging-15x15-3p-3f-det** scenario without parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

Table 12: Average time required to calculate agent contributions compared to the baseline with standard deviation

| Number of Agents | Baseline | Agent Importance | Shapley Value |
|---|---|---|---|
| 2 | 0.0018 ±0.0002 | 0.0074 ±0.0001 | 0.0079 ±0.0014 |
| 4 | 0.0023 ±0.0003 | 0.0118 ±0.0024 | 0.0313 ±0.0041 |
| 10 | 0.0038 ±0.0012 | 0.0392 ±0.0022 | 3.544 ±0.156 |
| 20 | 0.0088 ±0.0007 | 0.1697 ±0.0159 | 8065.3697 ±832.1829 |
| 50 | 0.0401 ±0.0019 | 1.6947 ±0.2295 | — |

# D Further Validation of Agent Importance

In this section, we present additional results, for further validation of the agent importance metric, thus proving its effectiveness. To do so, we set the tests on a deterministic version of a fixed scenario, within LBF, to assess the reliability of the metric and compare its scalability to the Shapley Value. We provide additional plots to analyze the correlation between the agent importance and the Shapley value.

## Metric Reliability

Figures 36 to 39 showcase the results for the agent importance analysis for all tested algorithms, considering both the parameter-sharing and non-parameter-sharing cases. In the deterministic LBF setting, as outlined in Section 1, agents 0, 1, and 2 are assigned levels of 1, 2, and 3, respectively. The figures demonstrate that agents with higher levels contribute more significantly. These findings are consistent across all algorithms and the reported values resulted from an aggregation over the 10 independent runs.

## Metric Scalability

In Table 12, we present the average time taken per step along with the standard deviation for the baseline algorithm without any metrics, with agent importance, and with the Shapley Value. These values were calculated over three independent runs for each scenario. The specific scenarios used in these

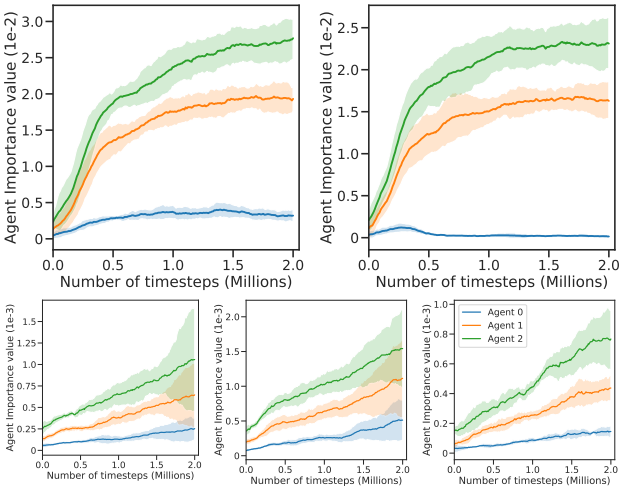



Figure 38: Agent importance scores on the **Foraging-15x15-3p-3f-det-max-food-sum** scenario with parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

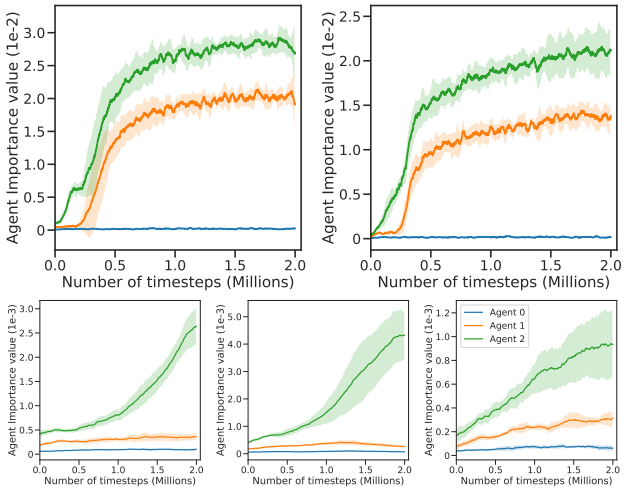



Figure 39: Agent importance scores on the **Foraging-15x15-3p-3f-det-max-food-sum** scenario without parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

