Analysing factorizations of action-value networks for cooperative multi-agent reinforcement learning

Castellini, Jacopo; Oliehoek, Frans A.; Savani, Rahul; Whiteson, Shimon

DOI
10.1007/s10458-021-09506-w

Publication date
2021

Document Version
Final published version

Published in
Autonomous Agents and Multi-Agent Systems

Citation (APA)
Castellini, J., Oliehoek, F. A., Savani, R., & Whiteson, S. (2021). Analysing factorizations of action-value networks for cooperative multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, *35*(2), 1-53. Article 25. https://doi.org/10.1007/s10458-021-09506-w

Autonomous Agents and Multi-Agent Systems (2021) 35:25
https://doi.org/10.1007/s10458-021-09506-w

Analysing factorizations of action-value networks for cooperative multi-agent reinforcement learning

Jacopo Castellini[1] · Frans A. Oliehoek[2] · Rahul Savani[1] · Shimon Whiteson[3]

Accepted: 10 May 2021

Abstract
Recent years have seen the application of deep reinforcement learning techniques to cooperative multi-agent systems, with great empirical success. However, given the lack of theoretical insight, it remains unclear what the employed neural networks are learning, or how we should enhance their learning power to address the problems on which they fail. In this work, we empirically investigate the learning power of various network architectures on a series of one-shot games. Despite their simplicity, these games capture many of the crucial problems that arise in the multi-agent setting, such as an exponential number of joint actions or the lack of an explicit coordination mechanism. Our results extend those in Castellini et al. (Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'19.International Foundation for Autonomous Agents and Multiagent Systems, pp 1862–1864, 2019) and quantify how well various approaches can represent the requisite value functions, and help us identify the reasons that can impede good performance, like sparsity of the values or too tight coordination requirements.

Keywords Multi-agent systems · Neural networks · Decision-making · Action-value representation · One-shot games

✉ Jacopo Castellini
J.Castellini@liverpool.ac.uk

Frans A. Oliehoek
F.A.Oliehoek@tudelft.nl

Rahul Savani
rahul.savani@liverpool.ac.uk

Shimon Whiteson
shimon.whiteson@cs.ox.ac.uk

[1] Department of Computer Science, University of Liverpool, Liverpool, UK

[2] Interactive Intelligence Group, Delft University of Technology, Delft, The Netherlands

[3] Department of Computer Science, University of Oxford, Oxford, UK

Published online: 07 June 2021

1 Introduction

Multi-agent reinforcement learning (MARL) uses reinforcement learning to train multiple agents for such systems, and can lead to flexible and robust solutions [3, 10, 23, 43, 47]. In recent years, a variety of *deep* MARL approaches have been developed and successfully applied [7, 26, 41]. While these approaches have shown good results, there is a general lack of theoretical insight, and often it remains unclear what the neural networks used by these approaches are learning, or how we should enhance their learning power to address the problems on which they fail.

Single-agent value-based reinforcement learning methods use (deep) neural networks to represent the discrete action-value function $Q(s, a;\theta)$ to select actions directly [29] or as a 'critic' in an actor-critic scheme [24, 30]. A straightforward way to extend such methods to the multi-agent setting is by simply replacing the action by the joint action $\langle a_1, \dots, a_n \rangle$ of all agents $Q(s, \langle a_1, \dots, a_n \rangle;\theta)$. However, this approach heavily relies on the function approximation abilities of the neural network, since it must generalize across a discrete action space whose size is exponential in the number of agents. Moreover, selecting a joint action that maximizes the Q-function usually requires that, as in deep Q-networks [29], the (now joint) actions are output nodes of the network. As a result, the computational and sample costs scale poorly in the number of agents.

Another approach to extend single-agent reinforcement learning methods to multi-agent systems is to apply them to each agent independently [42]. This improves scalability at the expense of quality, e.g., independent deep Q-learners may not be able to accurately represent the value of coordination, as every agent learns on its own and ignores the others. Furthermore, the environment becomes non-stationary from the perspective of a single agent due to the other agents' simultaneous learning and thus their learning process may not converge [6, 42, 46].

A middle ground is to learn a *factored Q-value function* [12, 15], which represents the joint value but decomposes it as the sum of a number of local components, each involving only a subset of the agents. Compared to independent learning, a factored approach can better represent the value of coordination and eases the non-stationarity problem introduced by the other agents' changing policies. Compared to a naive joint approach, it has better scalability in the number of agents, thus reducing the number of samples required to learn a correct value function. Recently, factored approaches have shown success in deep MARL [35, 38, 40], although an in-depth analysis of the learned representations is not provided and the methods are mainly restricted to agent-wise factorizations.

In this paper, we focus on centralized learning of value-based MARL approaches for cooperative multi-agent systems. Although the usual focus of MARL is on decentralized execution [9, 27, 35, 40], proper centralized learning is still useful in cases in which centralized execution is available or as a mean to inform decentralized agents under the centralized training-decentralized execution (CTDE) learning framework [22], for example when learning a centralized critic to inform decentralized policies [9] or in methods that involve bootstrapping [44] or message passing schemes [20]. We critically extend the work in [4] in examining the learning capacity of these various approaches by studying the accuracy of the learned Q-function approximations $\hat{Q}$. Our main contribution consists of a wide set of diverse experiments across different axes, and an in-depth analysis of the results to assess the potential benefits of these methods, as well as their possible drawbacks. Amongst the others, we aim at investigating the following points:

– How do factored methods compare to baseline algorithms,
– Impact of factors size on the learned representations,
– Scalability to larger systems,
– Sample efficiency,
– Effects of using an exploratory policy.

Furthermore, we extend the scope of this investigation beyond agent-wise factorizations usually investigated by deep MARL methods, showing the great advantages these "higher-order" factorization can bring both in terms of learning accuracy and sample efficiency, even when only few agents are comprised into each factor. To minimise confounding factors, we focus on one-shot (i.e., non-sequential) problems [33] and a stationary uniform sampling of the actions. Specifically, we investigate the learning power of various network architectures on a series of one-shot games that require a high level of coordination. Some of these games have an underlying factored structure (that we do not assume to be known in advance) and some do not. Despite their simplicity, these games capture many of the crucial problems that arise in the multi-agent setting, such as an exponential number of joint actions. As our results show, factored methods prove extremely effective on a variety of such games, achieving correct reconstruction even on those games that do not present a true underlying factored structure, and outperforms both independent learners and joint approaches in terms of learning speed. These benefits are even more apparent when the size of such systems grows larger, and a completely centralized solution proves impractical or even infeasible. Thus, an empirical evaluation to assess the accuracy of various representations in one-shot problems is key to understanding and improving deep MARL techniques, and our takeaways can help the community in taking informed decisions when developing solutions for multi-agent systems. We also discuss additional links to standard sequential MARL in Sect. 5, as well as depicting some possible future directions to further clarify our understanding of action-value functions in this setting.

2 Background

In the following we provide some basilar notions required to understand the remainder of this work.

2.1 One-shot games

Definition 1 *One-shot games*: a one-shot game [33] consists of the tuple

$$\mathcal{M} = \langle D, \{A_i\}_{i=1}^n, \{Q_i\}_{i=1}^n \rangle,$$

where $D = \{1, \dots, n\}$ is the set of agents, A_i is the set of actions for agent i, and Q_i is the reward function for agent i that depends only on the joint action $a \in A = \times_{i=1}^n A_i$ performed by the full team of agents, which expresses how much reward agent i gets from the overall team decision[1].

[1] We write $Q_i(a)$ for the reward function in the one-shot problem to make the link with sequential MARL more apparent.

Definition 2 *Cooperative one-shot game*: a cooperative one-shot game is a game in which all agents share the same reward function $Q(a)$, so that the goal of the team is to maximize this shared reward by finding the optimal joint action $a \in A$ to perform.

In this work, we focus on cooperative games. Our work aims at investigating the representations of the action-value function obtained with various neural network approaches and how close these are to the original one. Although we do not explicitly rely on any of the cooperative properties of these settings, and thus we could in principle extend our analysis to cooperative and mixed scenarios as well, we think that modelling agents with opposing or mixed interests into the same factor component would not make sense from a logical perspective, as these agents may not gain any benefit from sharing their own information locally with the other agents into the same factor.

Problem Statement: Given the original action-value function $Q(a)$ and a learned representation $\hat{Q}(a)$, we are interested in investigating the quality of this learned representation, both in terms of action ranking, i.e.,

$$\sigma(\mathfrak{R}(Q), \mathfrak{R}(\hat{Q})),$$

where σ is a similarity measure and $\mathfrak{R}$ is a partial ordering of the joint actions according to their action-values, so that the learned function can reliably be used for decision making; and in terms of reconstruction error of the representation, computed using the mean squared error (MSE):

$$MSE = \frac{1}{|A|} \sum_{a \in A} (Q(a) - \hat{Q}(a))^2.$$

A related, but different, setting is that of repeated games [33], in which agents repeatedly play a one-shot game over time, and can condition their strategies on the history of play in earlier rounds. In this work, we do *not* investigate learning strategies for repeated play, but only for the one-shot game in isolation. Also related is the multi-agent bandit problem [37], in which a team of agents has to agree on which arm to choose in a classical multi-armed bandit problem. The main difference with our setting is that we are not interested in considering the regret during learning, but only in learning good approximations as close as possible to the original action-value function.

2.2 Coordination graphs

In many problems, the decision of an agent is directly influenced by the decisions of only a small subset of other agents [12]. The structure of the interactions between the agents can be represented with a (hyper-) graph called a *coordination graph* [14, 20]. A coordination graph has a node for each agent in the team and (hyper) edges $\mathcal{E}$ connecting agents in the same subset, called a *factor*. Figure 1 shows some example coordination graphs. This locality of interaction means the joint action-value function $Q(a)$ can be represented as the sum of smaller reward functions, one for each factor $e \in \mathcal{E}$:

$$Q(a) = \sum_{e \in \mathcal{E}} Q_e(a_e), \tag{1}$$

where $\mathcal{E}$ is the set of these factors and $a_e = \langle a_i \rangle_{i \in e}$ is the *local joint action* of the agents that participate in factor e. Coordination graphs are a useful instrument to represent interactions

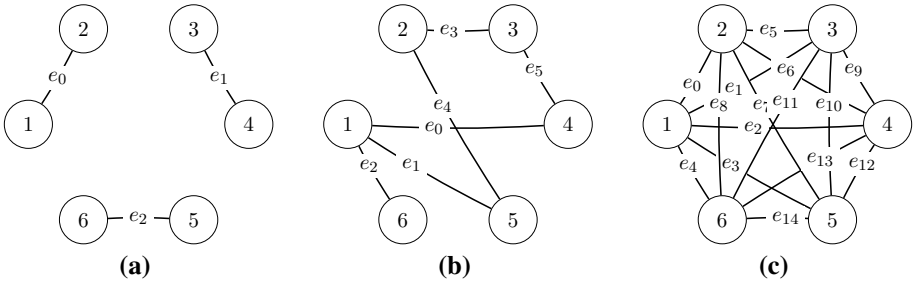


Fig. 1 Example coordination graphs for: **a** random partition, **b** overlapping factors, **c** complete factorization

between agents and many algorithms exploit such structure and require good approximations of the action-value function in order to efficiently select a maximizing joint action, e.g., *variable elimination* [12] or *max-sum* [20, 36].

However, there are many cases in which the problem itself is not factored, or the factorization is not known in advance and thus cannot be exploited. In these cases, however, it can still be useful to resort to an approximate factorization [15]:

$$Q(a) \approx \hat{Q}(a) = \sum_{e} \hat{Q}_e(a_e), \tag{2}$$

obtained by decomposing the original function into a number of local approximate terms $\hat{Q}_e(a_e)$, thus approximating the original action-value function $Q(a)$.

3 Investigated action-value factorizations

Most current value-based deep MARL approaches (of which a noticeable exception is [2]) are either based on the assumption that the joint-action value function $Q(s, a)$ can be represented efficiently by a single neural network (when, in fact, the exponential number of joint actions can certainly make a good approximation hard to learn), or that it suffices to represent (approximated) individual action values $Q_i(s_i, a_i)$ [28]. Our aim is to investigate to what degree these assumptions are valid by exploring them in the one-shot case, as well as assessing if higher-order factorizations are an improved representations of such functions, while speeding learning (as only small factors need to be learned). When a problem presents an underlying factored structure, knowing such structure beforehand and being able to exploit it properly can be of the greatest benefit both in terms of learning speed and accuracy, but we argue that resorting to an approximate factorization can still be beneficial in many cases.

We use neural networks as function approximators to represent the various components of these factorizations. In our study, we vary two distinct aspects of the problem. Firstly, we study two learning algorithms, which we describe in Sect. 3.1. Secondly, we study different coordination graph structures, which capture how the team of agents are modelled, presented in Sect. 3.2. Finally, the full set of investigated games is presented in Sect. 3.3.

3.1 Learning algorithms

Here we present the two different learning algorithms that are investigated in our experiments. We choose these two as these are highly related to many standard sequential MARL algorithms: the mixture of experts learning rule follow the same idea of standard independent learning approach used by early works [42], while the factored Q-function rule uses a joint optimization process resembling that of value decomposition networks [40], but it is also similar to the QMIX algorithm [35], but uses a linear constant mixing rather than an additional mixing network.

