A EXPERIMENTAL DETAILS

A.1 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.1.1, while the RWARE scenarios are explained in Section A.1.2.

A.1.1 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 a 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 centered 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 to 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 version 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.

A.1.2 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).

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 centered 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.

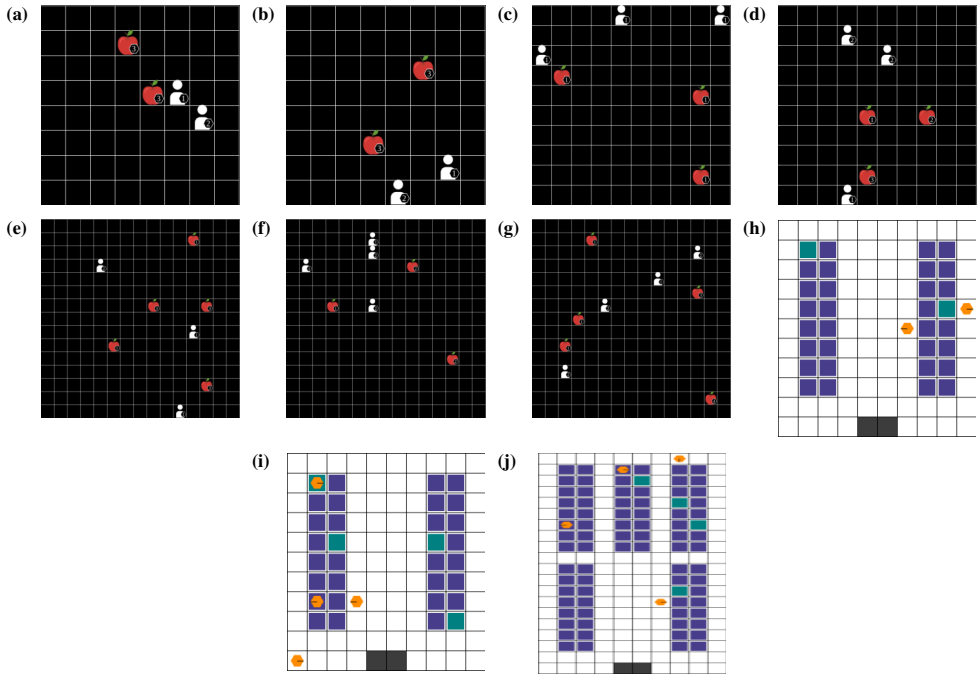


Figure 9: *Illustration of the seven LBF and three RWARE tasks used for the main experiments.*

A.2 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.

A.2.1 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.

A.2.2 Policy Gradients (PG)

Policy-Based:

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.

A.3 Evaluation Protocol

A.3.1 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.

A.3.2 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.

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
ϵ schedule steps	2e6	5e4	5e4	5e4
ϵ 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

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	0.0003	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)

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.

A.3.3 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 3: Hyperparameters for VDN 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 4: Hyperparameters for QMIX 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 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

Table 6: Hyperparameters for MAPPO with and without parameter sharing

	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 7: Hyperparameters for MAA2C 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

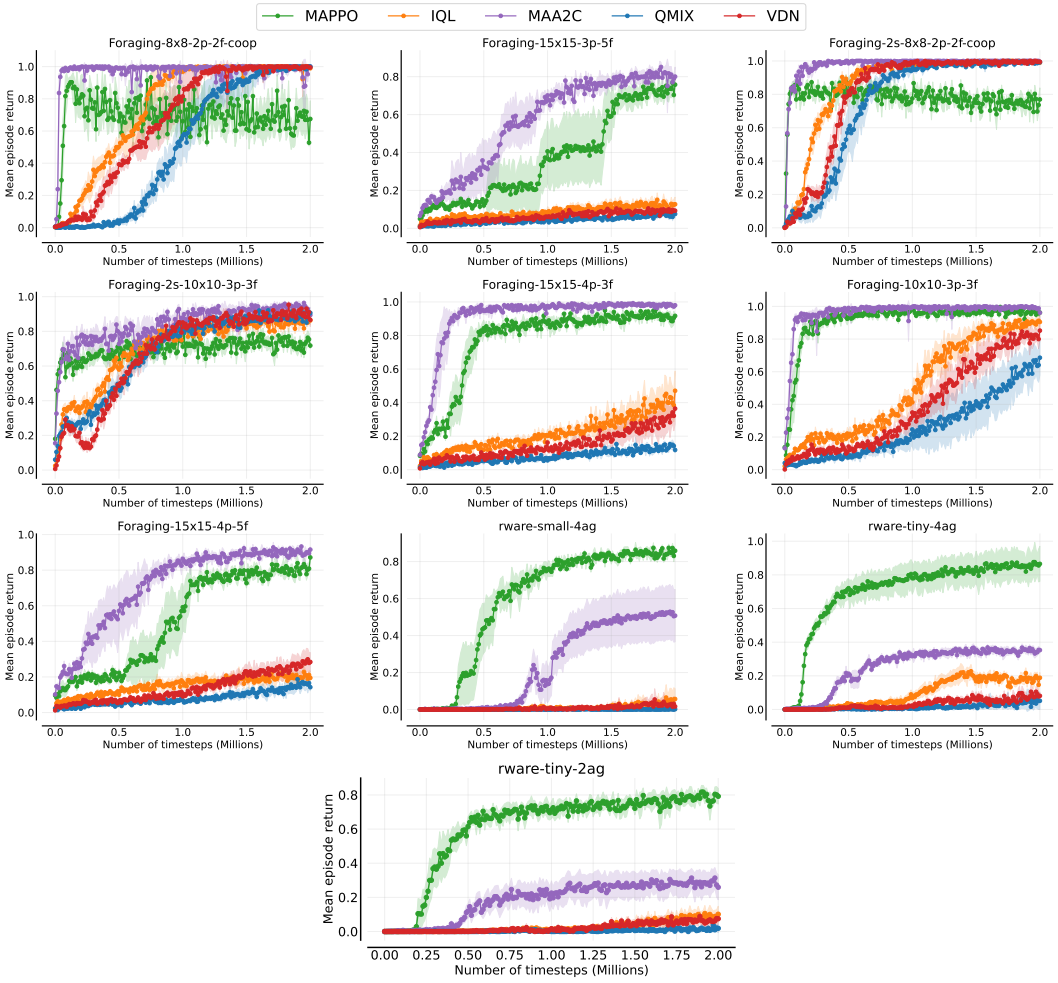


Figure 10: Mean episode returns for all algorithms with parameter sharing in seven LBF scenarios and three RWARE scenarios, with the mean and 95% confidence intervals over 10 distinct seeds.

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.

B.1 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 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

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

B.2 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.1, the results of algorithm performance in the LBF and RWARE environments are presented in Tables 8 and 9.

B.3 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

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

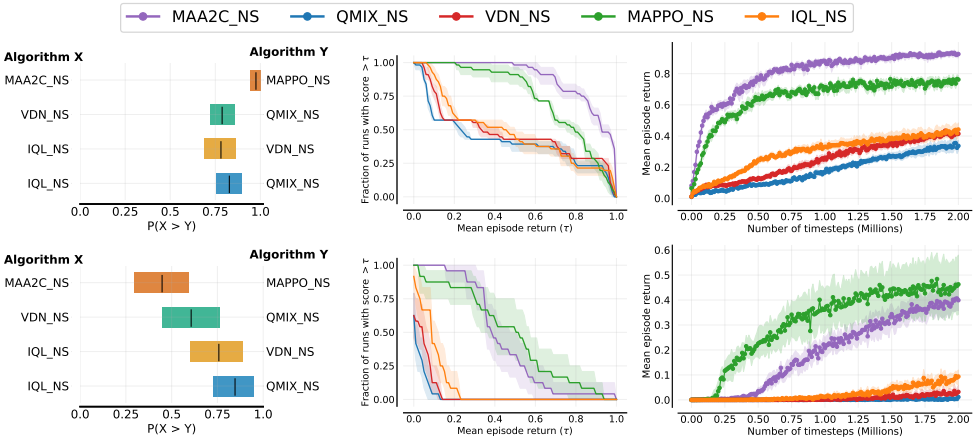


Figure 11: *Results from running the same experimental hyperparameters on the same tasks as Papoudakis et al. (2021) including probability of improvement, performance profiles and sample efficiency curves*. **Top row:** Performance of algorithms on 7 LBF tasks. **Bottom row:** Performance of all algorithms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