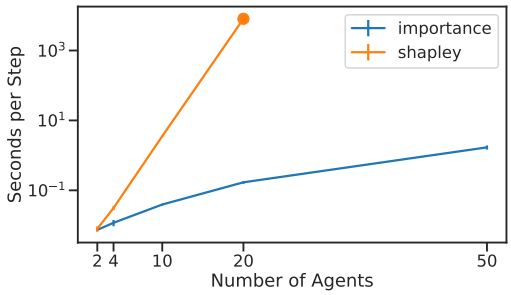



Figure 40: Computational cost of computing the Agent Importance and the Shapley value in seconds per environment step (log scale).

experiments are detailed in section A. We examined various cases by changing and varying the number of agents from 2 to 50. However, it is important to note that the experiment with 50 agents using the Shapley value took an exceptionally long time to complete and was therefore omitted. Nevertheless, the results obtained from the other experiments demonstrate the scalability w.r.t the number of agents of the Agent Importance metric compared to the Shapley value. In Figure 40, we present the time consumed by each metric per step, with the values plotted in a logarithmic scale. This scaling helps visualize the significant difference in the required time when using the Shapley Value.

## Correlation between Agent Importance and the Shapley value

To demonstrate the robust correlation between Agent Importance, the Shapley value, and individual reward, we present a comprehensive set of correlation heatmaps in Figures 45 to 54. Each heatmap corresponds to a specific algorithm-task

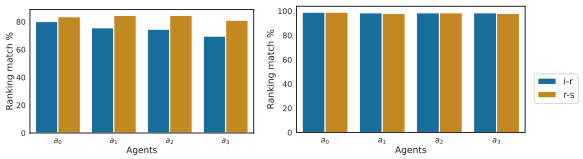



Figure 41: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

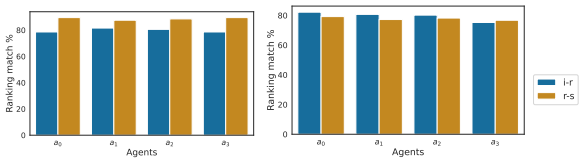



Figure 42: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

combination, as indicated by the title of the respective plot. It's important to note that these figures show a symmetrical pattern, unlike the asymmetry seen in Figure 2(a). Furthermore, in Figures 42 to 44, we have extended the analysis by examining the consistency of rankings among individual rewards, agent importance, and the Shapley value for the scenario depicted in Figure 2. However, it is worth noting that these specific plots are specific to alternative algorithms, namely IQL, QMIX, MAPPO, and MAA2C.

Moreover, we provide an overarching assessment of the correlation between Agent Importance and the Shapley value in Table 13. This evaluation entails computing the average correlation coefficient across multiple independent runs, algorithms, tasks, and agents, thereby yielding a consolidated metric. To ensure fairness, tasks involving 2, 3, and 4 agents are aggregated separately in the analysis.

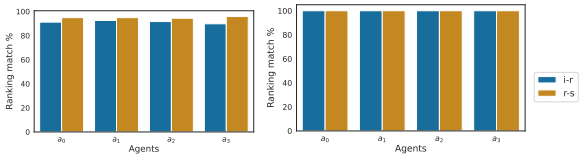



Figure 43: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

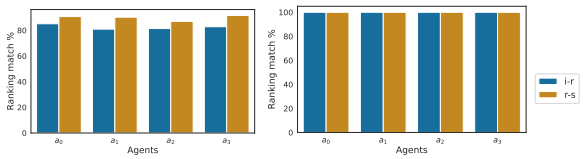



Figure 44: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

Table 13: *Average correlation of Agent Importance and the Shapley value.* Even when aggregating over multiple independent runs, algorithms, tasks, and agents, the strong correlation still holds.

| Number of Agents | Correlation Value |
|---|---|
| 2 | 0.97 $\pm$0.01 |
| 3 | 0.96 $\pm$0.01 |
| 4 | 0.96 $\pm$0.01 |