– *Mixture of experts* [1]: each factor network optimizes its own output $\hat{Q}_e$ individually to predict the global reward, thus becoming an "expert" on its own field of action. The loss for the network representing factor $e \in \mathcal{E}$ at training step t is defined as:

$$\mathcal{L}_t^e(a_t^e) = \frac{1}{2}\big(Q(a_t) - \hat{Q}_e(a_t^e)\big)^2, \tag{3}$$

where $Q(a_t)$ is the common reward signal received after selecting joint action a_t and $\hat{Q}_e(a_t^e)$ is the output of the network for local joint action a_t^e. As we aim to assess how good the approximate action-value function $\hat{Q}$ is, *after* training we compute the reconstruction obtained from the factors as the mean over the appropriate local Q-values (the "opinion" of each expert is weighted equally):

$$\hat{Q}(a) = \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} \hat{Q}_e(a_e) \quad \forall a \in A. \tag{4}$$

– *Factored Q-function* [13, 40]: the algorithm jointly optimizes the factor networks to predict the global reward as a sum of their local Q-values $\hat{Q}_e$. The loss for the experience at time t is identical for all factor networks:

$$\mathcal{L}_t(a_t) = \frac{1}{2}\left(Q(a_t) - \sum_{e\in\mathcal{E}} \hat{Q}_e(a_t^e)\right)^2. \tag{5}$$

Rather than learning proper action-value functions, the optimization problem in Equation 5 learns utility functions for each factor, that do not really represent the values of actions on their own, while Equation 3 learns individual Q-values for each factor. After learning, the approximate joint action-value function $\hat{Q}$ is reconstructed by summing the appropriate local Q-values (the components collectively reconstruct the approximation):

$$\hat{Q}(a) = \sum_{e\in\mathcal{E}} \hat{Q}_e(a_e) \quad \forall a \in A. \tag{6}$$

3.2 Coordination graphs

We study four different coordination graphs. Their structures differ both in the number of components and the degree of connection for each agent. Our empirical study

considers all eight combinations of the two learning rules described above and the four coordination graphs described below.

- *Single agent decomposition*: each agent i is represented by an individual neural network and computes its own individual action-values $\hat{Q}_i(a_i)$, based on its local action a_i. Under the mixture of experts learning rule, this corresponds to the standard independent Q-learning approach in MARL [42], in which we learn local agent-wise components, while under the factored Q-function approach this corresponds to value decomposition networks (VDN) [40].
- *Random partition*: agents are randomly partitioned to form factors of size f, with each agent i involved in only one factor.[2] Each of the $|\mathcal{E}| = \frac{n}{f}$ factors has a different neural network that represents local action-values $\hat{Q}_e(a_e)$ for that factor.
- *Overlapping factors*: a fixed number of factors $|\mathcal{E}|$ is picked at random from the set of all possible factors of size f. We require the sampled set to not include duplicate factors (we use only distinct components) and that every agent i appears in at least one factor. Every factor $e \in \mathcal{E}$ is represented by a different neural network learning local action-values $\hat{Q}_e(a_e)$ for the local joint action a_e. In our experiments we choose $|\mathcal{E}| = n$, to keep the number of networks comparable to that of the single agent decomposition.
- *Complete factorization*: each agent i is grouped with every possible combination of the other agents in the team $D \setminus i$ to form factors of size f, resulting in $|\mathcal{E}| = \binom{n}{f}$ factors, each represented by a network. Each of these networks learns local action values $\hat{Q}_e(a_e)$.

A fundamental problem in MARL is that there is currently no method capable of predicting the accuracy of a factored representation on a certain problem in advance (the problem is equivalent to predicting the result of a linear regression problem with a given set of basis functions). Therefore, assessing the performance and eventual advantages of different structures and approaches is a fundamental step for MARL research, as it can further improve our understanding of these settings and existing algorithms. In our empirical study, we mainly consider factors of size $f \in \{2, 3\}$. The small size of these factors allow us to effectively explore the improvements in the complexity of learning; if the size of each factor is similar to the size of the full team of agents, we would not expect significant improvement over a full joint learner in terms of sample complexity and scalability (although we also conduct some experiments on this in Sect. 4.3).

3.3 Investigated games

We investigate the proposed methods on a number of cooperative one-shot games that require a high degree of coordination. Some of these games do not present an underlying factored structure, while others are truly factored games. For the latter, none of the methods exploit prior knowledge of their true factored structure (but we also report results for the true underlying factorization to show the possible benefits when that is known beforehand).

[2] If f is not a divisor of n, the random partition factorization would partition into factors that are all of size close to f.

3.3.1 Non-factored games

Dispersion Games: In the Dispersion Game, also known as Anti-Coordination Game, the team of agents must divide as evenly as possible between the two local actions that each agent can perform [11]. Think of a town with two different pubs: the inhabitants like both the same, but the two are quite small and cannot contain all the people in the town at once, so the customers have to split up across the two pubs in order to enjoy the situation and not overcrowd them. This game requires explicit coordination, as none of the local actions is good per se, but the obtained reward depends on the decision of the whole team. We investigate two versions of this game: in the first one the agents obtain reward proportional to their *dispersion coefficient* (i.e., how split the agents are in performing one of their two local actions). The reward function $Q(a)$ for this game with n agents, each with a local action set $A_i = \{a_0, a_1\}$ is:

$$Q(a) = n - \max\{\#a_0, \#a_1\}. \tag{7}$$

In the second version, which we call Sparse Dispersion Game, the agents receive a reward (which we set to the maximum dispersion coefficient with n agents: $\frac{n}{2}$) only if they are perfectly split:

$$Q(a) = \begin{cases} \frac{n}{2} & \text{if } \#a_0 = \#a_1, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

Platonia Dilemma: In the Platonia Dilemma [18], an eccentric trillionaire gathers 20 people together and tells them that if one and only one of them sends him a telegram by noon the next day, that person will receive a billion dollars. In our cooperative version the reward is set to the number of agents n and is received by the whole team, not just a single agent. Thus, the reward function for n agents with local action sets $A_i = \{send, idle\}$ is:

$$Q(a) = \begin{cases} n & \text{if } \#send = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Climb Game: In the Climb Game [45], each agent has three local actions $A_i = \{a_0, a_1, a_2\}$. Action a_0 yields a high reward if all the agents choose it, but no reward if only some do. The other two are suboptimal actions that give lower reward but do not require precise coordination. This game enforces a phenomenon called *relative overgeneralization*, [45] that pushes the agents to underestimate a certain action (in our example, a_0) because of the low rewards they usually receive, while they could get a higher reward by perfectly coordinating on it. The reward function $Q(a)$ is:

$$Q(a) = \begin{cases} n & \text{if } \#a_0 = n, \\ \frac{n}{2} & \#a_0 = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Penalty Game: Similarly to the Climb Game, in the Penalty Game [45] each agent has three local actions $A_i = \{a_0, a_1, a_2\}$. In this game, two local actions (for example, action a_0 and a_2) give a high reward if the agents perfectly coordinate on one of them, but also give a negative penalty if they mix them together. The third action a_1 is suboptimal and gives a lower reward when the team coordinates on it, but also no penalty if at least one of the agents uses it. This game could also lead to relative overgeneralization, as the suboptimal

action is perceived as giving a higher reward than the optimal ones on average. We use the following reward function:

$$Q(a) = \begin{cases} n & \text{if } \#a_0 = n, \text{ or } \#a_2 = n, \\ \frac{n}{2} & \text{if } \#a_1 = n, \\ 0 & \text{if } 0 < \#a_1 < n, \\ -n & \text{otherwise.} \end{cases} \tag{11}$$

3.3.2 Factored games

Generalized Firefighting: The Generalized Firefighting problem [32] is an extension of the standard two-agent firefighting problem to n agents. This is a cooperative graphical Bayesian game, so each agent i has some private information, called its local type $\theta_i \in \Theta_i$, on which it can condition its decisions. The combination of the various agents types $\theta = \langle \theta_1, \dots, \theta_n \rangle$ determines the values of the reward function $Q(a, \theta)$. We have a team of n firefighters that have to fight possible fires at N_h different houses. Each house j can be burning, F_j, or not, N_j. Each agent i has a limited observation and action field: it can observe only N_o houses (so its local type is $\theta_i \in \{F_j, N_j\}^{N_o}$) and can fight the fire only at N_a houses (the sets of the observed and reachable houses are fixed beforehand and are part of the problem specification, with N_o and N_a being their cardinality respectively). Each house h yields a reward component q_h: if one and only one agent fights the fire at a burning house, that house gives a positive reward $q_h = 2$; if the house is not burning (or if it is burning but no-one is fighting the fire at it) it does not provide any reward $q_h = 0$. The reward function is sub-additive: if two agents fight the fire at the same burning house, this gives a reward $q_h = 3 < 2 \cdot 2$. The overall value of the reward function $Q(a, \theta)$ experienced by agents for a given joint type θ and joint action a is the sum of the rewards given by each house q_h:

$$Q(a, \theta) = \sum_{h \in N_h} q_h. \tag{12}$$

Therefore, the optimal strategy for the n agents is to split as evenly as possible across all the burning houses $F_j \in \theta$. If the number of burning houses is more than that of the agents, each agent should attend at a different house and fight the fire there, while if there are less burning houses than agents, the remaining agents should exploit sub-additivity and help their colleagues at already attended houses.

In our experiments we do not input the local types θ_i to the neural networks, but we instead use these to artificially inflate the size of the local action sets (and the joint one thereby) by considering the cardinal product $A_i \times \Theta_i$ as the new action set for agent i, where the agent choose the action $a_i \in A_i$ and the problem choose the local type $\theta_i \in \Theta_i$. In practice, this correspond to individually consider each local action for each possible local type, as if the agents are playing a different game (a different joint type) chosen by the environment every time, and they model the values of their actions on each game separately.

Aloha: In Aloha [31] there is a set of nearby islands, each provided with a radio station, trying to send messages to their inhabitants. We present a slightly altered one-shot version in which the ruler of each island wants to send a radio message to its inhabitants, but, given that some of the islands are near one to another, if they all send the message the radio frequencies interfere and the messages are not correctly received by the respective populations. Given that all the rulers are living in peace and they want to maximize the number of received messages by their populations, the reward

Table 1 Combinations of factorizations and learning rules

	Mix. of experts	Factored Q
Single agent	M1(=IQL [42])	F1(=VDN [40])
Random partition ($f = 2, 3$)	M2R, M3R	F2R, F3R
Complete factorization ($f = 2, 3$)	M2C, M3C	F2C, F3C
Overlapping factors ($f = 2, 3$)	M2O, M3O	F2O, F3O
True factorization (Factored games only)	MTF	FTF

signal is shared and thus the game is cooperative. It is a graphical game, as the result of each island transmission is affected only by the transmissions of nearby islands. Every ruler i has two possible actions: send a message or not. If they do not send a message, they do not contribute to the total reward. If they send one and the message is correctly received by the population (no interference occurs) they get a reward $q_i = 2$, but if they interfere with someone else, they get a penalty of $q_i = -1$. The common reward that all the rulers receive at the end is the sum of their local contributions:

$$Q(a) = \sum_{i \in n} q_i. \tag{13}$$

4 Experiments

With our analysis, we aim at investigating the following research questions (RQs):

1. Comparisons to baselines: how well can the investigated methods represent the action-value function of different cooperative multi-agent systems (both truly factored or not)? How do these compare to both independent learners and joint learners?
2. Impact of factors size: how small can the factors of these methods be with respect to the team size? How is the factor size affecting the learned representations?
3. Scalability: how do the compared methods scale in the number of agents?
4. Sample efficiency: how is the sample efficiency of these methods compared to both independent learners and joint learners?
5. Exploratory policy: how do the same investigated methods behave with a non-uniform, time-varying policy used to select actions?

The remainder of this Section is organized as follows: we address RQ1 in Sect. 4.2 by comparing the methods against both independent learners and a joint learner on a variety of different games, we then investigate RQ2 by selecting one of the games and comparing the effect of using small factors versus larger ones in Sect. 4.3, a couple of games with an increasing number of agents is then investigated in Sect. 4.4 to address RQ3, while RQ4 is tackled in Sect. 4.5. An initial step toward RQ5 is made in Sect. 4.6 and finally, a summary of the results and general takeaways are given in Sect. 4.7.

4.1 Experimental setup

Table 1 defines the abbreviations and acronyms of the combinations of learning approach, coordination graph structure, and factor size used throughout our analysis (other than where differently stated). In our empirical evaluation, we investigate these combinations on the one-shot coordination games presented above.

We hypothesize that factored representations, by avoiding the combinatorial explosion in the number of joint actions and allowing for some internal coordination inside each factor, are going to produce representations closer to the original action-value function for these multi-agent problems. We also expect them to be sample efficient due to this small size of the factors, speeding up the required training time and learning good representations faster than the other approaches.