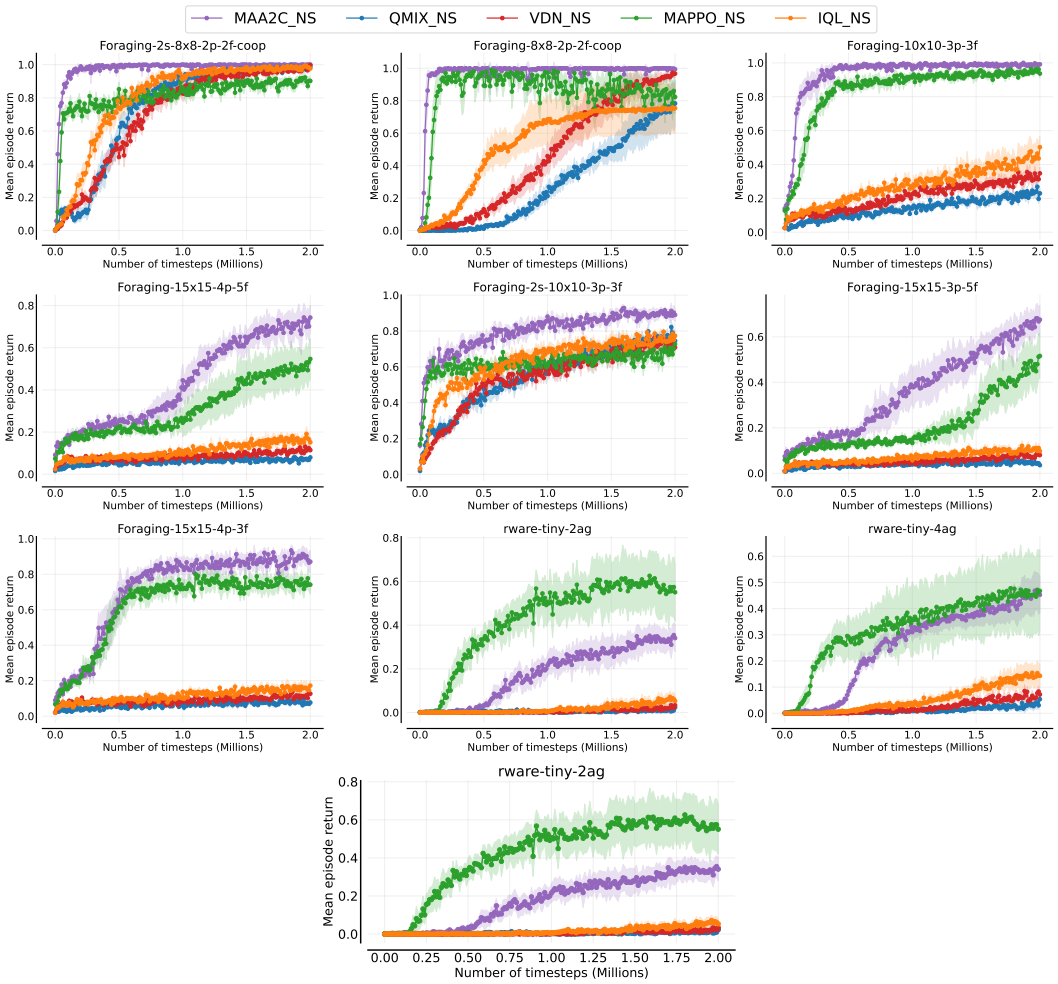


Figure 12: Mean episode returns for all algorithms without parameter sharing in seven LBF scenarios and three RWARE scenarios, with the mean and 95% confidence intervals over 10 distinct seeds.

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

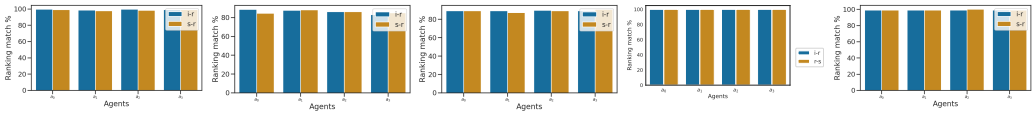


Figure 13: Comparison of Shapley value and agent importance rankings to individual rewards for IQL, MAA2C, MAPPO, QMIX and VDN on rware-small-4ag-v1.

similarly in the sparse setting when performance 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.

B.4 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 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 setting it is important to determine how effectively *agent importance* is able to determine the contributions of agents in complex heterogeneous settings where there are clear agent types with differing capabilities that compose the coalition.

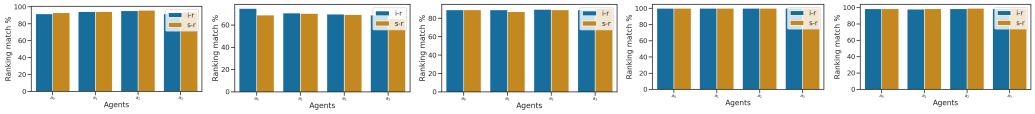
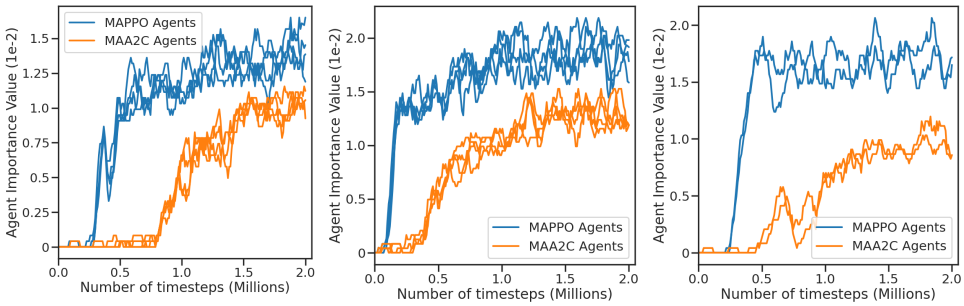
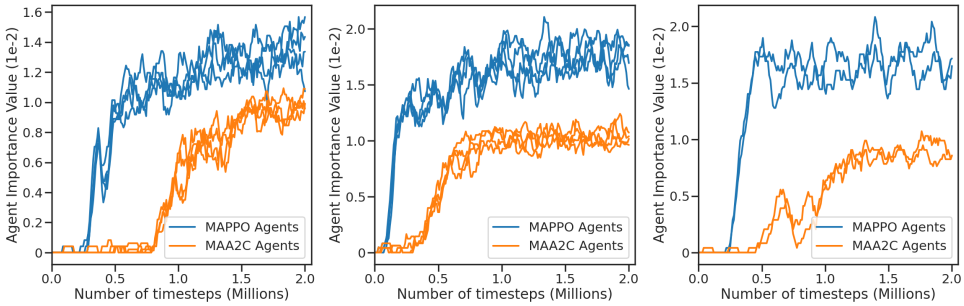


Figure 14: Comparison of Shapley value and agent importance rankings to individual rewards for IQL, MAA2C, MAPPO, QMIX and VDN on rware-tiny-4ag-v1



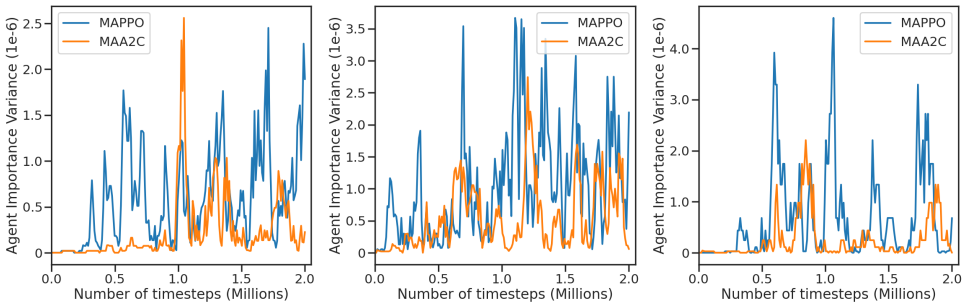
(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

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



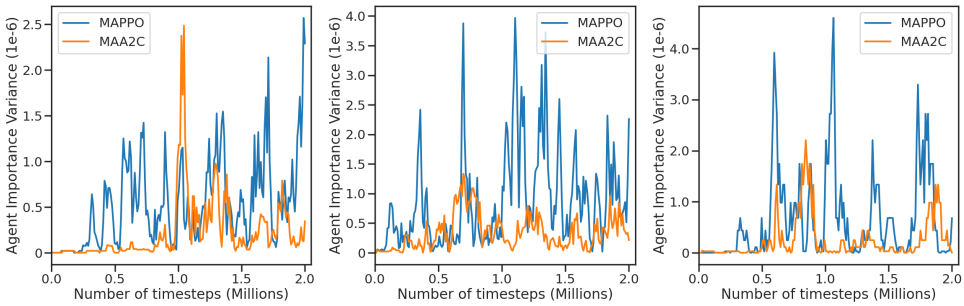
(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

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



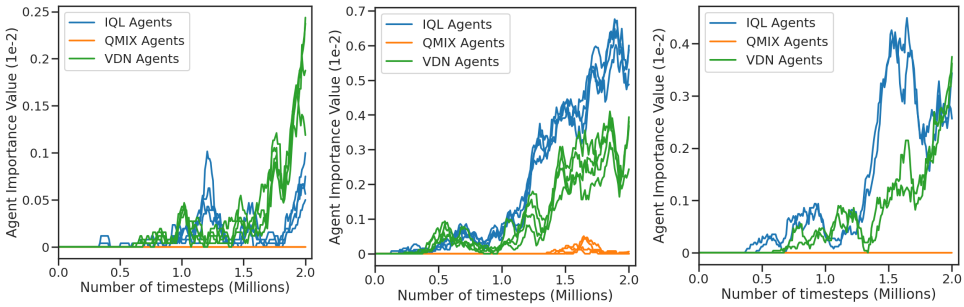
(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

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



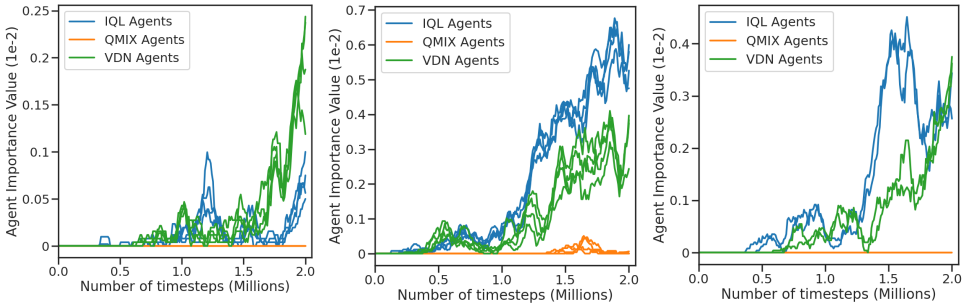
(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

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



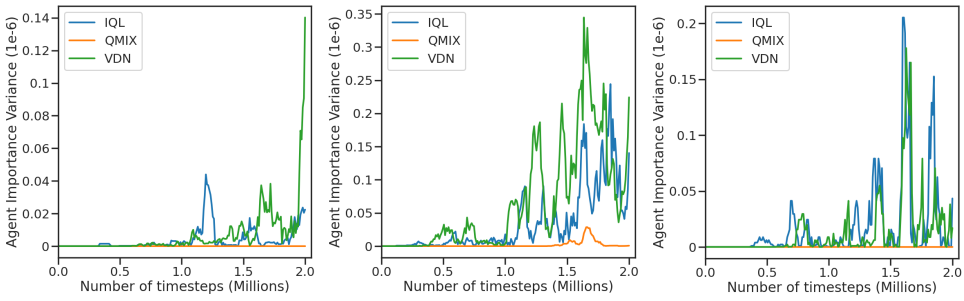
(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