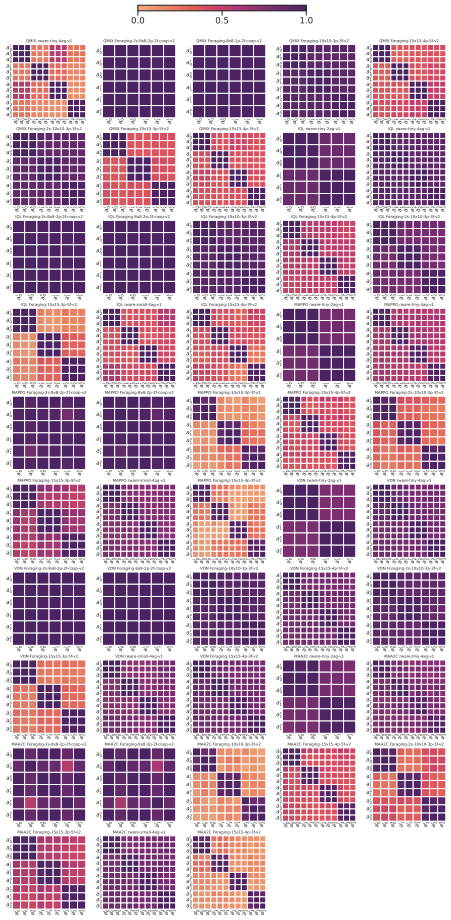


Figure 45: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the first independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

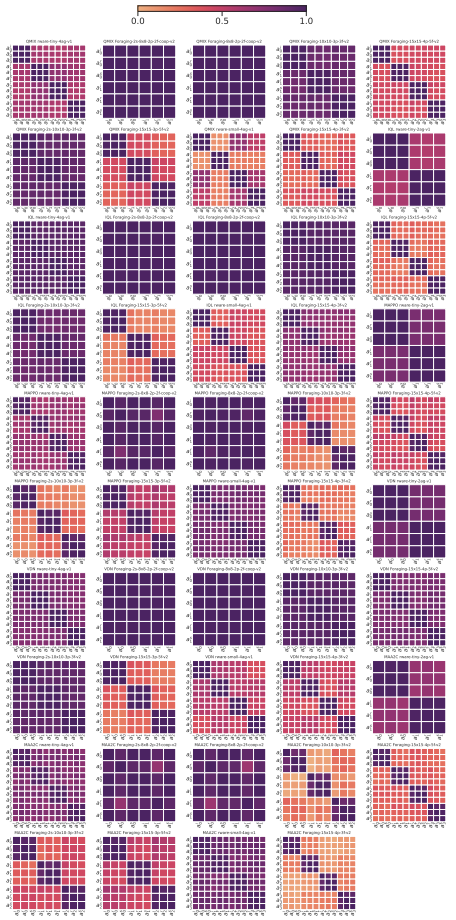


Figure 46: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the second independent run*. Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

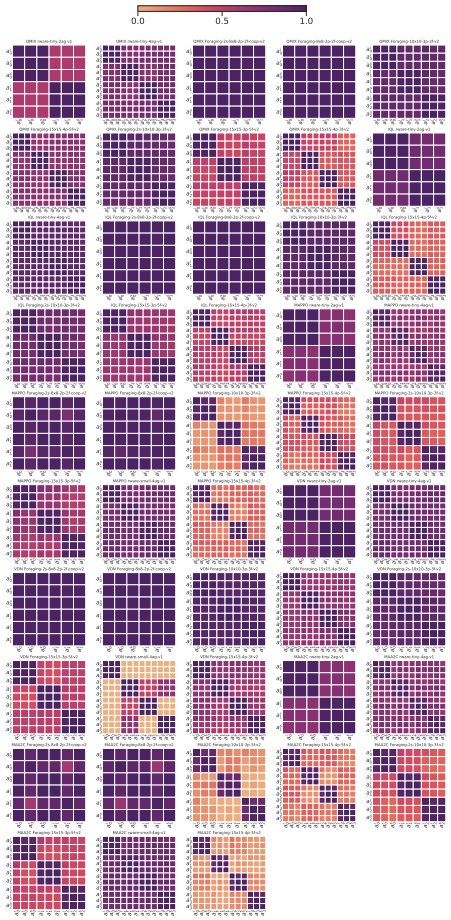


Figure 47: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the third independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