We train the neural networks of the factored representations to reproduce action-value functions for the detailed cooperative one-shot games, using the loss functions and coordination graph structures described in Sect. 3 as combined in Table 1. After training, the representation $\hat{Q}(a)$ is reconstructed from the factor components' outputs and compared with the original action value function $Q(a)$ (complete knowledge of this function is withheld from the networks during training, but only samples corresponding to the selected joint action a are provided at every step) to assess the quality of the representation, both in terms of action ranking and reconstruction error, as defined in the Problem Statement in Sect. 2.1.

We keep the same hyperparameters for all the investigated representations to favour a fair comparison of the learned representations: using the same learning rates ensures that no method can learn faster than the others, while using the same structure for all the neural network guarantees that none is given with more representational power. Every neural network has a single hidden layer with 16 hidden units using the leaky ReLU activation function, while all output units are linear and output local action-values $\hat{Q}_e(a_e)$ for every local joint action a_e[3]. Given the absence of an environment state to feed to the networks as an input, at every time step they just receive a constant scalar value. We use the mean square error (MSE) defined in Sect. 2.1 as the loss function and the RMSprop training algorithm with a learning rate of $\eta = 10^{-5}$. For every game, we train the networks with 100, 000 examples by sampling a joint action a_t uniformly at random.[4] Then, we propagate the gradient update through each network e from the output unit $\hat{Q}_e(a_t^e)$. The loss function minimizes the squared difference between the collected reward $Q(a_t)$ at each training step and the approximation computed by the networks. After training, the learned action-value function $\hat{Q}$ is compared to the original Q. We also consider a baseline joint learner (a single neural network with an exponential number $|A| = |A_i|^n$ of output units). Every experiment was repeated 10 times with random initialization of weights, each time sampling different factors for the random partitions and the overlapping factors; we report the averages of these 10 runs.

[3] We also did some preliminary experiments with deeper networks with 2 and 3 hidden layers, but did not find improvements for the considered problems.

[4] We do not use ϵ-greedy because we are interested in representing the whole value function and not just the best performing action at every training step and collecting the reward $Q(a_t)$, as noted in Sect. 2.1.

Table 2 Details of the investigated games in this section

Game	n	$\lvert A_i \rvert$	$\lvert A \rvert$	Optimal	Factored
Dispersion Game	6	2	64	20	No
Platonia Dilemma	6	2	64	6	No
Climb Game	6	3	729	1	No
Penalty Game	6	3	729	2	No
Generalized Firefighting	6	2 (per type)	64 (8192 total)	779	Yes
Aloha	6	2	64	2	Yes

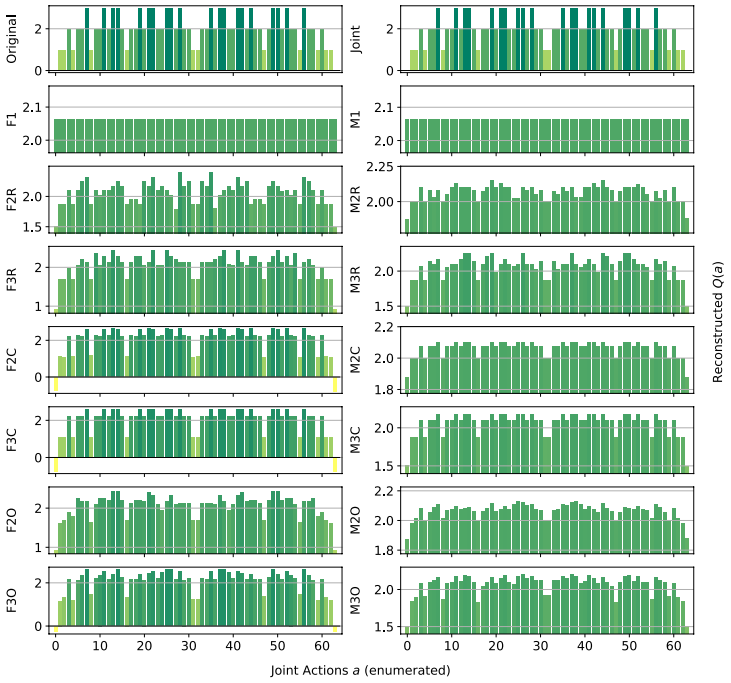
4.2 Comparison to baselines

Our aim here is to show that factored representation are suitable to represent a wide variety of games, including many that do not present any real underlying factorization, and that these can perform better than both independent learners and a joint learner baselines. We start by discussing the approximate value functions obtained by the investigated representations, with the following Table 2 summarizing the games that we use and their associated parameters.

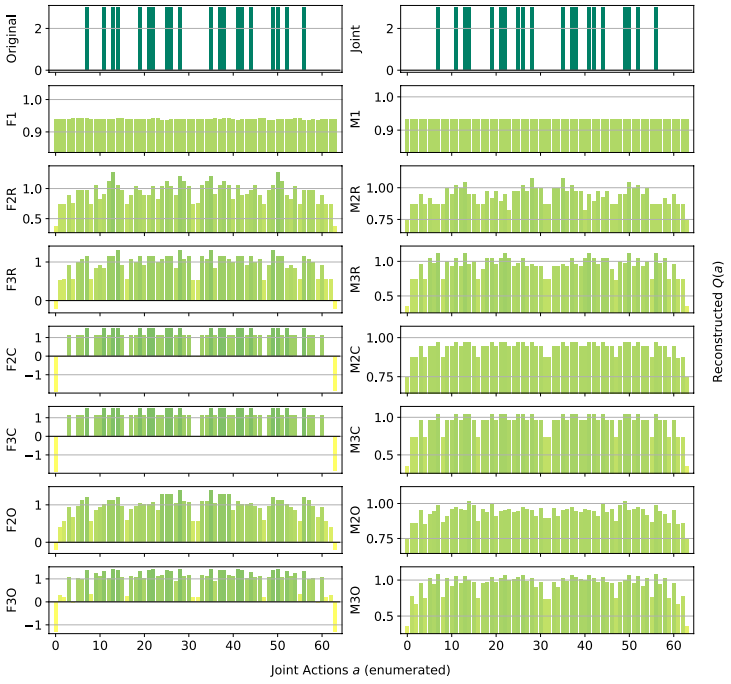
In the following plots, the x-axis enumerates the joint actions $a \in A$ and the y-axis shows the corresponding values $\hat{Q}(a)$ for the reconstructed functions, with the heights of the bars encoding the magnitude of the action-values $\hat{Q}(a)$. As defined in the Problem Statement in Sect. 2.1, we analyse the quality of the computed reconstructions considering two aspects: the total reconstruction error of $\hat{Q}(a)$ with respect to the true reward function $Q(a)$ $\forall a \in A$, and whether a reconstruction produces a correct ranking of the joint actions. For a good reconstruction, the bars have to have the same relative heights, indicating that the representation correctly ranks the joint actions with respect to their value, and to be of a similar value to those in the original one (the representation can reconstruct a correct value for that joint action). However, reconstruction error alone is not a good accuracy measure because lower reconstruction error does not imply better decision making, as a model could lower the total error by over- or underestimating the value of certain joint actions.

Dispersion Games: Figure 2 shows the Q-function reconstructed by the proposed factorizations and learning approaches for the two variants of the Dispersion Game. Figure 2a shows that the proposed complete factorizations are able to almost perfectly reconstruct the relative ranking between the joint actions, meaning that these architectures can be reliably used for decision making. Moreover, the ones using the factored Q-function (F2C and F3C in the plot) are also able to produce a generally good approximation of the various values (expressed by the height of the bars), while those based on the mixture of experts produce a less precise reconstruction: the joint optimization of the former gives an advantage in this kind of extremely coordinated problems.

Smaller factorizations, like the random pairings, are not sufficient to correctly represent this function, probably because a higher degree of connection is required to achieve good coordination. Figure 2b is similar but in this case the reconstruction is less accurate and the values of the bars are quite different from those of the original one. This is possibly due to the sparsity of the function, requiring the networks to correctly approximate quite different values with the same output components. In this case, the sparsity of the function to represent fools the representations into being similar to those of the non-sparse version.



(a)



(b)

Fig. 2 Reconstructed $Q(a)$ for **a** the Dispersion Game, and **b** its sparse variant

Table 3 Best (bold) and worst (italic) performing methods on the two variants of the Dispersion Game

Model	MSE	Opt. Found	Ranked
Dispersion Game			
Joint	**0.00 ± 0.0**	**20 ± 0**	**64 ± 0**
F1	*0.62 ± 0.0*	*5 ± 2*	*21 ± 2*
F2R	*0.52 ± 0.0*	*8 ± 0*	*24 ± 1*
F2C	**0.09 ± 0.0**	**20 ± 0**	**64 ± 0**
F3C	**0.09 ± 0.0**	**20 ± 0**	**64 ± 0**
F3O	**0.19 ± 0.0**	13 ± 1	47 ± 3
M1	*0.62 ± 0.0*	*6 ± 1*	*24 ± 2*
M2R	*0.56 ± 0.0*	*8 ± 0*	*24 ± 1*
M2C	*0.55 ± 0.0*	**20 ± 0**	**64 ± 0**
M3C	0.43 ± 0.0	**20 ± 0**	**64 ± 0**
M2O	*0.56 ± 0.0*	10 ± 2	36 ± 4
Dispersion Game (sparse)			
Joint	**0.00 ± 0.0**	**20 ± 0**	**64 ± 0**
F1	*1.93 ± 0.0*	*7 ± 1*	*37 ± 1*
F2R	*1.83 ± 0.0*	*8 ± 0*	40 ± 0
F2C	1.41 ± 0.0	**20 ± 0**	**64 ± 0**
F3C	1.41 ± 0.0	**20 ± 0**	**64 ± 0**
M1	*1.93 ± 0.0*	*6 ± 1*	*37 ± 2*
M2R	*1.88 ± 0.0*	*8 ± 0*	40 ± 0
M2C	*1.87 ± 0.0*	**20 ± 0**	**64 ± 0**
M3C	1.74 ± 0.0	**20 ± 0**	**64 ± 0**
M2O	*1.87 ± 0.0*	10 ± 1	44 ± 2

In Table 3 (as well as in the similar ones for the other games) we report the best and worst performing methods on the two variants of this game against a set of different measures that explore our results in terms of the Problem Statement presented in Sect. 2.1: the mean squared error (MSE) tells us how far is the reconstructed action-value function with respect to the original one, while the number of optimal joint actions found (Opt. Found, i.e. how many of the true optimal actions are also ranked as optimal by the reconstruction), and the total number of correctly ranked actions (Ranked) points out how reliably these can reconstructions be used in decision making. Methods performing not too bad but also not the best are left out for compactness, for more data and measures for this game (and similarly for the following ones) please see the Appendix.

We observe how the joint learner can easily learn the entire action-value function for this small setting, resulting in a perfect ranking and a very small error. However, methods using the complete factorizations are also able to do so, with the mixture of experts achieving a larger reconstruction error but still a correct ranking of actions, including identifying all of the optimal ones, on both variants of this game. Independent learners instead do not seem able to correctly identify all of the optimal actions, also achieving a very large reconstruction error.

Platonia Dilemma: Figure 3 shows the reconstructed action-value functions for the Platonia Dilemma. For this problem, none of the proposed factorizations can correctly represent the action-value function. In fact, while they are perfectly able to correctly

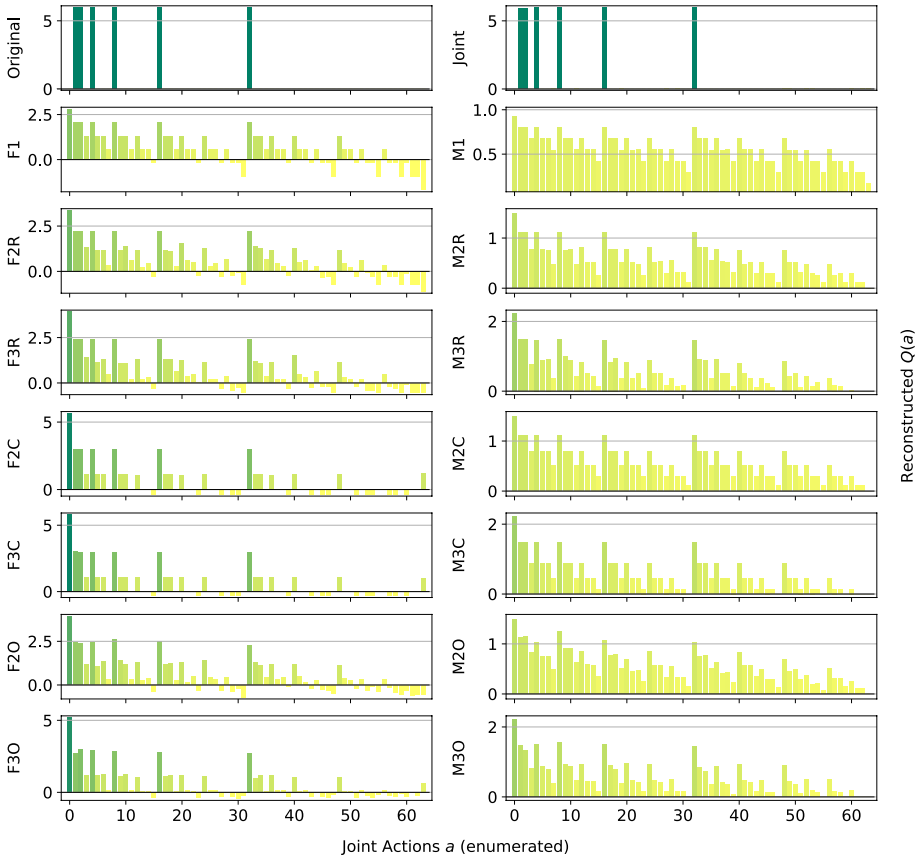


Fig. 3 Reconstructed $Q(a)$ for the Platonia Dilemma

Table 4 Best (bold) and worst (italic) performing methods on the Platonia Dilemma

Model	MSE	Opt. Found	Ranked
Joint	**0.00 ± 0.0**	**6 ± 0**	**64 ± 0**
Ml	*2.80 ± 0.0*	5 ± 0	62 ± 0
M2O	2.54 ± 0.0	*4 ± 0*	*61 ± 1*
M3O	2.28 ± 0.0	*4 ± 0*	*61 ± 1*

rank all the optimal actions (the ones in which only a single agent sends the telegram) at the same level, they all fail to correctly rank and reconstruct the same joint action (that is, the one in which none of the agents sends the telegram). In fact, the unique symmetric equilibrium for the team in this game is that each of them sends the telegram with probability $\frac{1}{n}$, so the agents usually gather more reward by not sending it themselves, but relying on someone else to do so. This results in an 'imbalanced' action-value function in which the high reward is more often obtained, from an agent perspective, by choosing

a certain action instead of the other, thus resulting in overestimating one of the actions (the one in which all the agents perform the same action, i.e., not sending the telegram).