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



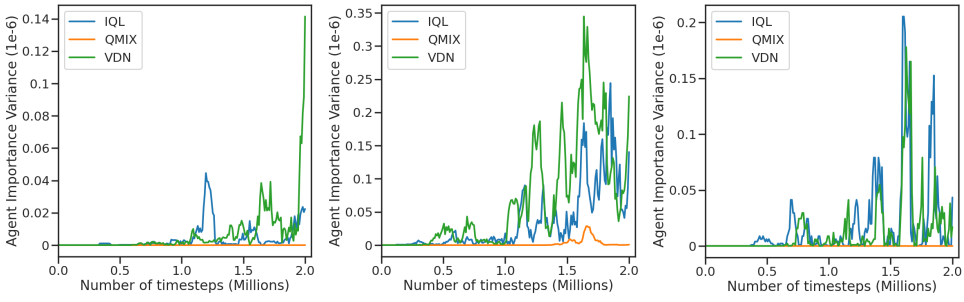
(a) rware-small-4ag-v1 (b) rware-tiny-4ag-v1 (c) rware-tiny-2ag-v1

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

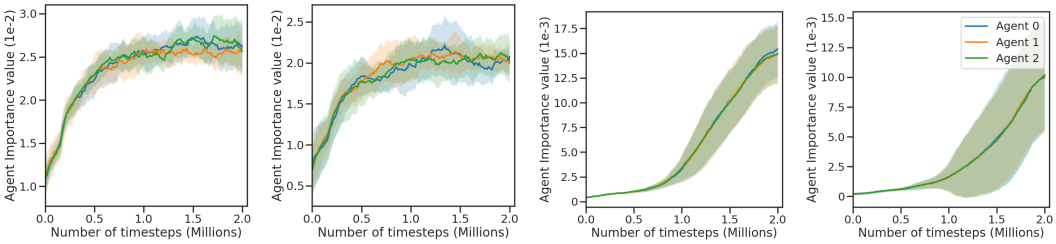


Figure 23: *Agent importance scores on the stochastic Foraging-10x10-3p-3f-v2 LBF scenario for MAA2C, MAPPO, VDN and QMIX.*

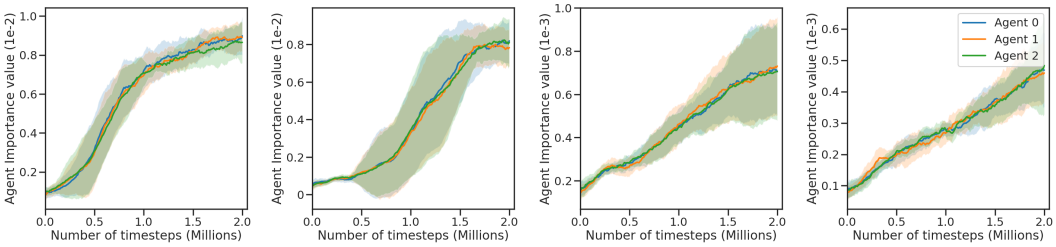
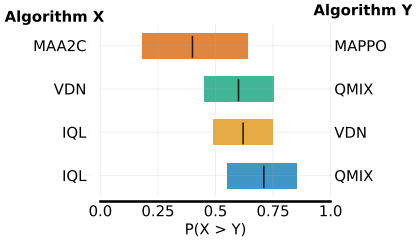
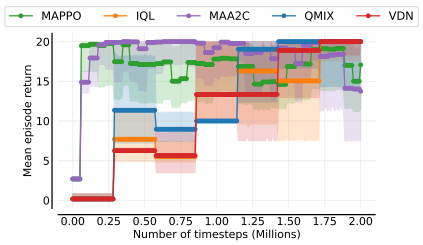


Figure 24: *Agent importance scores on the stochastic lbforaging:Foraging-15x15-3p-5f-v2 LBF scenario for MAA2C, MAPPO, VDN and QMIX.*

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 make 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 et al., 2023), which implements a setting similar to SMAC but in purely python code which allows it to be copied and reducing computational requirements.



(a) Probability of improvement without parameter sharing



(b) Sample efficiency without parameter sharing

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.

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

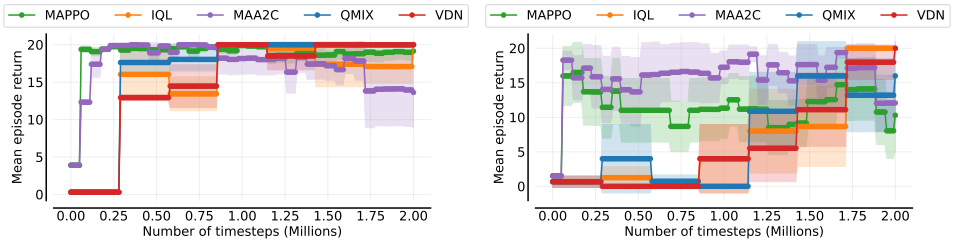


Figure 26: **Left:** Mean episode return on 2s3z. **Righ:** Mean episode return on 3m.

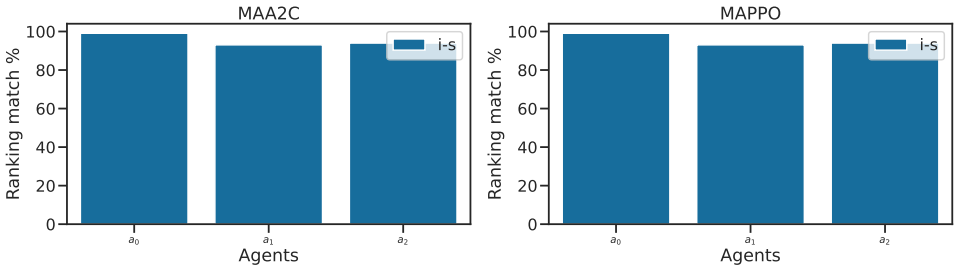


Figure 27: Ranking agreement percentages for MAPPO and MAA2C for 3m

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.

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 **??**.

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.

C.1 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)

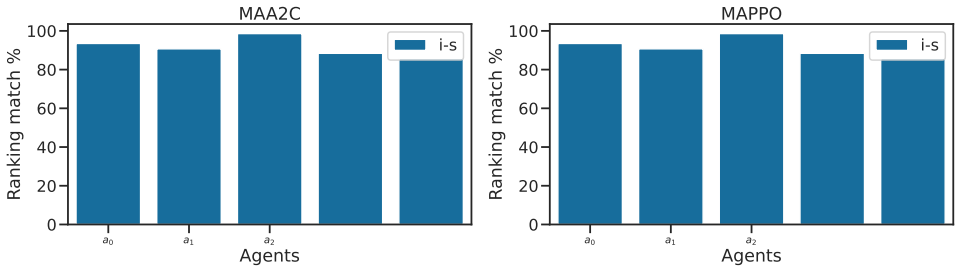


Figure 28: Ranking agreement percentages for MAPPO and MAA2C for 2s3z

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.

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.

D Further Validation of Agent Importance

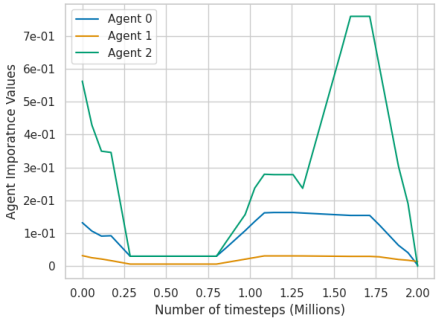
In this section, we present additional results, for further validation of the agent importance metric, thus proving its effectiveness. To do so, we set the tests on a deterministic version of a fixed scenario, within LBF, to assess the reliability of the metric and compare its scalability to the Shapley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

D.1 Metric Reliability

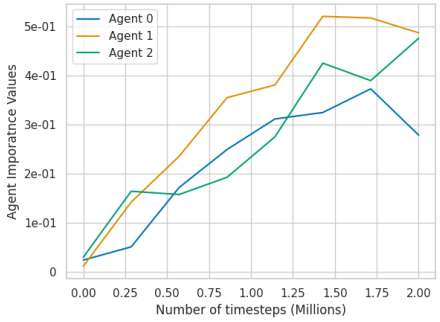
Figures 36 to 39 showcase the results for the agent importance analysis for all tested algorithms, considering both the parameter-sharing and non-parameter-sharing cases. In the deterministic LBF setting, as outlined in Section 1, agents 0, 1, and 2 are assigned levels of 1, 2, and 3, respectively. The 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.

D.2 Metric Scalability

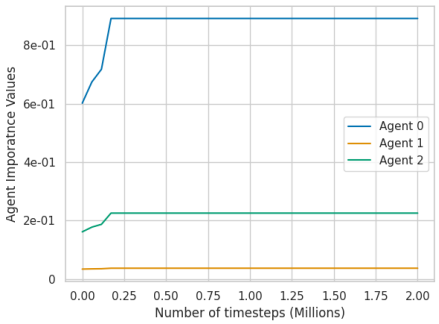
In Table 12, we present the average time taken per step along with the standard deviation for the baseline algorithm without any metrics, with agent importance, and with the Shapley Value. These values were calculated over three independent runs for each scenario. The specific scenarios used in these experiments are detailed in section A.1.1. 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.



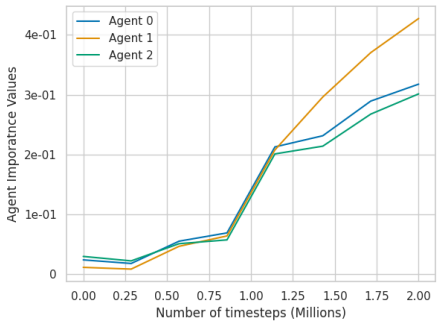
(a) MAPPO agent importance



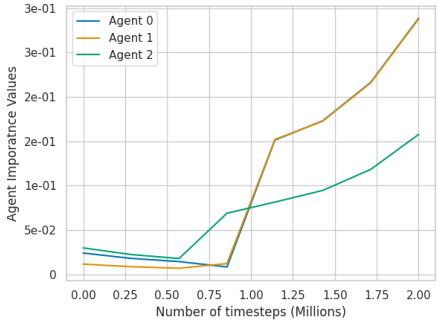
(b) IQL agent importance



(c) MAA2C agent importance



(d) QMIX agent importance



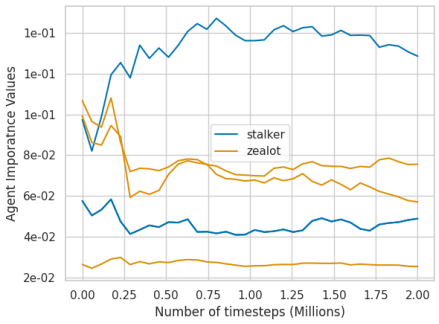
(e) VDN agent importance

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

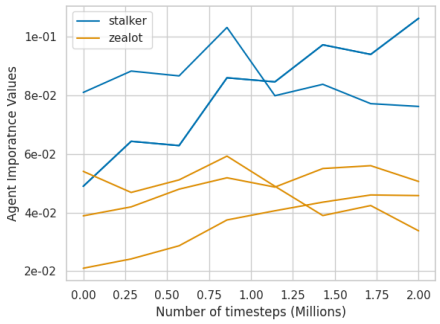
Table 12: Average time required to calculate agent contributions compared to the baseline with standard deviation

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	—

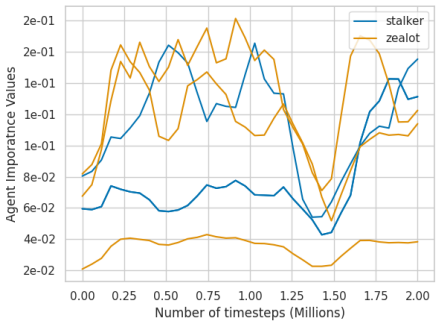
This scaling helps visualize the significant difference in the required time when using the Shapley Value.