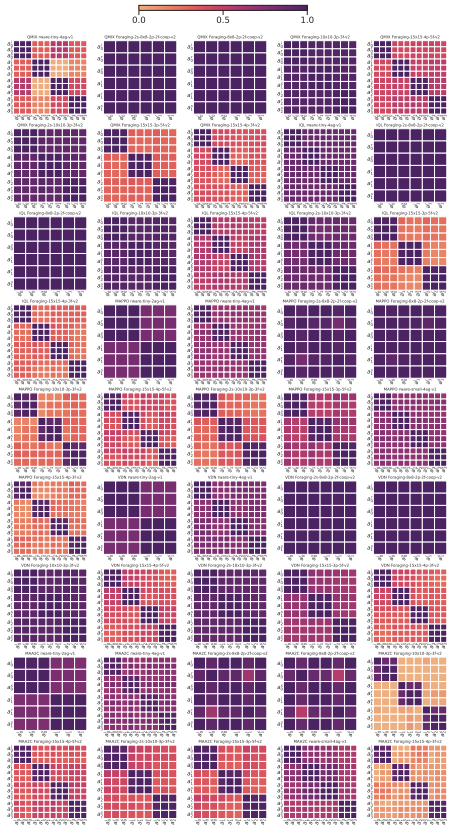


Figure 48: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the fourth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

[H]

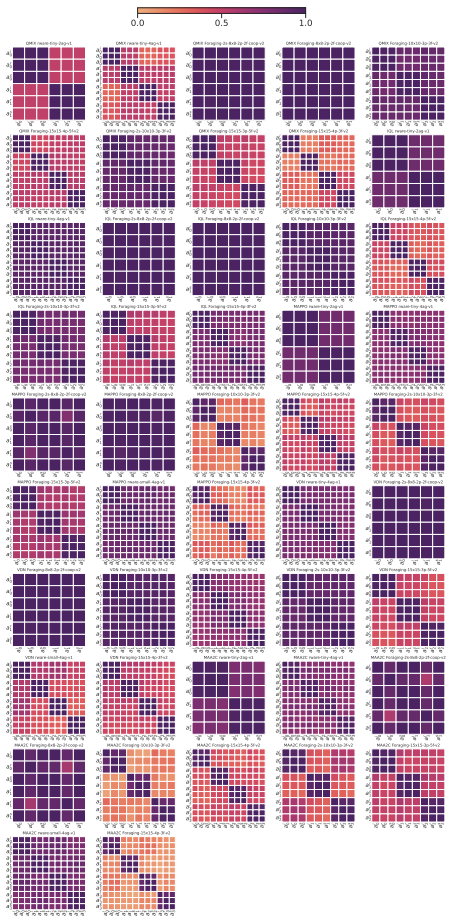


Figure 49: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the fifth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

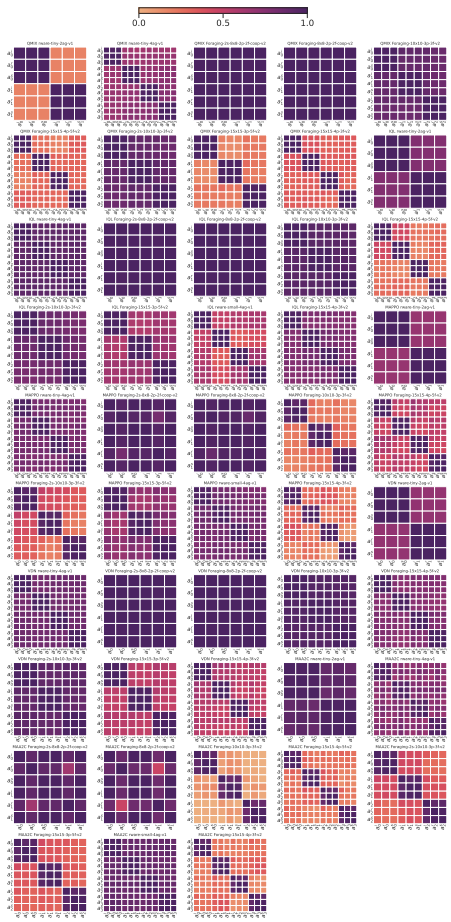


Figure 50: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the sixth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