This imbalance in the reward given by the two actions is probably the cause of the poor reconstruction. Thus, for this kind of tightly coupled coordination problem, none of the techniques to approximate action-values currently employed in deep MARL suffice to guarantee a good action is taken, even if the coordination problem is conceptually simple. Table 4 reports the best and worst performing methods on this game.

All of the methods using the factored Q-function learning approach are left out, as these achieve average performances. As already showed by Fig. 3, here only the joint learner is able to correctly identify all of the optimal actions. The mixture of experts with the overlapping factorization are the ones that perform the worst, possibly because the connectivity of this structure is too sparse to help in such an imbalanced reward game.

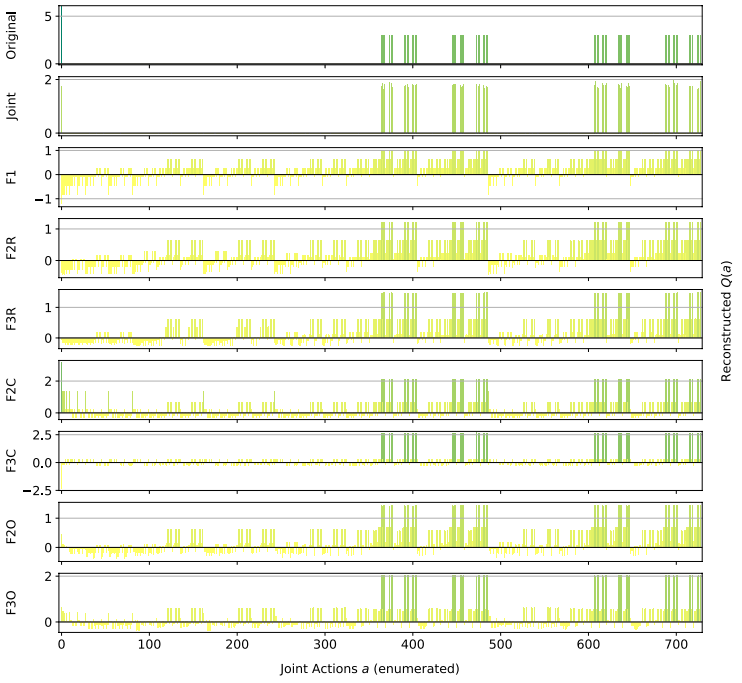
Climb Game: Figure 4 shows the results obtained on the Climb Game. The joint network is not able to learn the correct action-value function in the given training time, due to the large number of joint actions. This highlights again how joint learners are not suited for this kind of even moderately large multi-agent system. By contrast, all the other architectures correctly rank the suboptimal actions.

The methods using the factored Q-function and a complete factorization are also able to correctly reconstruct the values for most of the joint actions, as can be seen from the bars. However, only F2C can correctly rank and reconstruct the optimal action (the coordinated one), while even F3C fails to do so and gives it a large negative value. A likely cause for this effect is that, where optimizing the loss function, assigning negative values to the components forming that joint action reduces the overall mean squared error, even if one of the reconstructed reward value is totally wrong. We can also observe how the mixture of experts plot looks somewhat comparable to the one for the factored Q-function learning approach, but more 'compressed' and noisy. Table 5 reports the best and worst performing methods on the Climb Game.

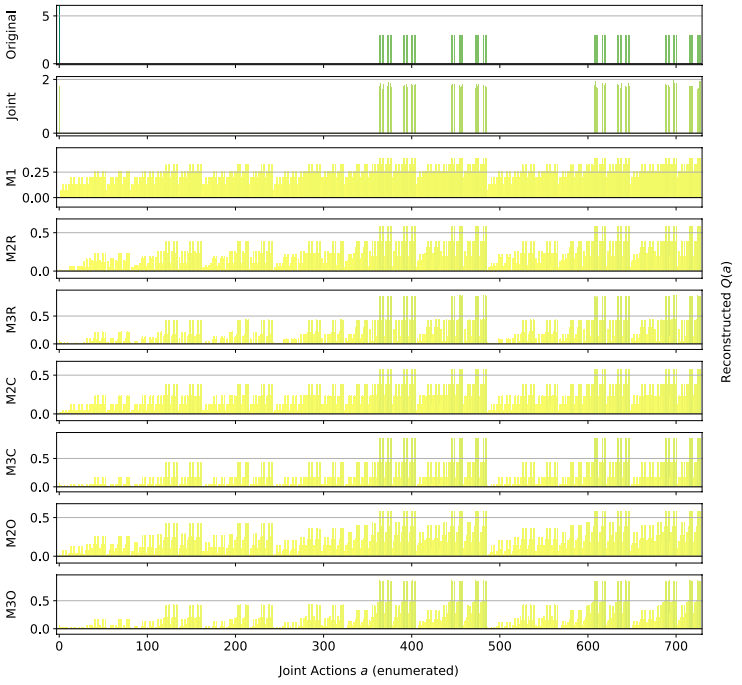
With a larger joint action space, the joint learner begins to struggle and achieves a larger reconstruction error than some of the factored methods. Interestingly, F2C is the only method capable of identifying the optimal action of this game, when also F3C fails. We hypothesize that this happens because the larger factors push the overall representation to further improve the reconstructed values for the other local joint actions at the expense of these forming the optimal action itself. This points out how, although generally a larger factor size entails a better representation, it may not always be so. On the other hand however, it also shows how small factors can result in a good representation that is also easier and faster to be learned.

Penalty Game: Figure 5 presents the representations obtained by the investigated approximations. Given the high level of coordination required, all of the architectures using the mixture of experts learn a totally incorrect approximation, biased by the larger number of joint actions that yield a penalty rather than a positive reward.

For this game, none of the architectures can correctly reconstruct the whole structure of the action value function, and they all fail at the two optimal joint actions (at the two sides of the bar plots). This is probably due to the large gap in the reward values that the agents can receive when choosing one of their coordinated actions: they can get a high reward if all the agents perfectly coordinate, but it is more common for them to miscoordinate and receive a negative penalty, resulting in an approximation that ranks those two joint actions as bad in order to correctly reconstruct the other cases. Furthermore, the suboptimal action is hard to correctly approximate because, similarly to the optimal ones, it also usually results in a smaller reward than the one it gives when all the agents coordinate on



(a)



(b)

Fig. 4 Reconstructed $Q(a)$ for the Climb Game: **a** factored Q-function learning approach, and **b** mixture of experts learning approach

Table 5 Best (bold) and worst (italic) performing methods on the Climb Game

Model	MSE	Opt. Found	Ranked
Joint	**0.17 ± 0.1**	0 ± 0	727 ± 1
F2C	**0.25 ± 0.0**	**1 ± 0**	**729 ± 0**
F3C	**0.17 ± 0.0**	0 ± 0	726 ± 0
Ml	*0.71 ± 0.0*	0 ± 0	726 ± 0

it. Only F1 and F3C rank it as better than the other, but surprisingly only F1 is also able to reconstruct the correct value. Table 6 reports the best and worst performing methods on this game.

For this setting as well, the joint learner is struggling to represent the entire action-value function, although it is the only method capable of correctly identify one of the optimal joint actions. All the other methods fail in doing so, even though some of those that use the factored Q-function learning approach achieve a very small MSE.

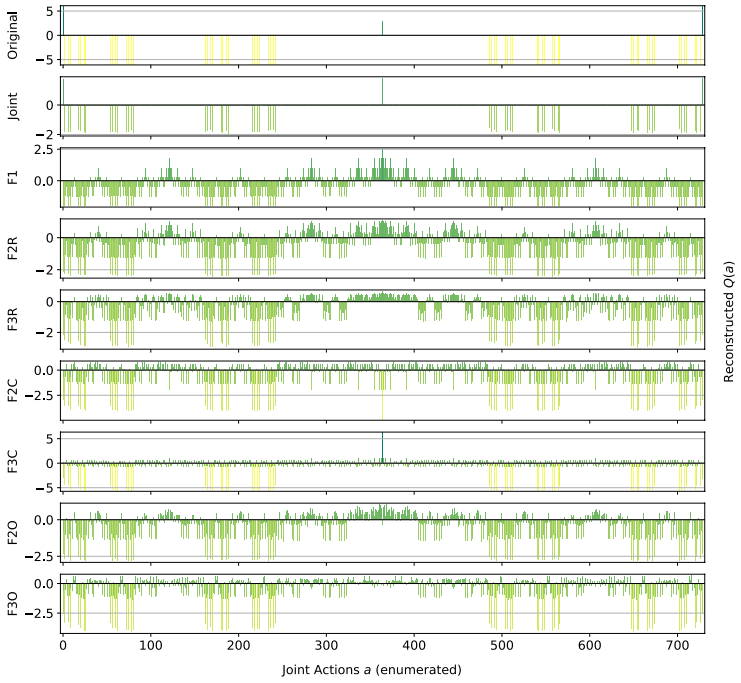
Generalized Firefighting: In our experiments, a team of $n = 6$ agents have to fight fire at $N_h = 7$ houses. Each agent can observe $N_o = 2$ houses and can fight fire at the same set of locations ($N_a = 2$), disposed as shown in Fig. 6. Figure 7 shows the representations learned for the joint type $\theta = \{N_1, F_2, N_3, F_4, N_5, N_6, F_7\}$.

This game requires less coordination than those studied earlier (agents have to coordinate only with other agents that can fight fire at the same locations), and every investigated architecture correctly ranks all the joint actions, even the single agent factorizations F1 and M1. However, while those using the factored Q-function can also correctly reconstruct the reward value of each action, those using the mixture of experts are less precise in their reconstructions. Overall, this experiment demonstrates that there exist non-trivial coordination problems that can effectively be tackled using small factors, including even individual learning approaches. Also, it is to note how both learning approaches, when coupled with the true underlying factorization, are achieving very good reconstruction and can rank all of the joint actions correctly.

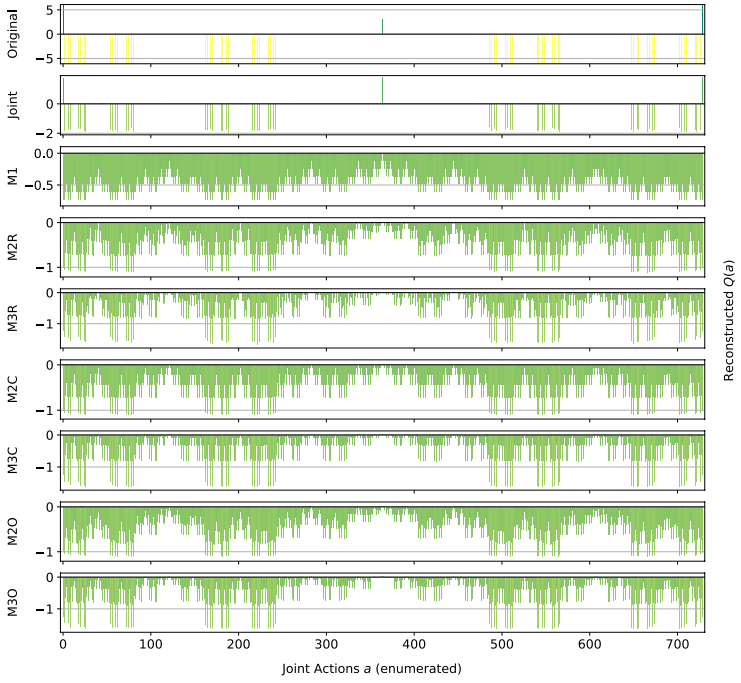
Figure 8 shows the results for a different joint type, $\theta = \{F_1, F_2, F_3, F_4, F_5, N_6, F_7\}$:

This type presents multiple adjacent houses burning at the same time, so the agents have to correctly estimate the value of fighting fire at a certain location both on their own or collaborating with other agents. The joint learner is not able to correctly learn the values for this type in the given training time, thus resulting in ranking as optimal actions that are not. Simpler factorizations like F1 or M1 on the other hand fail as well, ranking suboptimal actions as optimal. However, the other factored representations are quite accurate and correctly represent the value of coordination: even simpler factorization using overlapping factors with both learning approaches or random pairing coupled with the factored Q-function learning approach, can correctly identify the optimal joint action. Again, the representations obtained with the factored Q-function learning approach are more accurate in terms of values of the actions. Table 7 is showing the best and worst performing methods on this game. On this type as well, both FTF and MTF are achieving very good reconstructions, with FTF also approximating the values of the joint actions correctly.