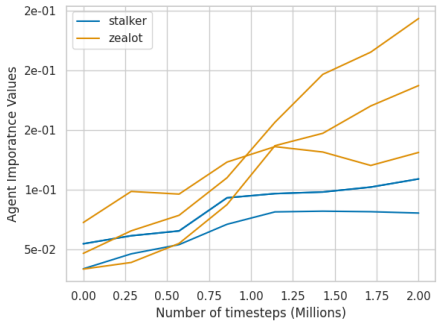
(a) MAPPO agent importance



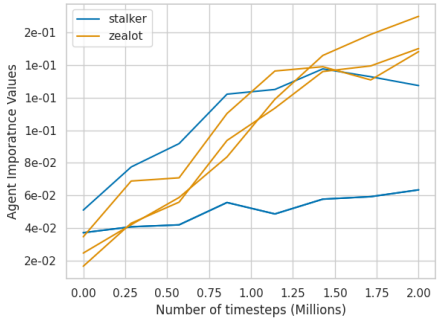
(b) IQL agent importance



(c) MAA2C agent importance



(d) QMIX agent importance



(e) VDN agent importance

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

D.3 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 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

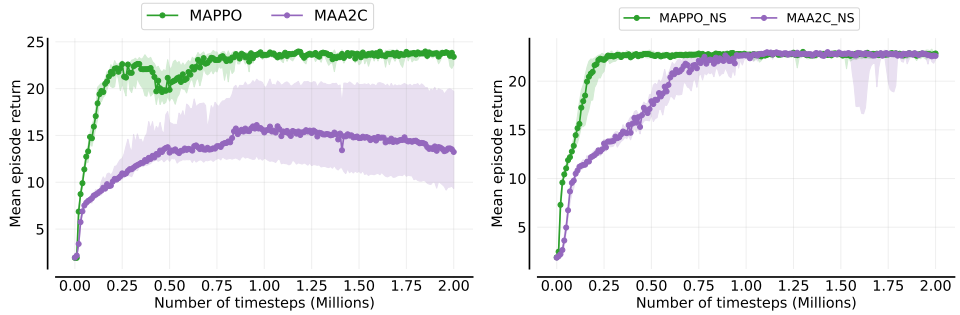


Figure 31: **Left:** Mean episode return for MMM2 with parameter sharing. **Righ:** Mean episode return for MMM2 with without parameter sharing.

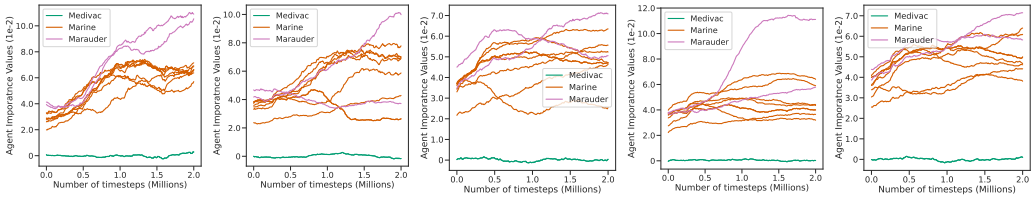


Figure 32: Agent importance plots for MAA2C with parameter sharing across 5 seeds in MMM2

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.

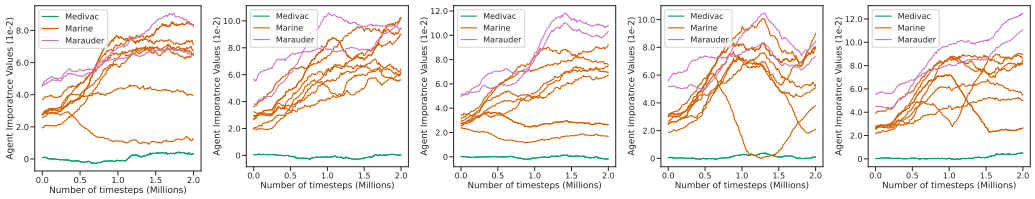


Figure 33: Agent importance plots for MAA2C without parameter sharing across 5 seeds in MMM2

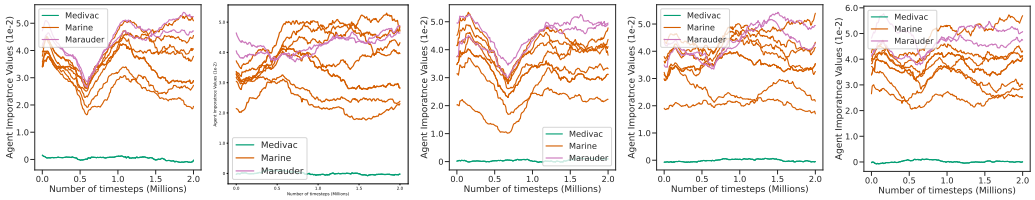


Figure 34: Agent importance plots for MAPPO with parameter sharing across 5 seeds in MMM2

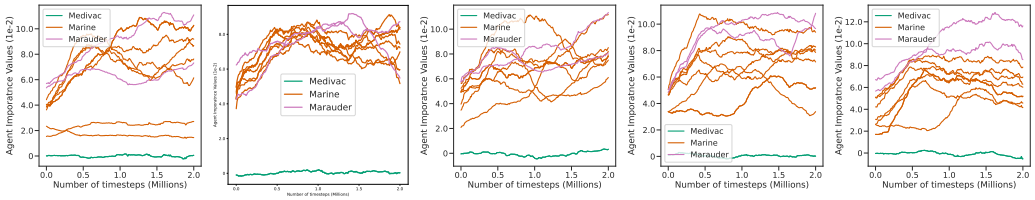


Figure 35: Agent importance plots for MAPPO without parameter sharing across 5 seeds in MMM2

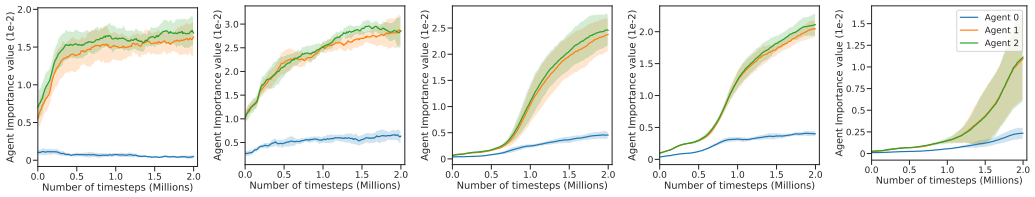


Figure 36: Agent importance scores on the **Foraging-15x15-3p-3f-det** scenario with parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

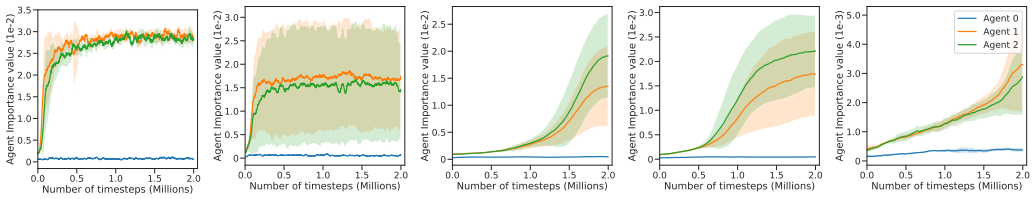


Figure 37: Agent importance scores on the **Foraging-15x15-3p-3f-det** scenario without parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

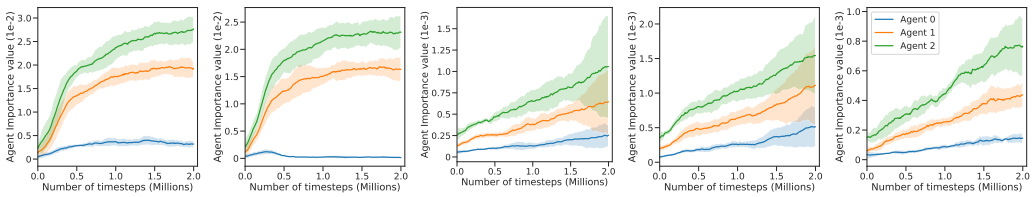


Figure 38: Agent importance scores on the **Foraging-15x15-3p-3f-det-max-food-sum** scenario with parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

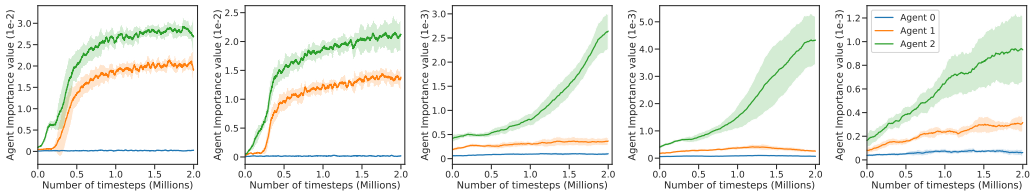


Figure 39: Agent importance scores on the **Foraging-15x15-3p-3f-det-max-food-sum** scenario without parameter sharing for MAA2C, MAPPO, VDN, IQL and QMIX. Agents 0, 1, and 2 are assigned fixed levels of 1, 2 and 3.