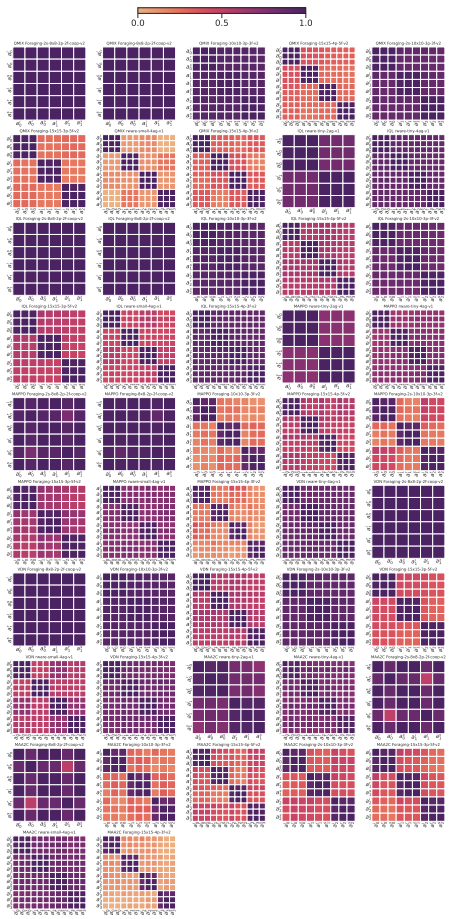


Figure 51: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the seventh independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

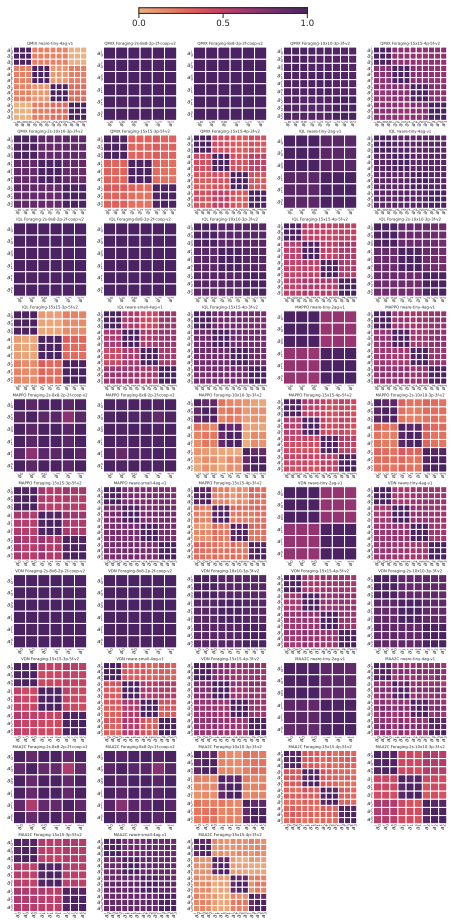


Figure 52: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the eighth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

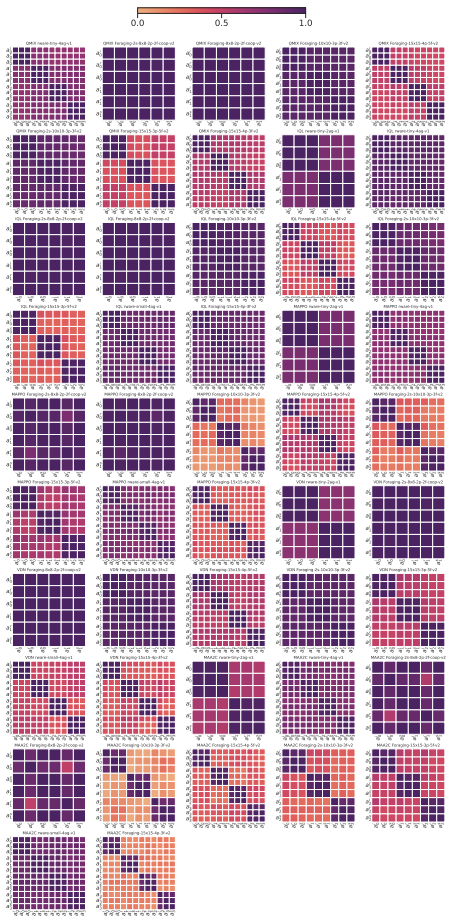


Figure 53: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the ninth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