Although the joint action space is very large here (more than 8000 joint actions), most of the factored methods achieves very good performance both in terms of MSE (factored Q-function learning approach methods) and action ranking. Also smaller factorizations like the overlapping factors ones are able to identify almost all of the optimal actions and produce a very good ranking. Both methods using the true factorizations



(a)



(b)

Fig. 5 Reconstructed $Q(a)$ for the Penalty Game: **a** factored Q-function learning approach, and **b** mixture of experts learning approach

Table 6 Best (bold) and worst (italic) performing methods on the Penalty Game

Model	MSE	Opt. Found	Ranked
Joint	1.60 ± 0.4	$\mathbf{1 \pm 0}$	$\mathbf{727 \pm 1}$
F1	2.18 ± 0.0	0 ± 0	*722 ± 0*
F2C	$\mathbf{1.29 \pm 0.0}$	0 ± 0	*722 ± 0*
F3C	$\mathbf{0.54 \pm 0.0}$	0 ± 0	724 ± 0
F3O	$\mathbf{1.27 \pm 0.0}$	0 ± 0	723 ± 0
M1	*2.71 ± 0.0*	0 ± 0	*722 ± 0*

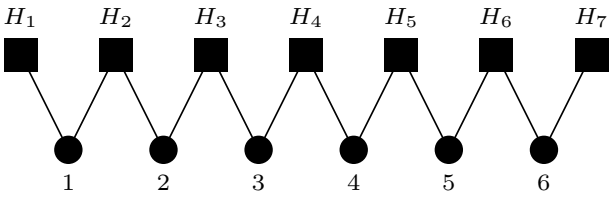


Fig. 6 Firefighters formation with $n = 6$ agents and $N_h = 7$ houses

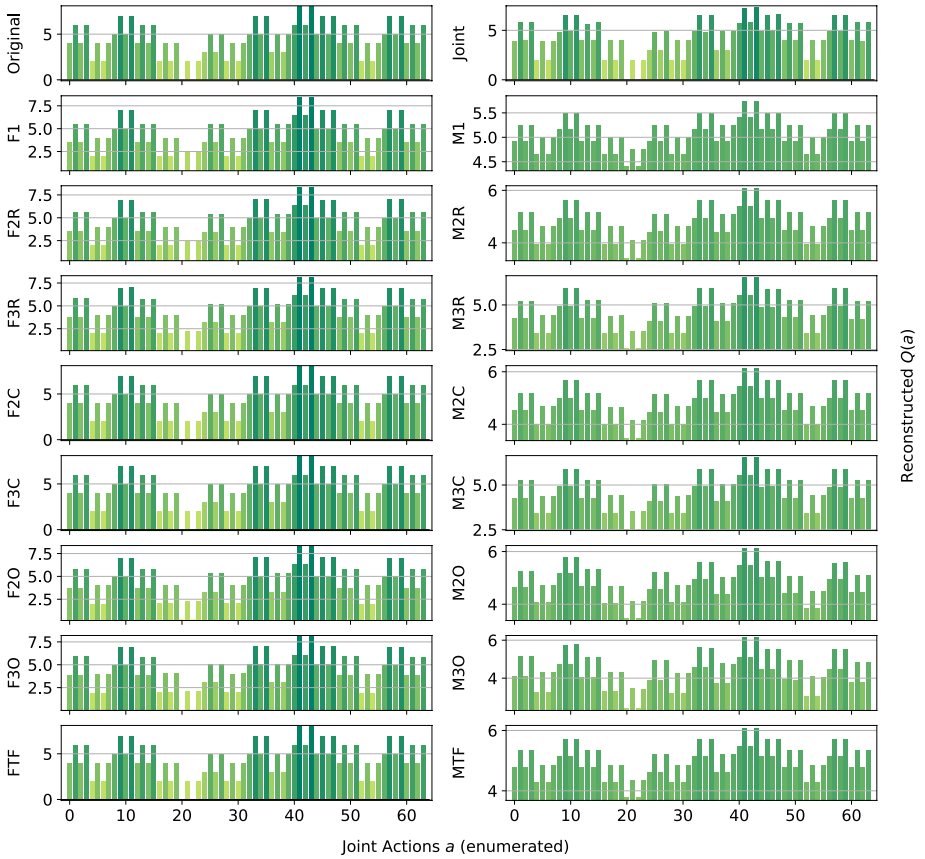


Fig. 7 Reconstructed $Q(a)$ for a single joint type of the Generalized Firefighting problem

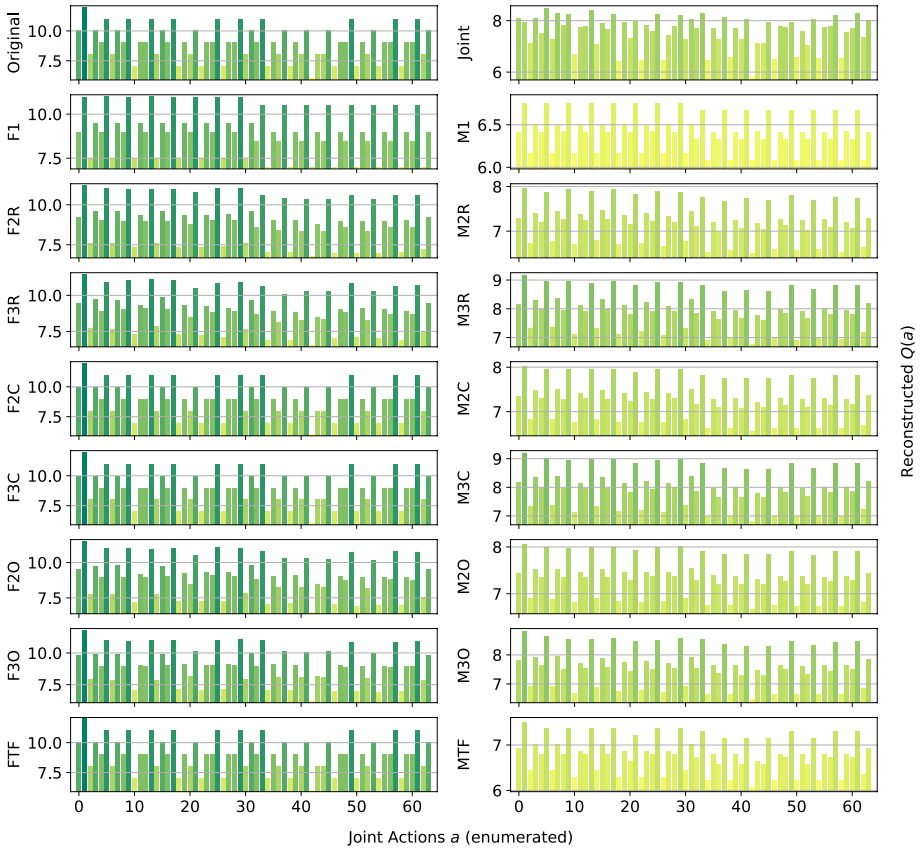


Fig. 8 Reconstructed $Q(a)$ for a different joint type of the Generalized Firefighting problem

Table 7 Best (bold) and worst (italic) performing methods on the Generalized Firefighting problem

Model	MSE	Opt. Found	Ranked
Joint	1.29 ± 2.5	*656 ± 123*	6,893 ± 1,475
F3R	**0.09 ± 0.0**	743 ± 25	7,288 ± 558
F2C	**0.00 ± 0.0**	**779 ± 0**	**8,192 ± 0**
F3C	**0.00 ± 0.0**	**779 ± 0**	**8,192 ± 0**
F2O	**0.09 ± 0.0**	747 ± 18	7,333 ± 382
F3O	**0.03 ± 0.0**	**778 ± 4**	**8,149 ± 130**
FTF	**0.00 ± 0.0**	**779 ± 0**	**8,192 ± 0**
Ml	*3.55 ± 0.0*	700 ± 6	6,220 ± 30
M2C	1.82 ± 0.0	**777 ± 0**	**7,826 ± 0**
M3C	0.85 ± 0.0	**778 ± 1**	**8,151 ± 4**
M2O	1.97 ± 0.1	741 ± 13	*5,628 ± 249*
M3O	1.73 ± 1.2	738 ± 55	*5,867 ± 682*
MTF	*2.60 ± 0.0*	**779 ± 0**	**8,177 ± 2**

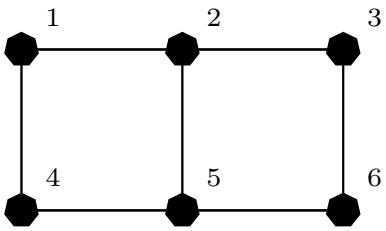


Fig. 9 Islands configuration with $n = 6$ agents

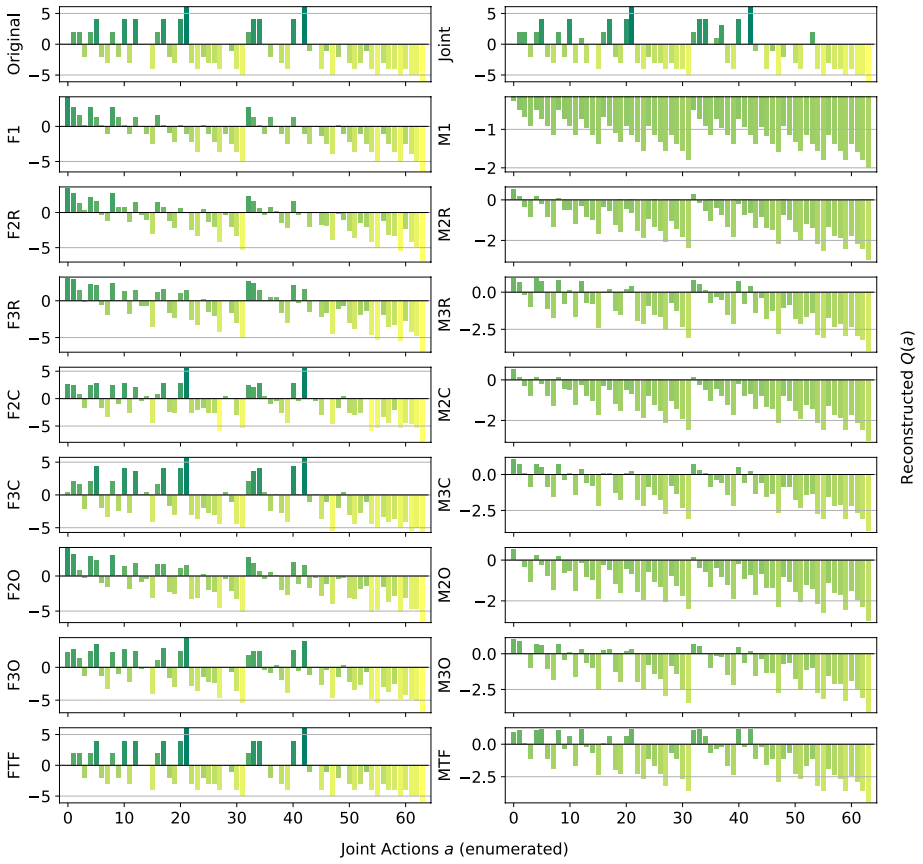


Fig. 10 Reconstructed $Q(a)$ for Aloha

are doing very well, with also the mixture of experts one identifying all of the optimal actions. On the other hand, the joint learner is failing in this task, being outperformed even by M1 (that has a higher MSE but a better ranking).

Aloha: Our experiment uses a set of $n = 6$ islands disposed in a 2×3 grid as in Fig. 9, with each island affected only by the transmissions of the islands on their sides and in front of them (islands on the corner of the grid miss one of their side neighbours). Representations learned for this game are reported in Fig. 10.