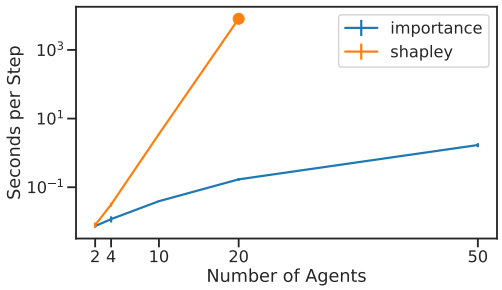


Figure 40: Computational cost of computing the Agent Importance and the Shapley value in seconds per environment step (log scale).

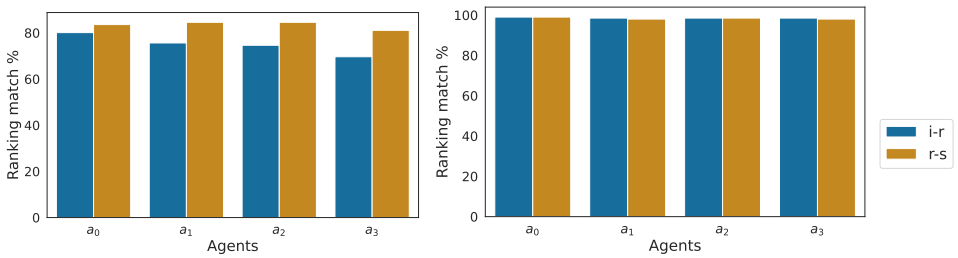


Figure 41: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

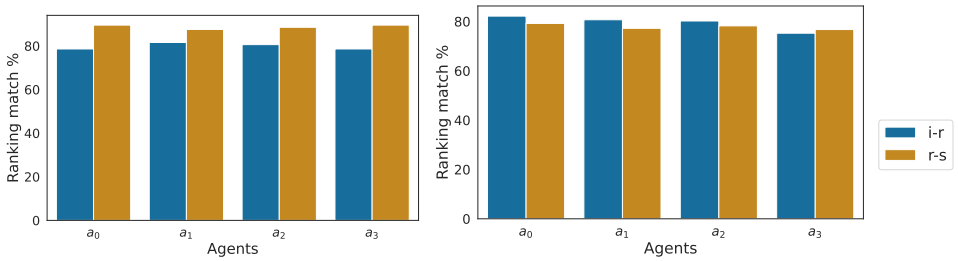


Figure 42: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

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

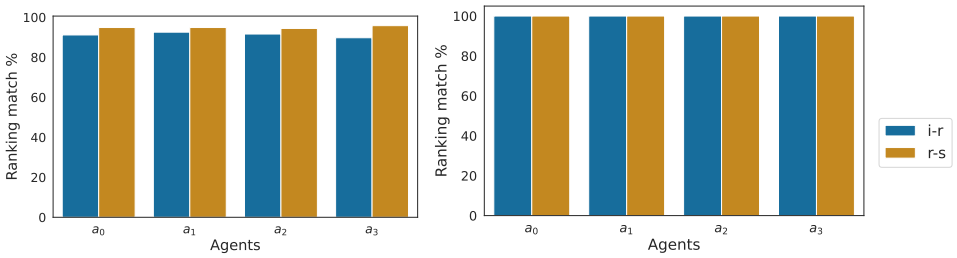


Figure 43: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

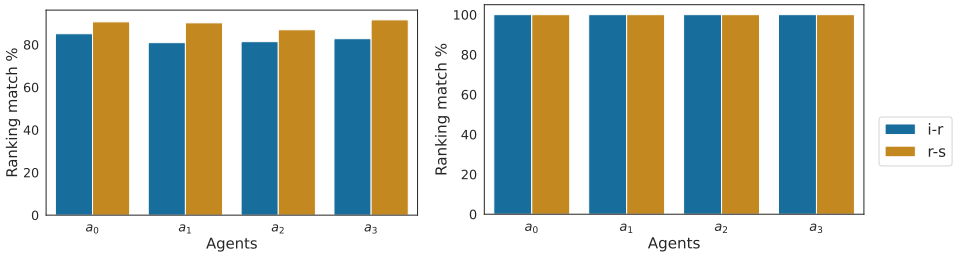


Figure 44: **Left:** Matching Rankings Comparison on LBF 15x15-4p-5f. **Right:** Matching Rankings Comparison on RWARE small-4ag.

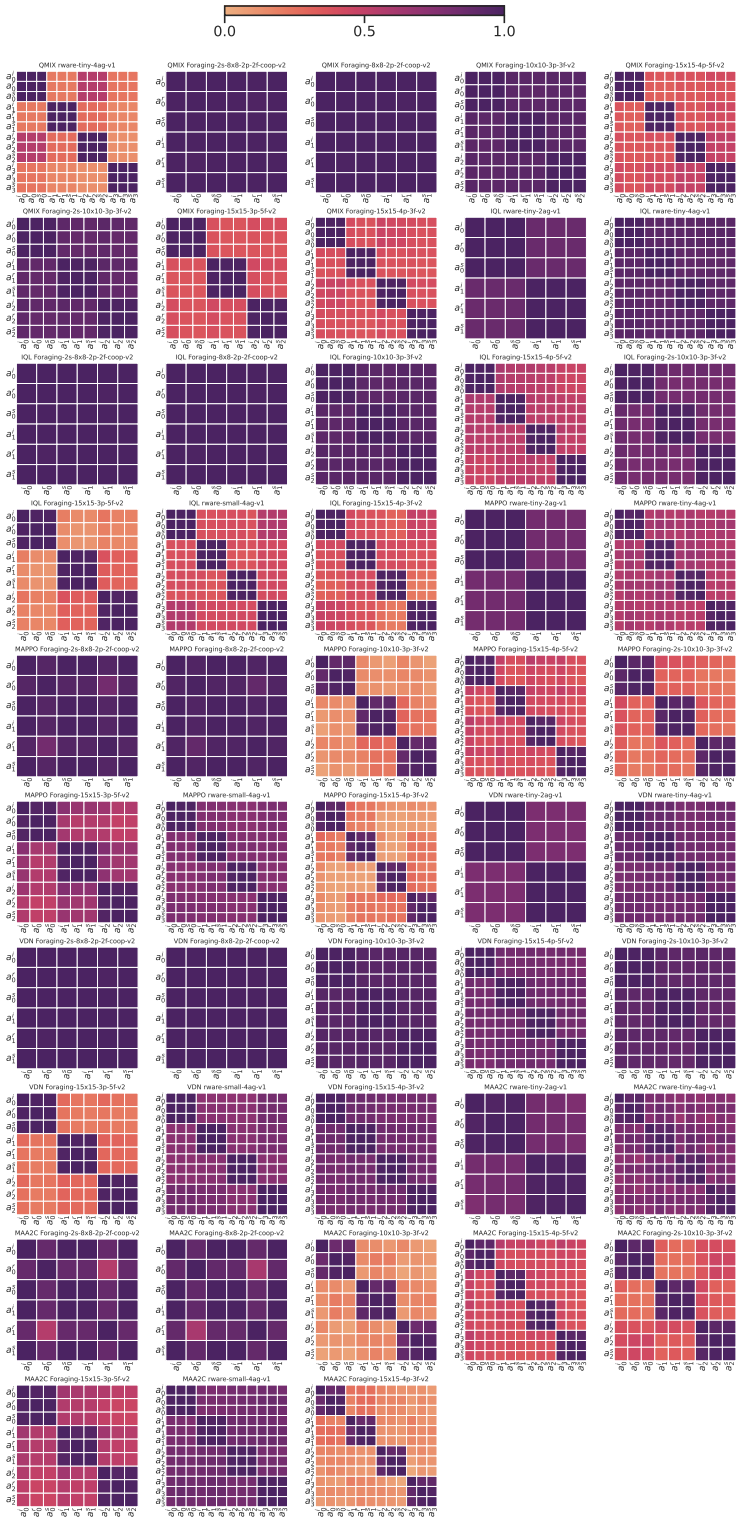


Figure 45: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the first independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

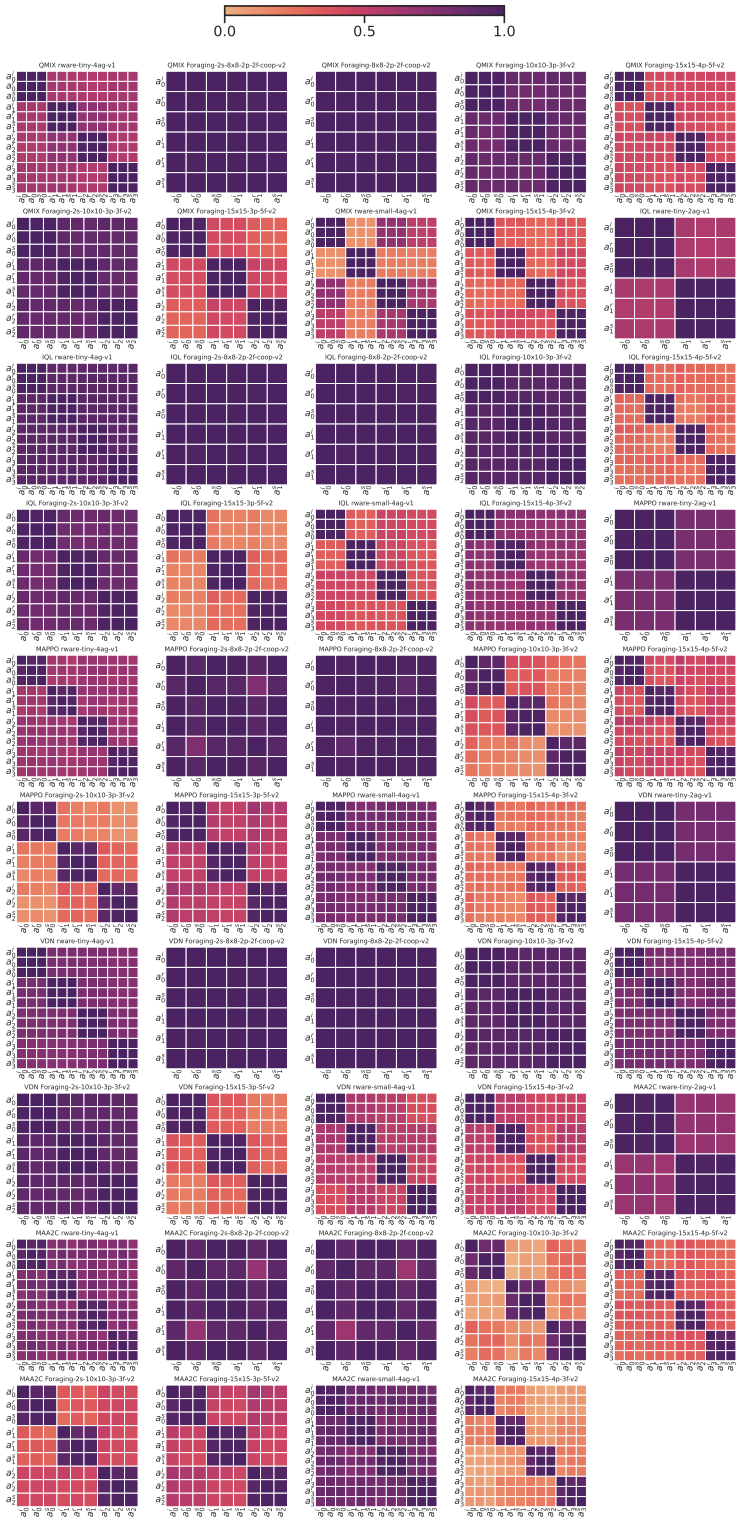


Figure 46: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the second independent run*. Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