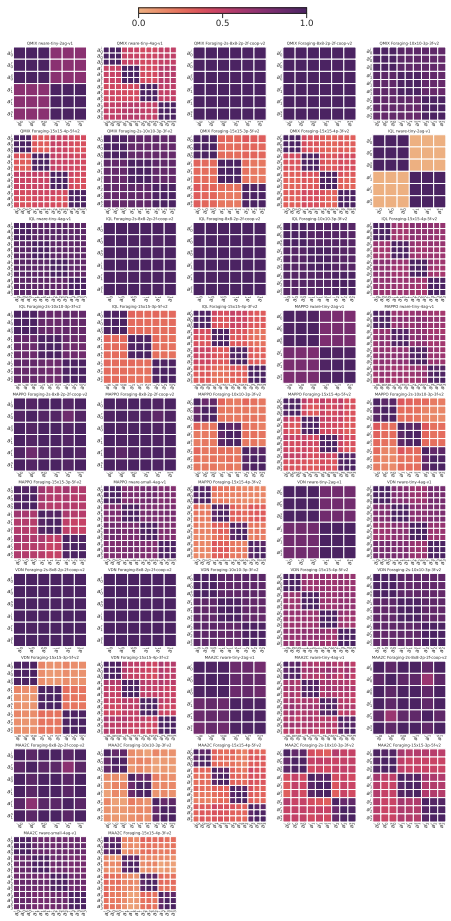


Figure 54: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the tenth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed across all algorithms and tasks.

## References


Agarwal, R.; Schwarzer, M.; Castro, P. S.; Courville, A.; and Bellemare, M. G. 2022. Deep Reinforcement Learning at the Edge of the Statistical Precipice. arXiv:2108.13264.

Agogino, A. K.; and Tumer, K. 2004. Unifying temporal and structural credit assignment problems. In *Autonomous Agents and Multi-Agent Systems Conference*.

Agogino, A. K.; and Tumer, K. 2008. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17: 320–338.

Albrecht, S. V.; and Ramamoorthy, S. 2015. A Game-Theoretic Model and Best-Response Learning Method for Ad Hoc Coordination in Multiagent Systems. arXiv:1506.01170.

Albrecht, S. V.; and Stone, P. 2019. Reasoning about Hypothetical Agent Behaviours and their Parameters. arXiv:1906.11064.

Arrieta, A. B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.;

Benjamins, R.; et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58: 82–115.

Bakhtin, A.; Brown, N.; Dinan, E.; Farina, G.; Flaherty, C.; Fried, D.; Goff, A.; Gray, J.; Hu, H.; Jacob, A. P.; Komeili, M.; Konath, K.; Kwon, M.; Lerer, A.; Lewis, M.; Miller, A. H.; Mitts, S.; Renduchintala, A.; Roller, S.; Rowe, D.; Shi, W.; Spisak, J.; Wei, A.; Wu, D. J.; Zhang, H.; and Zijlstra, M. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378: 1067 – 1074.

Bard, N.; Foerster, J. N.; Chandar, S.; Burch, N.; Lanctot, M.; Song, H. F.; Parisotto, E.; Dumoulin, V.; Moitra, S.; Hughes, E.; Dunning, I.; Mourad, S.; Larochelle, H.; Bellemare, M. G.; and Bowling, M. 2020. The Hanabi challenge: A new frontier for AI research. *Artificial Intelligence*, 280: 103216.

Boggess, K.; Kraus, S.; and Feng, L. 2022. Toward Policy Explanations for Multi-Agent Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*.

Bonnet, C.; Luo, D.; Byrne, D.; Abramowitz, S.; Coyette, V.; Duckworth, P.; Furelos-Blanco, D.; Grinsztajn, N.; Kalloniatis, T.; Le, V.; Mahjoub, O.; Midgley, L.; Surana, S.; Waters, C.; and Laterre, A. 2023. Jumanji: a Suite of Diverse and Challenging Reinforcement Learning Environments in JAX.

Brittain, M.; and Wei, P. 2019. Autonomous Air Traffic Controller: A Deep Multi-Agent Reinforcement Learning Approach. arXiv:1905.01303.

Chang, Y.-H.; Ho, T.; and Kaelbling, L. 2003. All learning is local: Multi-agent learning in global reward games. *Advances in neural information processing systems*, 16.

Christianos, F.; Schäfer, L.; and Albrecht, S. 2020. Shared Experience Actor-Critic for Multi-Agent Reinforcement Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 10707–10717. Curran Associates, Inc.

Colas, C.; Sigaud, O.; and Oudeyer, P.-Y. 2018. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms. In *International conference on machine learning*, 1039–1048. PMLR.

Dazeley, R.; Vamplew, P.; and Cruz, F. 2023. Explainable reinforcement learning for broad-xai: a conceptual framework and survey. *Neural Computing and Applications*, 1–24.