Table 8 Best (bold) and worst (italic) performing methods on Aloha

Model	MSE	Opt. Found	Ranked
Joint	1.13 ± 0.0	$\mathbf{2 \pm 0}$	51 ± 1
F2R	4.05 ± 0.4	0 ± 0	*22 ± 4*
F3R	3.16 ± 0.5	0 ± 0	*26 ± 4*
F2C	0.91 ± 0.0	$\mathbf{2 \pm 0}$	42 ± 0
F3C	$\mathbf{0.07 \pm 0.0}$	$\mathbf{2 \pm 0}$	$\mathbf{64 \pm 0}$
F2O	3.27 ± 0.3	0 ± 0	*23 ± 4*
F3O	1.46 ± 0.3	$\mathbf{1 \pm 1}$	29 ± 5
FTF	$\mathbf{0.00 \pm 0.0}$	$\mathbf{2 \pm 0}$	$\mathbf{64 \pm 0}$
M1	*8.26 ± 0.0*	0 ± 0	27 ± 1
M2R	6.52 ± 0.2	0 ± 0	*25 ± 4*
M2O	6.63 ± 0.4	0 ± 0	*22 ± 5*
M3O	4.71 ± 0.3	0 ± 0	*25 ± 4*

Table 9 Combinations of factorizations and learning rules with larger factors

	Mix. of Experts	Factored Q
Random partition ($f = 4, 5$)	M4R, M5R	F4R, F5R
Complete factorization ($f = 4, 5$)	M4C, M5C	F4C, F5C
Overlapping factors ($f = 4, 5$)	M4O, M5O	F4O, F5O

The plot shows clearly how this game is challenging for the proposed factorizations to learn, with only three of them (plus the joint learner) able to correctly represent the action-value function. The structure of the game is similar to that of Generalized Firefighting, with an agent depending directly only on a small subset of the others, but the different properties of its Q-function make it more challenging to correctly represent. This is possibly due to the large difference between the two rewards an agent can get when transmitting the radio message, depending on a potential interference. Observing only the total reward, this action looks neutral per se, similarly to what happens for the two actions in the Dispersion Game, its outcome depending on the action of the neighbouring agents, thus possibly fooling many of the proposed factorizations, especially those using the mixture of experts approach. Table 8 is showing the best and worst performing methods on this game.

On this more difficult game, all of the mixture of experts methods are not able to identify the optimal actions and achieve a very large MSE. However, the complete factorizations using the factored Q-function learning approach are able to do so, with F3C also ranking correctly all of the other joint actions. Again, FTF is performing the best, with a perfect ranking and a very low reconstruction error, outperforming even the joint learner. This once more shows how beneficial would it be to exploit an appropriate factored structure when that is known beforehand.

4.3 Impact of factors size

Although we mainly focus on factors of small size, we are also interested in investigating how the size of the factors is affecting the final representation, and if using factors of larger size can help to overcome some of the issues encountered with small factors. To investigate

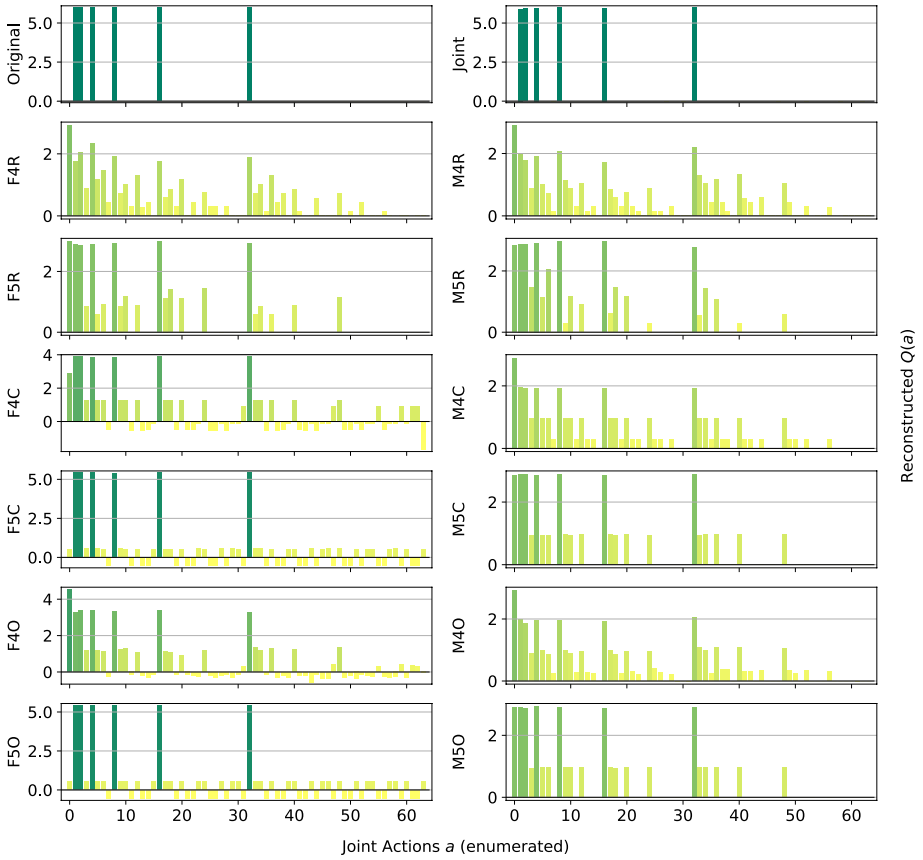


Fig. 11 Reconstructed $Q(a)$ for the Platonia Dilemma

this, we test the methods defined in Table 9 on the Platonia Dilemma and Penalty Game with $n = 6$ agents, two of the games that proved more problematic to correctly represent.

Platonia Dilemma: Figure 11 shows the reconstructed action-value functions for the Platonia Dilemma. We can see how this game, that none of the factored methods in Fig. 3 was able to solve, remains very challenging even with factors comprising more agents. Indeed, only methods with a factor size $f = 5$, thus very close to the entire team size $n = 6$, and using the factored Q-function learning approach (F5C and F5O), are able to correctly reconstruct the action-value function. The same factorizations using the mixture of experts learning approach are instead consistently ranking one of the suboptimal actions (the one in which none of the agents is sending the telegram) as an optimal one, the same as with factors of smaller size.

Penalty Game: Figure 12 presents the representations obtained by the new investigated approximations. Even with larger factors, none of the methods is able to reconstruct any of the optimal actions, but they only are able to discern the value of the sub-optimal one like F1 and F3C in Fig. 5 (that is seen as optimal). The same kind of problems that arose with smaller factors are also present here, with the mixture of experts methods tending to underestimate values for all the joint actions and generally

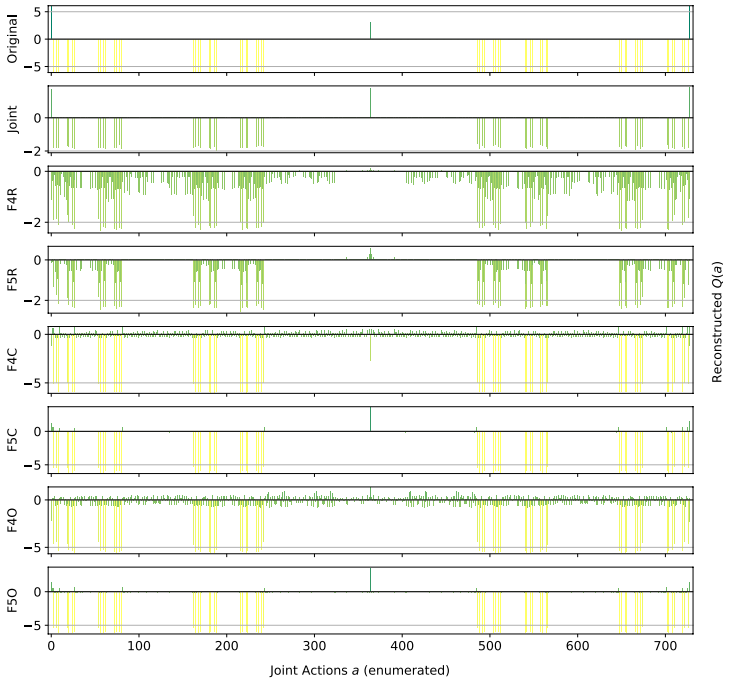


Fig. 12 Reconstructed $Q(a)$ for the Penalty Game **a** factored Q-function learning approach, and **b** the mixture of experts learning approach

Table 10 Details of the investigated games in this section

Game	n	$\lvert A_i \rvert$	$\lvert A \rvert$	Optimal	Factored
Dispersion Game	9	2	512	252	No
Dispersion Game	12	2	4096	924	No
Generalized Firefighting	9	2 (per type)	512 (524.288 total)	17.682	Yes
Aloha	9	2	512	1	Yes
Aloha	12	2	4096	2	Yes

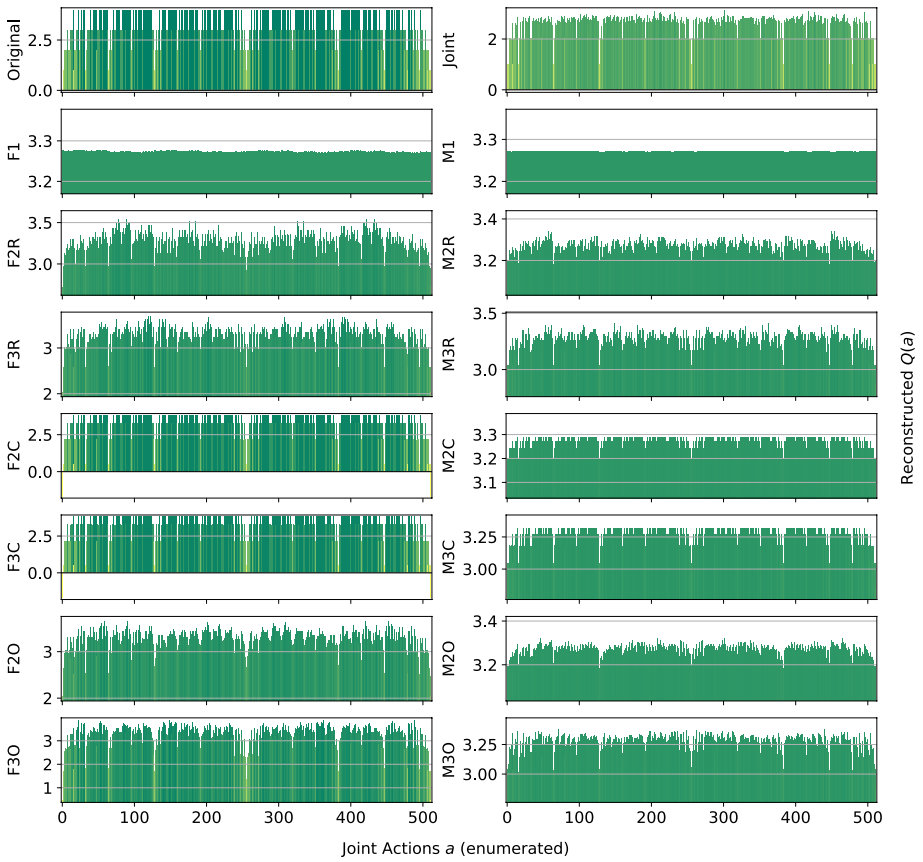


Fig. 13 Reconstructed $Q(a)$ for the Dispersion Game with $n = 9$ agents

none of the methods being able to represent the true value of coordination for this problem. However, the methods using a complete factorization coupled the factored Q-function learning approach are reconstructing small yet positive values for these optimal actions, meaning that the resulting reconstruction is at least identifying these as good actions that the agents may desire to perform.

Table 11 Best (bold) and worst (italic) performing methods on the two instances of the Dispersion Game

Model	MSE	Opt. Found	Ranked
Dispersion Game $n = 9$			
Joint	*0.93 ± 0.5*	162 ± 30	307 ± 77
F1	0.74 ± 0.0	*124 ± 1*	*191 ± 3*
F2C	**0.06 ± 0.0**	**252 ± 0**	**512 ± 0**
F3C	**0.06 ± 0.0**	**252 ± 0**	**512 ± 0**
M1	0.74 ± 0.0	*124 ± 1*	*192 ± 4*
M2C	0.70 ± 0.0	**252 ± 0**	**512 ± 0**
M3C	0.63 ± 0.0	**252 ± 0**	**512 ± 0**
Dispersion Game $n = 12$			
Joint	*19.48 ± 0.7*	*210 ± 8*	*1,108 ± 34*
F1	1.17 ± 0.0	*186 ± 39*	*1,137 ± 37*
F3R	0.99 ± 0.0	351 ± 7	*1,364 ± 20*
F2C	**0.17 ± 0.0**	**924 ± 0**	**4,096 ± 0**
F3C	**0.20 ± 0.0**	**774 ± 194**	**3,711 ± 510**
M1	1.17 ± 0.0	*187 ± 30*	*1,134 ± 36*
M3R	1.09 ± 0.0	350 ± 9	*1,363 ± 24*
M2C	1.14 ± 0.0	**813 ± 69**	**3,831 ± 246**
M3C	1.08 ± 0.0	**920 ± 5**	**4,089 ± 10**

4.4 Scalability

A fundamental aspect for a multi-agent algorithm is how well it can scale with the size of the system, i.e. when more agents are introduced and therefore the size of the joint action set exponentially increases. In this section we investigate how using a factored representation helps when such systems get larger, as well as analyse how this affects the performance of both independent learners and joint learners. Table 10 illustrates the game we use to investigate this:

Dispersion Games: Figure 13 shows the action-value function reconstructed by the proposed factorizations and learning approaches for the Dispersion Game with $n = 9$ agents (a similar figure for the case when $n = 12$ would have rendered unreadable and is thereby not included). We can observe how the complete factorizations are able to almost perfectly reconstruct the relative ranking between the joint actions even in this larger setting, showing how reliable and general can this kind of approach be. As usual, the ones using the factored Q-function are also able to produce a generally good approximation of the various components, while those based on the mixture of experts produce a less precise reconstruction: the joint optimization of the former seems to have an even bigger benefit when more agents are present.