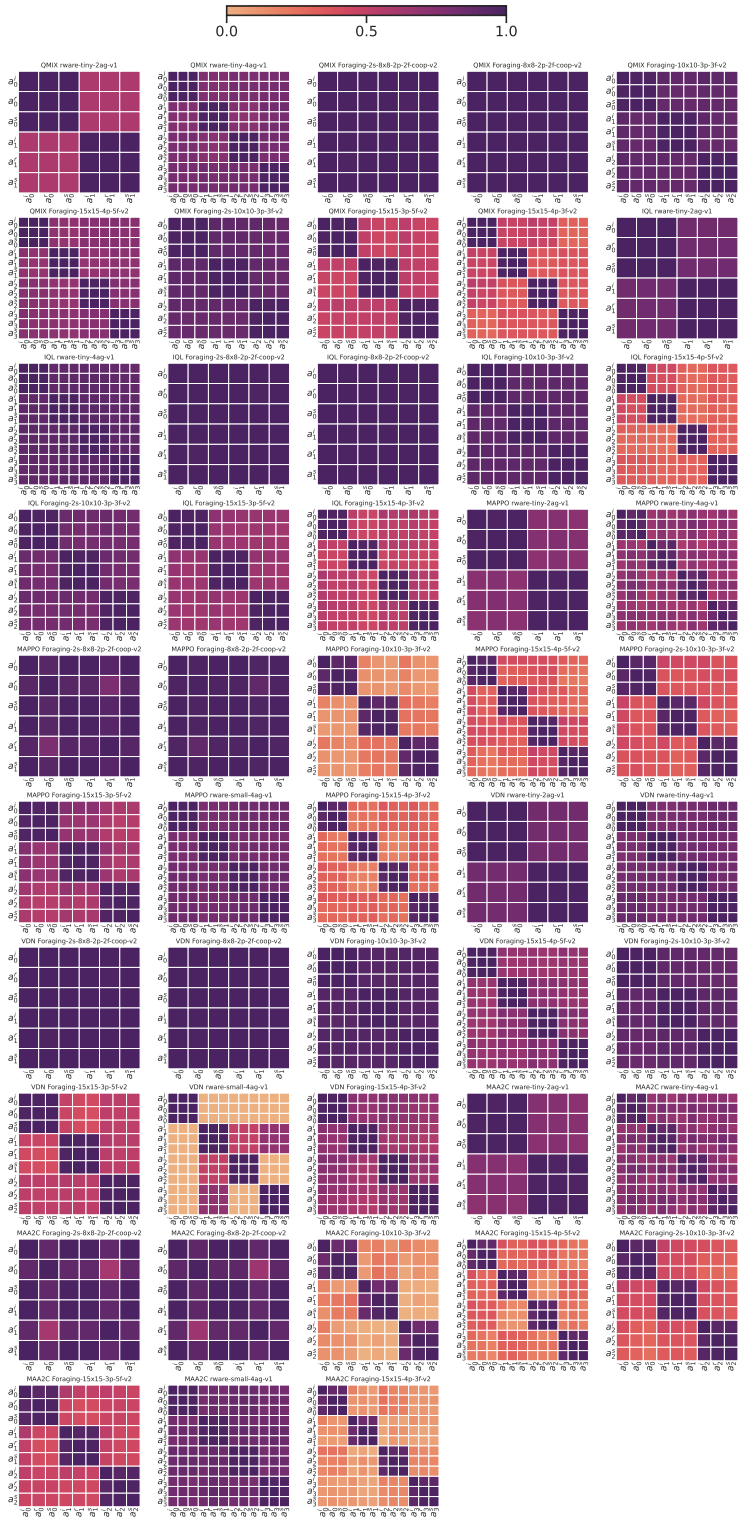


Figure 47: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the third independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

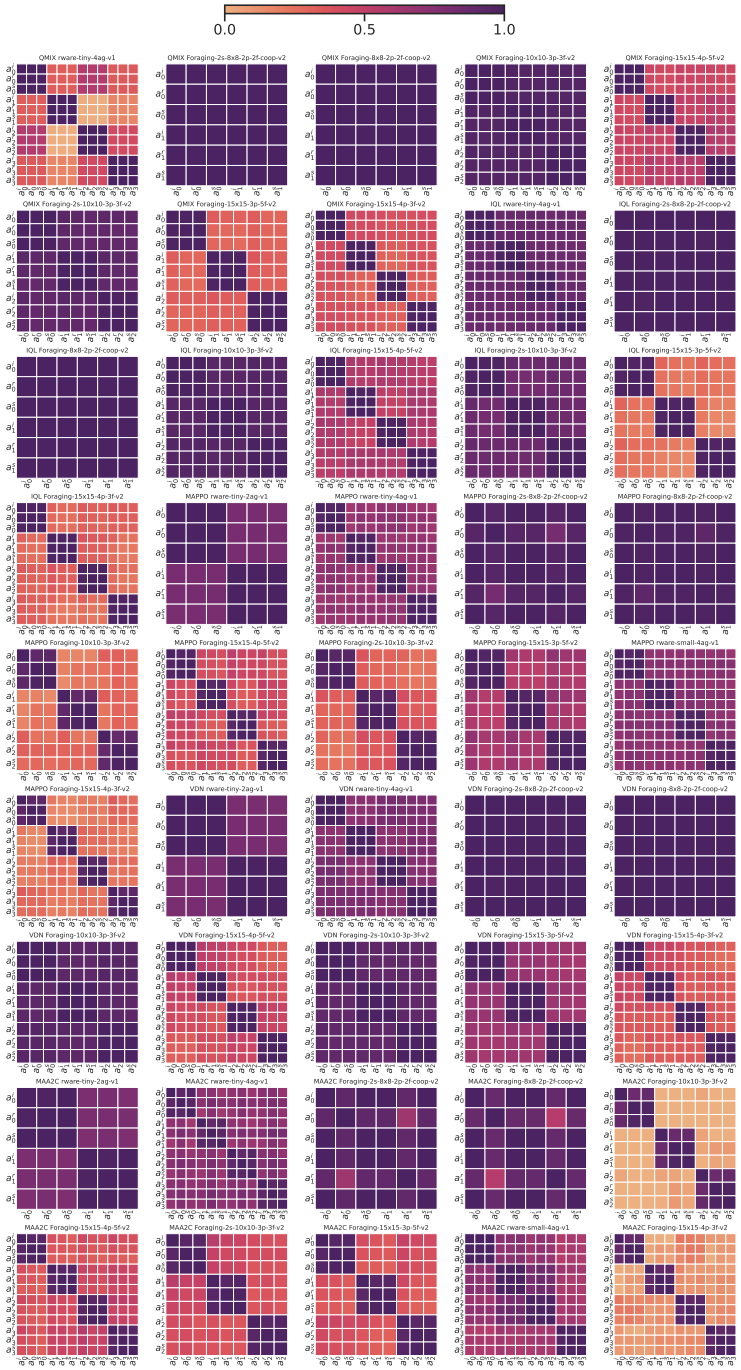


Figure 48: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the fourth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