Devlin, S.; Yliniemi, L.; Kudenko, D.; and Tumer, K. 2014. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 165–172.

Du, W.; Ding, S.; Zhang, C.; and Du, S. 2021. Modified action decoder using Bayesian reasoning for multi-agent deep reinforcement learning. *International Journal of Machine Learning and Cybernetics*, 12.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Foerster, J.; Song, F.; Hughes, E.; Burch, N.; Dunning, I.; Whiteson, S.; Botvinick, M.; and Bowling, M. 2019. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 1942–1951. PMLR.

Freeman, C. D.; Frey, E.; Raichuk, A.; Girgin, S.; Mordatch, I.; and Bachem, O. 2021. Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation.

Glanois, C.; Weng, P.; Zimmer, M.; Li, D.; Yang, T.; Hao, J.; and Liu, W. 2021. A survey on interpretable reinforcement learning. *arXiv preprint arXiv:2112.13112*.

Gorsane, R.; Mahjoub, O.; de Kock, R.; Dubb, R.; Singh, S.; and Pretorius, A. 2022. Towards a Standardised Performance Evaluation Protocol for Cooperative MARL. arXiv:2209.10485.

Gu, B.; Zhang, X.; Lin, Z.; and Alazab, M. 2021. Deep Multiagent Reinforcement-Learning-Based Resource Allocation for Internet of Controllable Things. *IEEE Internet of Things Journal*, 8(5): 3066–3074.

Han, S.; Wang, H.; Su, S.; Shi, Y.; and Miao, F. 2022. Stable and efficient Shapley value-based reward reallocation for multi-agent reinforcement learning of autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, 8765–8771. IEEE.

Heess, N. M. O.; Dhruva, T.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S. M. A.; Riedmiller, M. A.; and Silver, D. 2017. Emergence of Locomotion Behaviours in Rich Environments. *ArXiv*, abs/1707.02286.

Henderson, P.; Romoff, J.; and Pineau, J. 2018. Where Did My Optimum Go?: An Empirical Analysis of Gradient Descent Optimization in Policy Gradient Methods. *ArXiv*, abs/1810.02525.

Heuillet, A.; Couthouis, F.; and Díaz-Rodríguez, N. 2021. Explainability in deep reinforcement learning. *Knowledge-Based Systems*, 214: 106685.

Heuillet, A.; Couthouis, F.; and Díaz-Rodríguez, N. 2022. Collective explainable AI: Explaining cooperative strategies and agent contribution in multiagent reinforcement learning with shapley values. *IEEE Computational Intelligence Magazine*, 17(1): 59–71.

Hu, H.; and Foerster, J. N. 2021. Simplified Action Decoder for Deep Multi-Agent Reinforcement Learning. arXiv:1912.02288.

Juozapaitis, Z.; Koul, A.; Fern, A.; Erwig, M.; and Doshi-Velez, F. 2019. Explainable reinforcement learning via reward decomposition. In *IJCAI/ECAI Workshop on explainable artificial intelligence*.

Kitamura, T.; and Yonetani, R. 2021. ShinRL: A Library for Evaluating RL Algorithms from Theoretical and Practical Perspectives.

Kraus, S.; Azaria, A.; Fiosina, J.; Greve, M.; Hazon, N.; Kolbe, L. M.; Lembcke, T.-B.; Müller, J.; Schleibaum, S.;

and Vollrath, M. 2019. AI for Explaining Decisions in Multi-Agent Environments. In *AAAI Conference on Artificial Intelligence*.

Lange, R. T. 2022. gymnax: A JAX-based Reinforcement Learning Environment Library.

Liu, X.; Yu, J.; Feng, Z.; and Gao, Y. 2020. Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing. *China Communications*, 17(9): 220–236.

Lowe, R.; Wu, Y. I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30.

Madumal, P.; Miller, T.; Sonenberg, L.; and Vetere, F. 2020. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 2493–2500.

Michalski, A.; Christianos, F.; and Albrecht, S. V. 2023. SMAClite: A Lightweight Environment for Multi-Agent Reinforcement Learning. arXiv:2305.05566.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016a. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937. PMLR.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016b. Asynchronous Methods for Deep Reinforcement Learning. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 1928–1937. New York, New York, USA: PMLR.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Nasir, Y. S.; and Guo, D. 2019. Multi-Agent Deep Reinforcement Learning for Dynamic Power Allocation in Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 37(10): 2239–2250.