It is interesting to note how both independent learners and the joint learner are failing here, but for different reasons: both types of independent learners seem not able to correctly learn the value of coordination with the others (something already appearing on the smaller instance shown in Fig. 2), while the latter is struggling because of the increased number of agents that makes the function it has to represent too big to be reliably learned in the given training time. The other factored approaches instead are capturing the value of such coordination up to some extent (especially these using the overlapping factors), but the small number of factors is probably not sufficient to completely represent such a

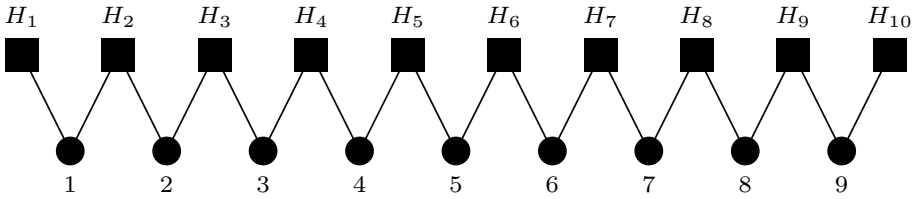


Fig. 14 Firefighters formation with $n = 9$ agents and $N_h = 10$ houses

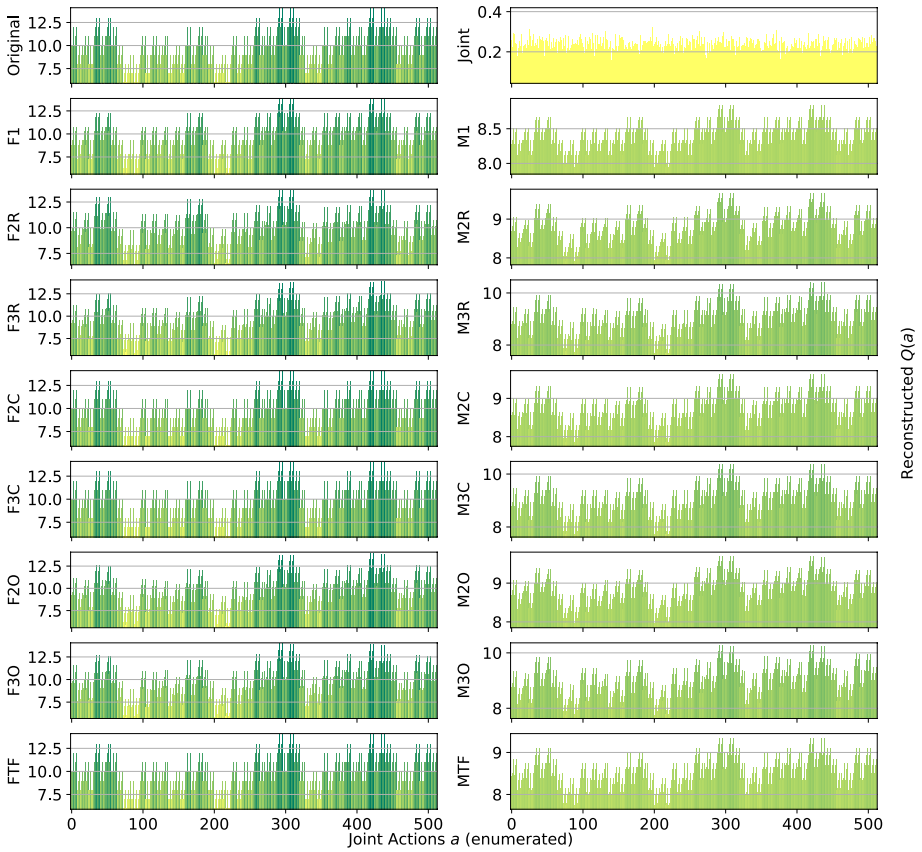


Fig. 15 Reconstructed $Q(a)$ for a single joint type of the Generalized Firefighting problem with $n = 9$ agents

function. However, the resulting MSE is still lower than the joint learner and some of the optimal actions are still ranked correctly, making these approaches still viable for decision making. Table 11 reports the best and worst performing methods on the two instances of this game, both in terms of action ranking and reconstruction error.

As already stated, when the size of the system increase both independent learners and the joint learner struggle in representing the corresponding action-value function correctly. The latter especially, that was achieving a perfect reconstruction for the same game with

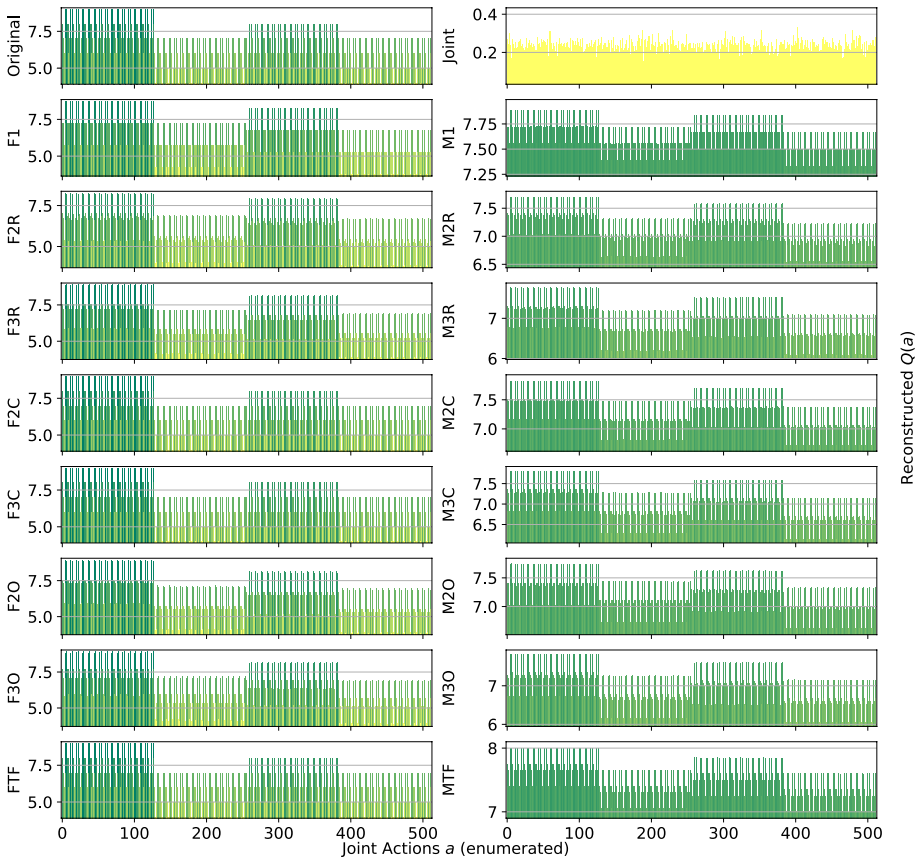


Fig. 16 Reconstructed $Q(a)$ for a different joint type of the Generalized Firefighting problem with $n = 9$ agents

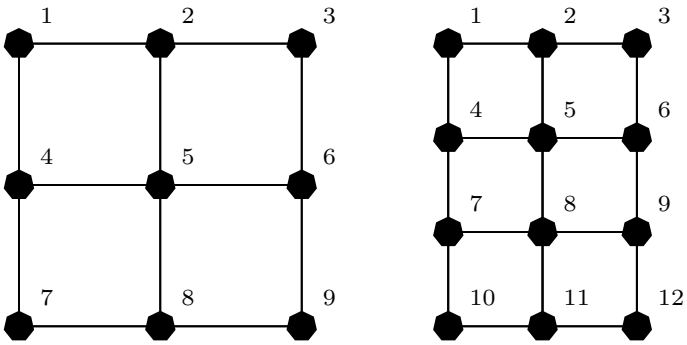
only $n = 6$ agents, is now resulting in a higher reconstruction error and fail in identifying all of the optimal joint actions. Methods using a complete factorization with both learning approaches instead are still able to identify most of them (all, when $n = 9$), while at the same time reducing the MSE considerably. Smaller factorizations are not reported because these aBLA BLA BLAre not achieving such good performances (as in the smaller case with $n = 6$), showing that on this kind of very tightly coordinated problems these may not suffice for a completely correct representation.

Generalized Firefighting: In this larger experiments, a team of $n = 9$ agents is fighting fire at $N_h = 10$ houses. As in the previous setting, each agent can observe $N_o = 2$ houses and can fight fire at the same set of locations ($N_a = 2$), as shown in Fig. 14. Reconstruction results for the joint type $\theta = \{N_1, F_2, F_3, N_4, F_5, F_6, N_7, N_8, F_9, F_{10}\}$ are reported in Fig. 15.

From these results, we can observe how, although the problem is very large (with more than half a million total joint actions in our formulation) most of the factored methods are perfectly representing the corresponding Q-function. While methods using the complete factorization or exploiting the true underlying structure with both learning approaches are capable of achieving a perfect reconstruction, even simpler methods like random pairing with the factored Q-function learning approach are

Table 12 Best (bold) and worst (italic) performing methods on the larger instance of the Generalized Firefighting problem

Model	MSE	Opt. Found	Ranked
Joint	*69.98 ± 0.9*	*2,556 ± 50*	*94,559 ± 522*
F2C	**0.00 ± 0.0**	**17,682 ± 0**	**524,288 ± 0**
F3C	**0.00 ± 0.0**	**17,682 ± 0**	**524,288 ± 0**
F3O	**0.09 ± 0.0**	**16,939 ± 415**	**464,893 ± 26,119**
FTF	**0.00 ± 0.0**	**17,682 ± 0**	**524,288 ± 0**
M2C	4.30 ± 0.0	**17,680 ± 0**	**465,019 ± 2**
M3C	2.71 ± 0.0	**17,680 ± 0**	**465,090 ± 2**
M3O	2.96 ± 0.1	**16,783 ± 303**	**298,182 ± 26,364**
MTF	5.30 ± 0.0	**17,682 ± 0**	**512,022 ± 810**



(a) $n = 9$ **(b)** $n = 12$

Fig. 17 Islands configuration for the two instances of Aloha

capable or almost perfectly reconstrut the values for this joint type. Conversely, the joint learner seems not capable of doing so, resulting in a totally wrong representation that is not close to the original function. Things are similar for a second joint type, $\theta = \{F_1, F_2, N_3, N_4, N_5, N_6, N_7, F_8, F_9, N_{10}\}$, whose resulting learned representations are shown in Fig. 16:

Again, the joint learner is not capable of achieving a good representation, but also some of the simpler factorizations are not resulting in a perfect reconstruction, although still capable of correctly identifying the optimal joint actions. Complete factorizations are instead perfectly representing the original Q-function for this joint type as well, even with the mixture of experts learning approach. General metrics and results for the best and worst performing methods on this problem are reported in Table 12.