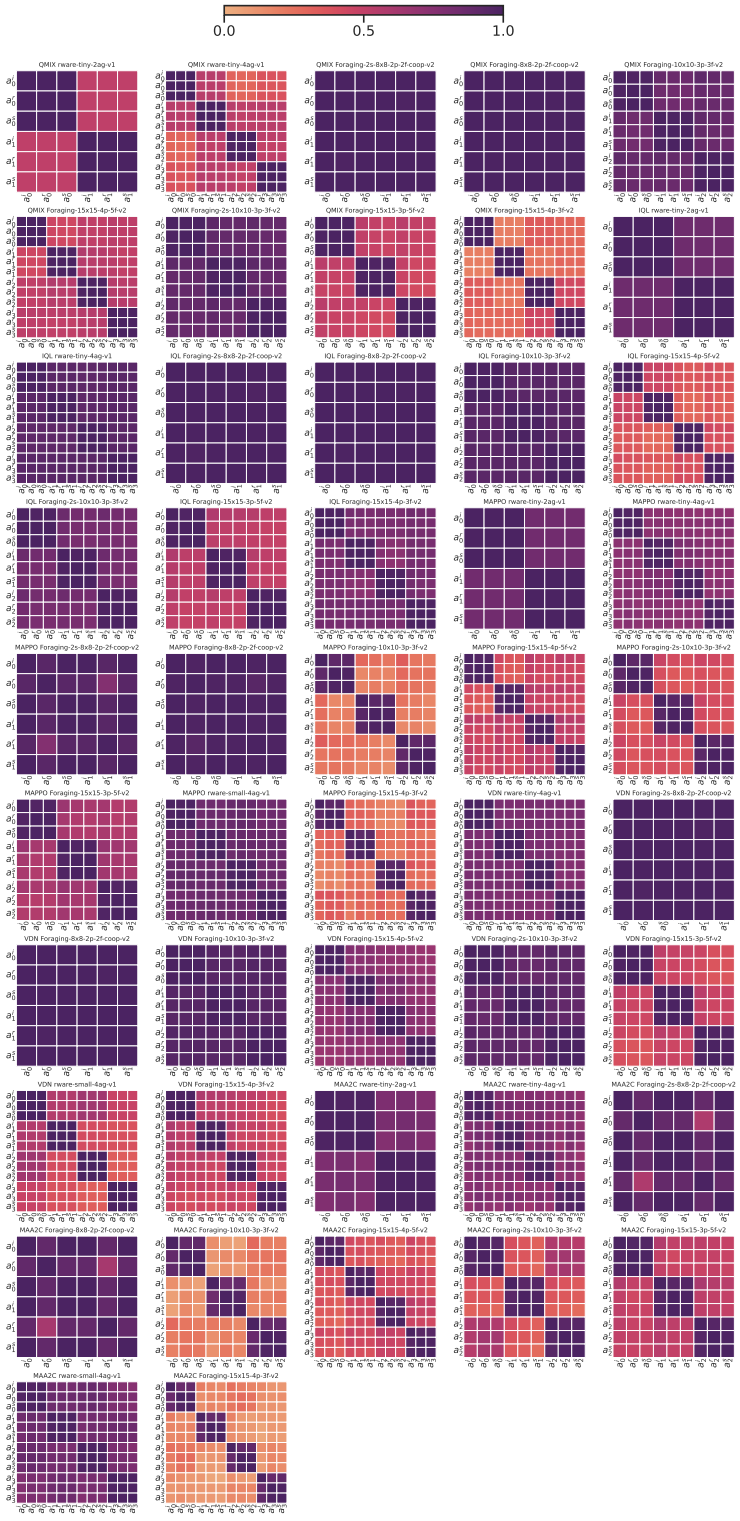


Figure 49: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the fifth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

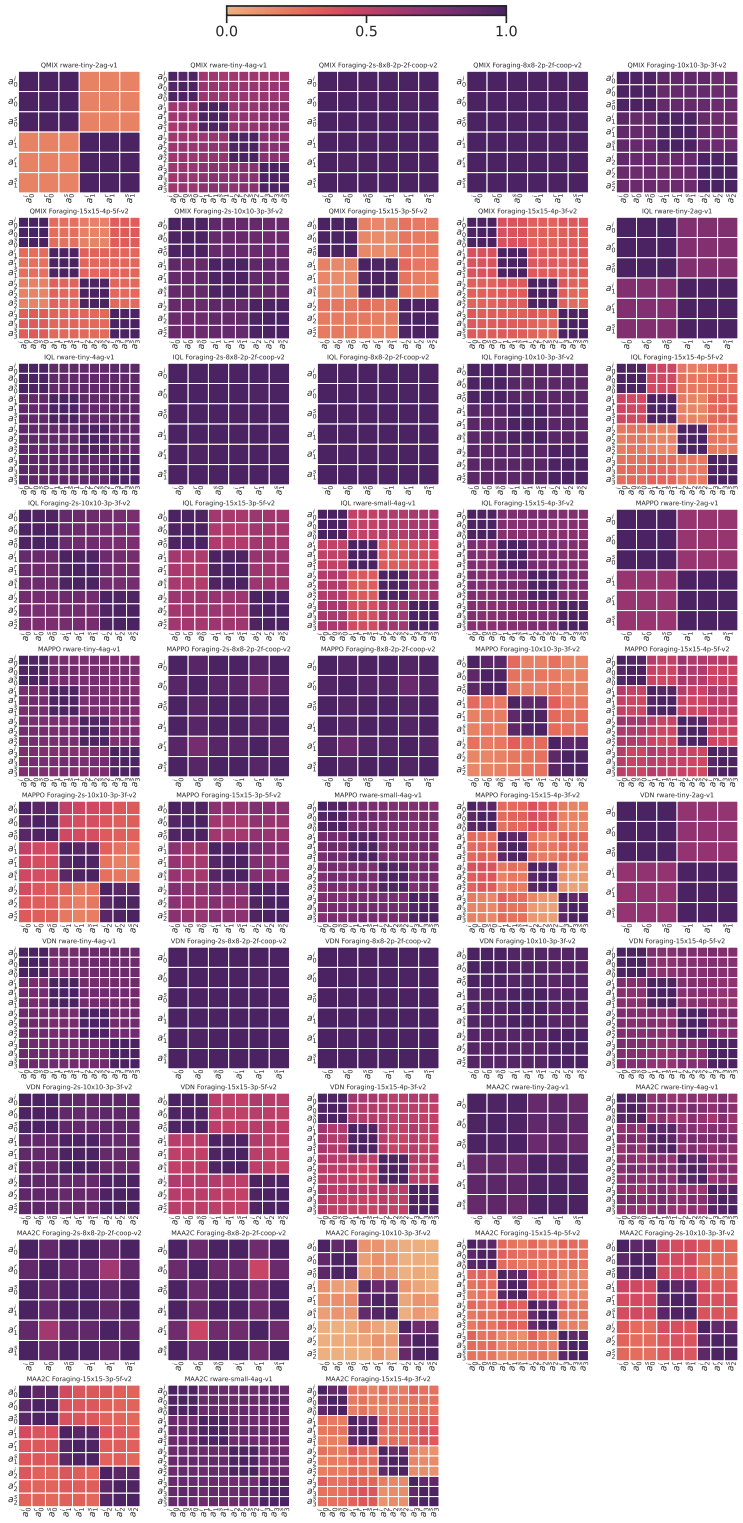


Figure 50: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the sixth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

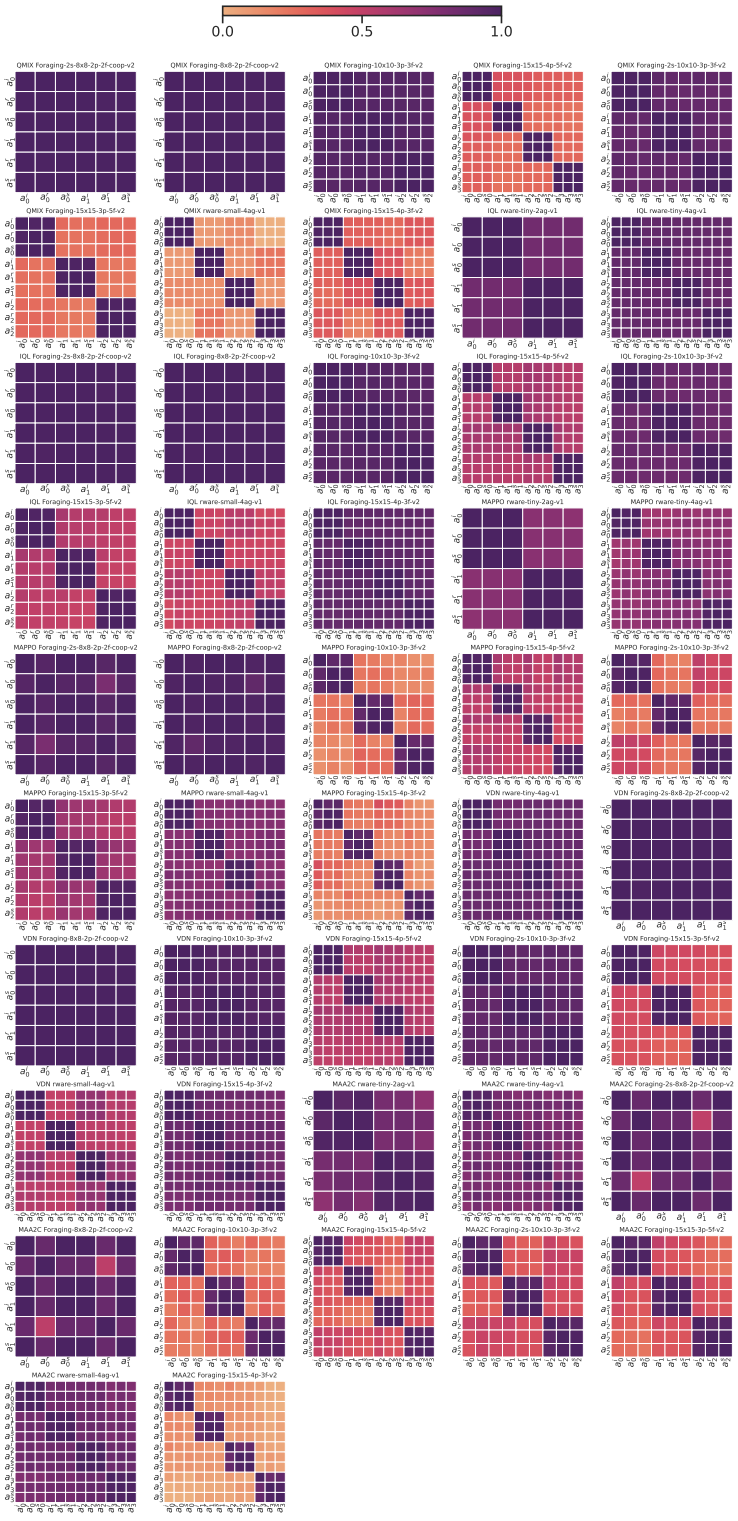


Figure 51: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the seventh independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is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