Osband, I.; Doron, Y.; Hessel, M.; Aslanides, J.; Sezener, E.; Saraiva, A.; McKinney, K.; Lattimore, T.; Szepesvari, C.; Singh, S.; et al. 2019. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*.

Papoudakis, G.; Christianos, F.; Schäfer, L.; and Albrecht, S. V. 2021. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. arXiv:2006.07869.

Pretorius, A.; Cameron, S.; Van Biljon, E.; Makkink, T.; Mawjee, S.; du Plessis, J.; Shock, J.; Laterre, A.; and Beguir, K. 2020. A game-theoretic analysis of networked system control for common-pool resource management using multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 9983–9994.

Puiutta, E.; and Veith, E. M. 2020. Explainable reinforcement learning: A survey. In *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*, 77–95. Springer.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018a. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:1803.11485.

Rashid, T.; Samvelyan, M.; de Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018b. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. arXiv:1803.11485.

Samvelyan, M.; Rashid, T.; de Witt, C. S.; Farquhar, G.; Nardelli, N.; Rudner, T. G. J.; Hung, C.-M.; Torr, P. H. S.; Foerster, J.; and Whiteson, S. 2019. The StarCraft Multi-Agent Challenge.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Shapley, L. S. 1953. Stochastic Games*. *Proceedings of the National Academy of Sciences*, 39(10): 1095–1100.

Singh, S. S.; and Rosman, B. 2023. The Challenge of Redundancy on Multi-agent Value Factorisation. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2436–2438.

Spatharis, C.; Blekas, K.; Bastas, A.; Kravaris, T.; and Vouros, G. A. 2019. Collaborative multiagent reinforcement learning schemes for air traffic management. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 1–8.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; and Graepel, T. 2017a. Value-Decomposition Networks For Cooperative Multi-Agent Learning. arXiv:1706.05296.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2017b. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

Tan, M. 1997. Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents. In *International Conference on Machine Learning*.

Vidhate, D. A.; and Kulkarni, P. 2017. Cooperative multi-agent reinforcement learning models (CMRLM) for intelligent traffic control. In *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*, 325–331.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J. P.; Jaderberg, M.; Vezhnevets, A. S.; Leblond, R.; Pohlen, T.; Dalibard, V.;

Budden, D.; Sulsky, Y.; Molloy, J.; Paine, T. L.; Gulcehre, C.; Wang, Z.; Pfaff, T.; Wu, Y.; Ring, R.; Yogatama, D.; Wünsch, D.; McKinney, K.; Smith, O.; Schaul, T.; Lillicrap, T. P.; Kavukcuoglu, K.; Hassabis, D.; Apps, C.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 1–5.

Vouros, G. A. 2022. Explainable deep reinforcement learning: state of the art and challenges. *ACM Computing Surveys*, 55(5): 1–39.

Wang, J.; Zhang, Y.; Gu, Y.; and Kim, T.-K. 2022. SHAQ: Incorporating Shapley Value Theory into Multi-Agent Q-Learning. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 5941–5954. Curran Associates, Inc.

Wang, J.; Zhang, Y.; Kim, T.-K.; and Gu, Y. 2020. Shapley q-value: A local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 7285–7292.

Wen, M.; Kuba, J. G.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; and Yang, Y. 2022. Multi-Agent Reinforcement Learning is a Sequence Modeling Problem. arXiv:2205.14953.

Wolpert, D. H.; and Tumer, K. 2001. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(02n03): 265–279.

Yang, Y.; Hao, J.; Chen, G.; Tang, H.; Chen, Y.; Hu, Y.; Fan, C.; and Wei, Z. 2020a. Q-value path decomposition for deep multiagent reinforcement learning. In *International Conference on Machine Learning*, 10706–10715. PMLR.

Yang, Y.; Hao, J.; Liao, B.; Shao, K.; Chen, G.; Liu, W.; and Tang, H. 2020b. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*.

Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; and Wu, Y. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35: 24611–24624.

Zhao, N.; Liu, Z.; and Cheng, Y. 2020. Multi-Agent Deep Reinforcement Learning for Trajectory Design and Power Allocation in Multi-UAV Networks. *IEEE Access*, 8: 139670–139679.