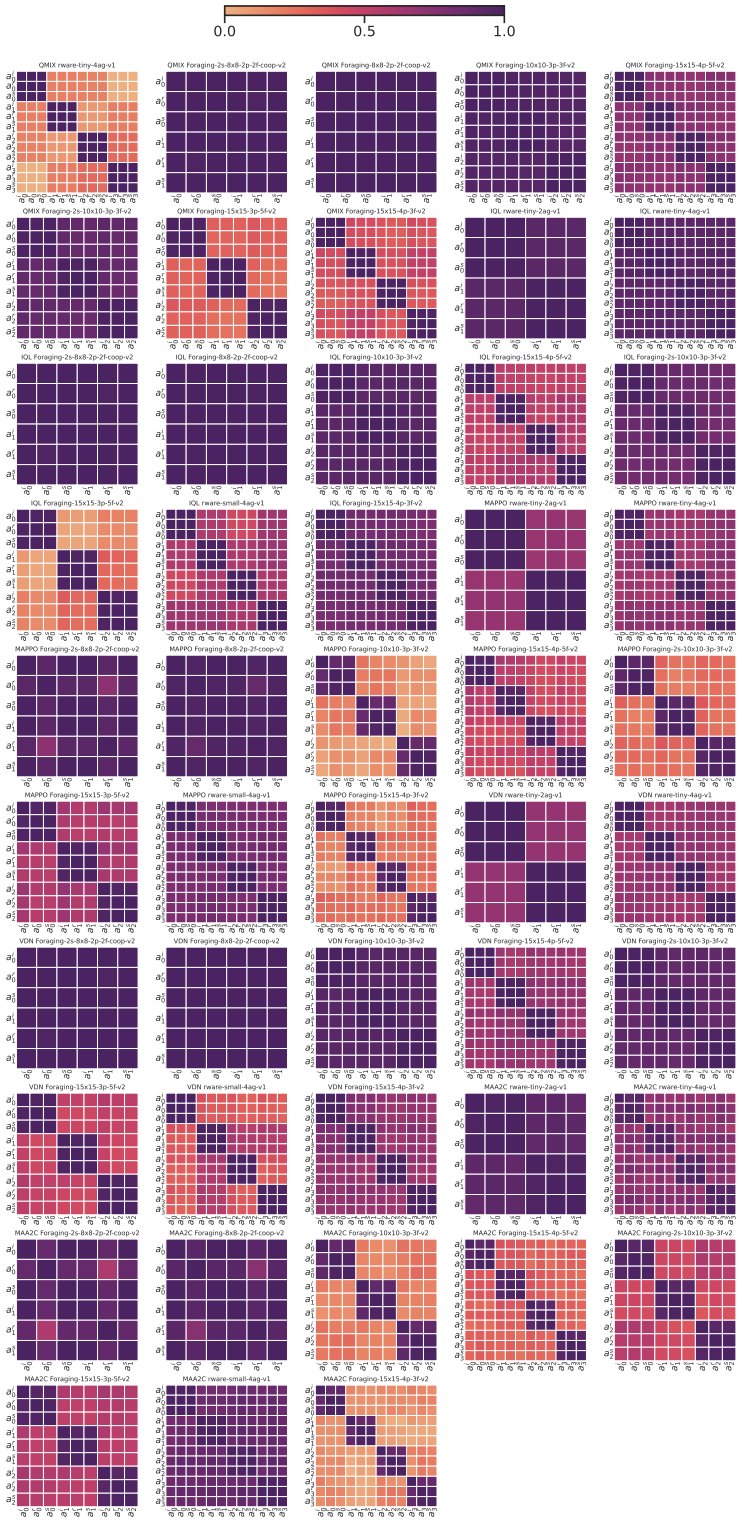


Figure 52: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the eighth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

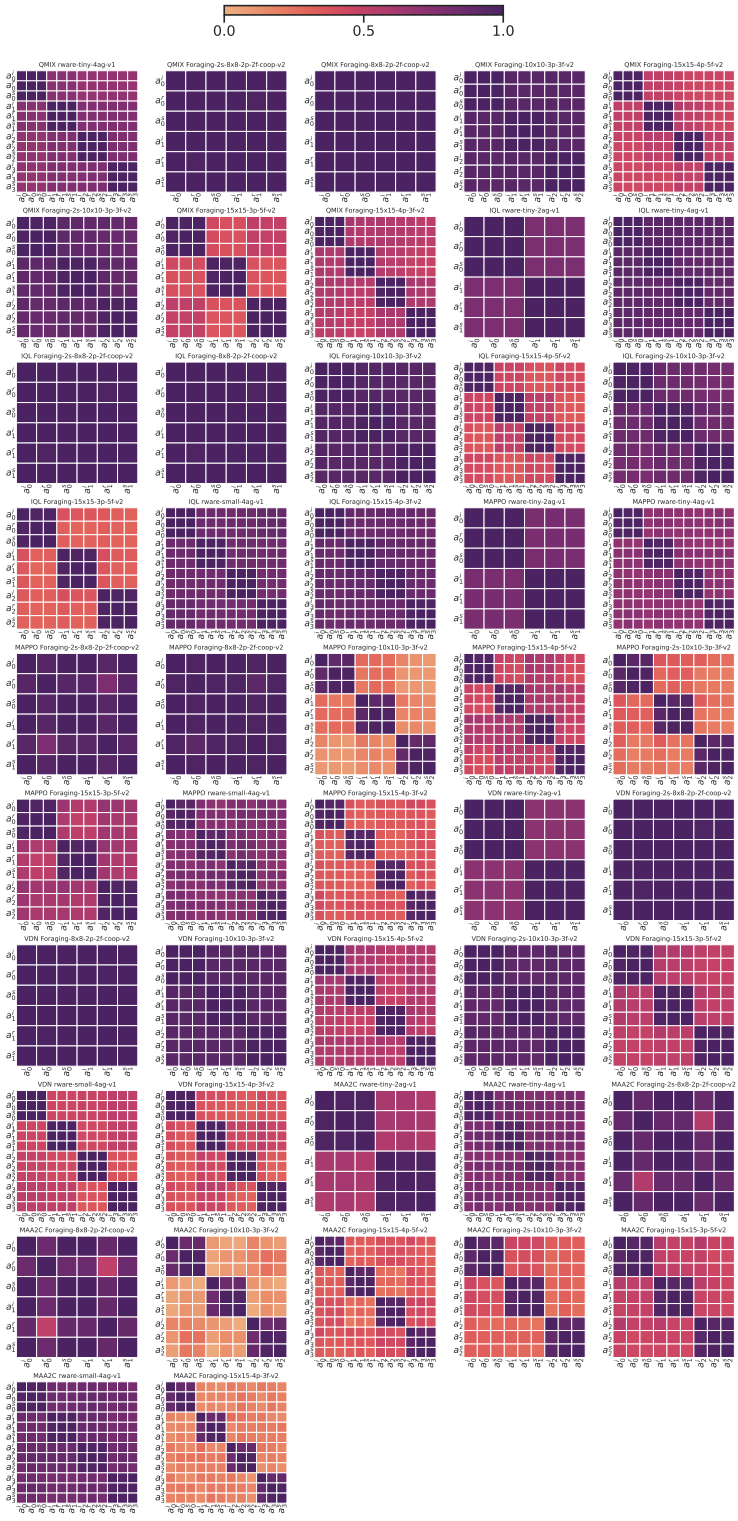


Figure 53: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the ninth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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

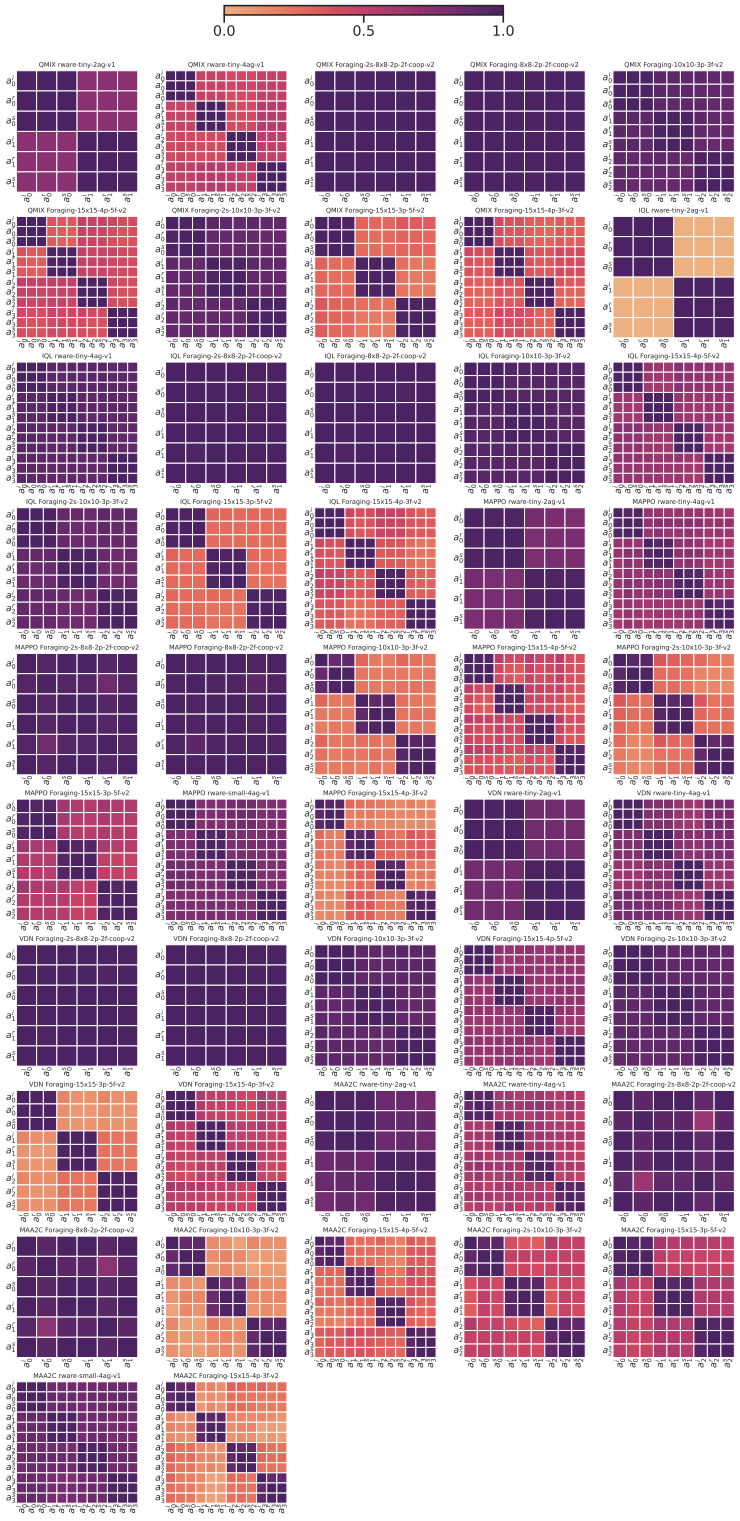


Figure 54: *The correlation between Agent Importance, Shapley values, and individual agent rewards is examined for the tenth independent run.* Each subplot corresponds to a specific algorithm and task, displaying the name of the algorithm followed by the task name. Notably, a strong correlation is observed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