As expected, the methods provided with the true underlying factorizations are performing best, with that using the factored Q-function learning approach capable of achieving a perfect reconstruction and ranking of the actions even on this very large problem. Also complete factorizations are always identifying all of the optimal joint actions and producing correct ranking (perfect for those using the factored Q-function learning approach). It is interesting to note that even overlapping factorizations, when coupled with larger factors, are performing very well, and can produce good rankings of the actions. As expected, the mixture of experts methods are resulting in a larger MSE, although being comparable on the other metrics with their counterparts, but are still

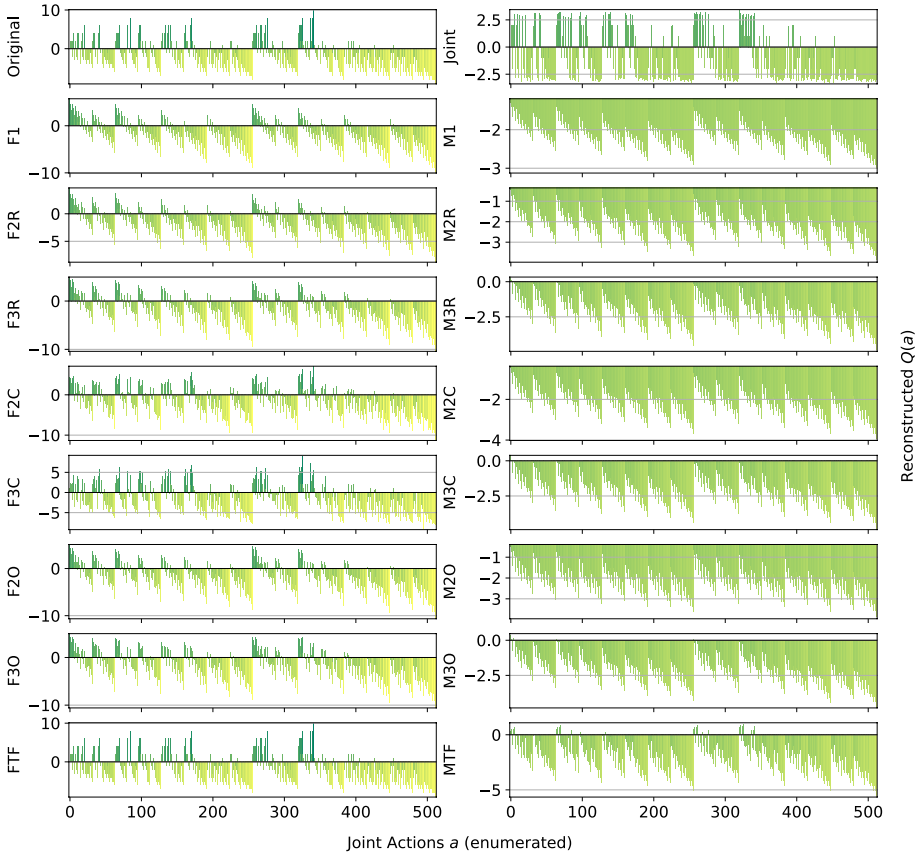


Fig. 18 Reconstructed $Q(a)$ for Aloha with $n = 9$ agents

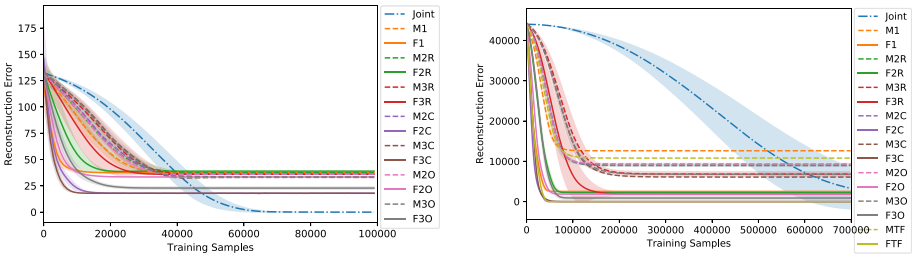
capable of learning more accutare representations than the joint learner, that is instead achieving the highest MSE and worst ranking among all the compared methods.

Aloha: Our experiments here use $n = 9$ and $n = 12$ islands disposed in a 3×3 and 4×3 grid respectively, as shown in Fig. 17. Representations learned for this game with $n = 9$ islands are reported in Fig. 18 (for identical reasons to those of the Dispersion Game, the figure with $n = 12$ agents is not included).

Again, this game proves to be challenging for almost all of the proposed factorizations. Indeed, other than the true underlying factorization coupled with the factored Q-function learning approach (that achieve a perfect reconstruction, showing how beneficial would it be to know and exploit such an underlying factorization in advance), only the complete factorizations seems able to learn something useful. All the other methods struggle to correctly identify the optimal action, probably because not enough coordination is achieved in order to discriminate between the two local actions for each agent (that seems similar from an agent perspective). Also for this game, the joint learner is not capable of correctly approximating the action-value function because of the increasing number of agents. Table 13 is showing the best and worst performing methods on this game.

Table 13 Best (bold) and worst (italic) performing methods on the two instances of Aloha

Model	MSE	Opt. Found	Ranked
Aloha $n = 9$			
F2R	6.97 ± 0.4	0 ± 0	*100 ± 11*
F2C	2.25 ± 0.0	**1 ± 0**	187 ± 1
FTF	**0.00 ± 0.0**	**1 ± 0**	**512 ± 0**
Ml	*11.93 ± 0.0*	0 ± 0	126 ± 1
M2R	*10.45 ± 0.2*	0 ± 0	*105 ± 11*
M2C	*10.31 ± 0.0*	0 ± 0	140 ± 1
M2O	*10.48 ± 0.2*	0 ± 0	*114 ± 10*
Aloha $n = 12$			
Joint	*23.98 ± 0.6*	0 ± 0	*700 ± 13*
F2C	2.18 ± 0.0	**2 ± 0**	$1{,}380 \pm 5$
F3C	**0.81 ± 0.0**	**1 ± 0**	**2,488 ± 10**
FTF	**0.00 ± 0.0**	**2 ± 0**	**4,096 ± 0**
Ml	*15.37 ± 0.0*	0 ± 0	845 ± 26
M2O	13.94 ± 0.2	0 ± 0	*671 ± 58*
M3O	12.56 ± 0.3	0 ± 0	*752 ± 87*



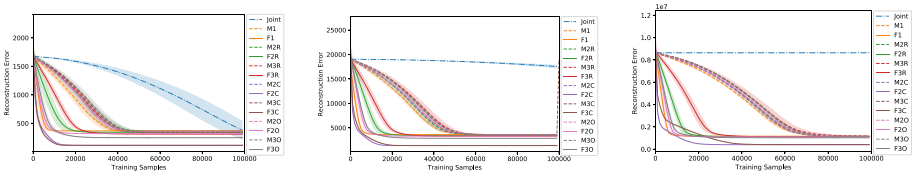
(a) Dispersion Game, $n = 6$ **(b)** Generalized Firefighting, $n = 6$

Fig. 19 Training curves for the investigated architectures on the two proposed problems

The table shows how, except for FTF (always capable of correctly represent the entire Q-function), all the methods start deteriorating their performance when the system size increases on this particular problem. Particularly, the joint learner achieves a very high reconstruction error and is not able to identify any of the optimal joint actions. On the other hand, although the corresponding ranking is not perfect, complete factorizations using the factored Q-function learning approach can identify such optimal actions. The mixture of experts instead are performing worse here, probably because the benefits of a coordinated optimization is crucial to correctly represent this problem.

4.5 Sample complexity

Another important consideration in multi-agent learning is sample complexity, as for example training data could be limited or expensive to obtain. Therefore it is a crucial aspect how efficiently we can use such data and how long does it take for a given representation



(a) $n = 9$ **(b)** $n = 12$ **(c)** $n = 20$

Fig. 20 Training curves for the investigated architectures on the Dispersion Game with an increasing number of agents

to converge, especially when the system grows larger in the number of agents. We expect factored representations to improve training efficiency, reducing the number of samples required to learn a good representation, as the size of the multiple components that have to be learnt is small compared to that of the overall problem. To show the benefits of using a factored representation, here we report in Fig. 19 the training curves for two of the proposed games, the Dispersion Game and the Generalized Firefighting, both with $n = 6$ agents.

Even for these problems of moderate size, factored approaches achieve a stable approximation of the action-value function with just a fraction of the given training time, while a full joint learner requires many more samples to get the same results. Especially, for the Generalized Firefighting problem (that has got more than 8000 overall joint actions), the joint learner achieves an accurate representation only after a much longer training time, while almost every factored architecture achieves a nearly perfect approximation with few samples, showing how the size of the joint action space is a critical problem that factored representations can help tackle. On one hand the mixture of experts approaches learn more slowly than the factored Q-function ones: each factor acts as an expert on its own, thus experiencing higher variance in the received rewards when performing a certain action. On the other hand, larger models learn more quickly, achieving the same final result as the smaller representations but with fewer samples. This could be due to the internal coordination happening inside each factor, helping the agents figure out their own contribution to the global reward, so that a stable representation is learned more easily. When the number of agents is larger, this benefit is even more apparent. Figure 20 shows the reconstruction error during the training process obtained on instances of the Dispersion Game with $n = \{9, 12, 20\}$ agents respectively.

We observe how the joint learner is struggling to achieve a good representation in the given training time when the size of the system increases, resulting in a higher reconstruction error. The increasingly large number of joint actions (more than 1 million with $n = 20$ agents) prevents it from converging in reasonable time, while the factored representations, although only approximating the original function, converge faster, as the size of each factor is small compared to that of the overall problem, and result in a lower reconstruction error.

4.6 Exploratory policy

Although we focused on a stationary uniform sampling of the actions throughout most of the paper, we also provide some preliminary results with a different, non-stationary action selection mechanism, more closely resembling those used in sequential MARL. We opted

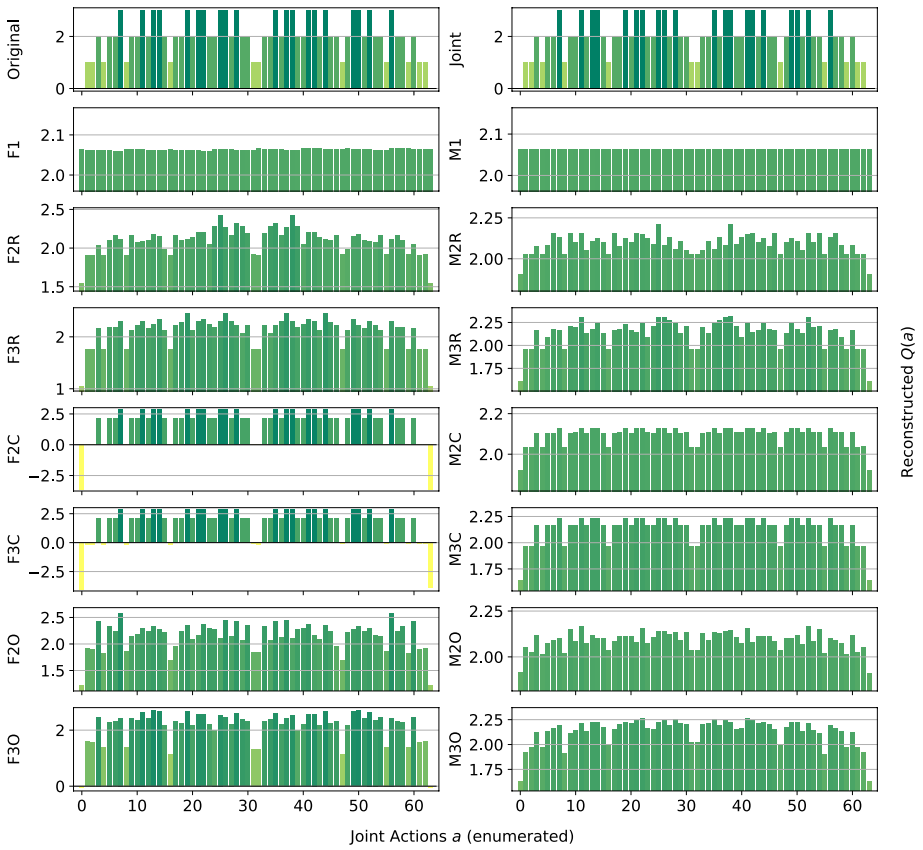


Fig. 21 Reconstructed $Q(a)$ for the Dispersion Game using the Boltzmann exploratory policy

for a Boltzmann policy [28] that, given a reconstruction of the action-value function $\hat{Q}(a)$, defines the probability for each joint action $a \in A$ to be selected as:

$$\pi(a) = \frac{e^{\hat{Q}(a)/\tau}}{\sum_{b \in A} e^{\hat{Q}(b)/\tau}}, \tag{14}$$

where τ is a temperature parameter governing the exploration rate. In our experiment, we set $\tau = 1$ for all methods. We choose to test these on the Dispersion Game with $n = 6$ agents, as many of the methods (including the joint learner) are doing reasonably well and thus any decrease in performance would be due to the new exploratory policy. For the factored methods and the independent learners, we reconstruct $\hat{Q}(a)$ at every step and then we apply the Boltzmann policy on this reconstruction. Figure 21 shows the learned reconstructions on this game.

If we compare the above to Fig. 2a, we can observe how the results are very much in line with those obtained under uniform sampling. Even if the joint action space is small enough to select each action a reasonable number of time, the fact that the policy is more frequently selecting the action that looks better is not providing benefits in term of accuracy of the final representation, especially for the independent learners, that are still not

able to clearly identify any of the optimal joint actions. The joint learners and the complete factorizations are still able to correctly rank all of the optimal actions, and those using the factored Q-function learning approach are achieving a smaller reconstruction error as under the uniform sampling. Although this is just a preliminary result, this is an important observation, as it gives more value to the previous results: if a method is not doing well under the uniform sampling of actions, it is unlikely that it can do better with a non-stationary, time-varying sampling mechanism like this.

4.7 Summary of results

We can gain many useful insights from the analysed results: first, we observe that the factorizations using mixture of expert learning approach, although generally achieving higher reconstruction errors than the factored Q-function counterparts, in many cases still results in a good approximation in terms of ranking of actions, therefore being a reliable choice for decision making. For example, on the two variants of the Dispersion Game, both M2C and M3C are able to correctly rank all of the joint actions, achieving better accuracy than some smaller factorizations like F2O or F3O, even with a higher mean square error of the reconstruction. This is probably due to the higher number of factors involved in their coordination graphs, allowing for better approximation of the true action-value function and coordination amongst the agents. We can therefore deduce how the number of factors used to learn an approximation is playing a major role in achieving accurate representations in terms of coordination and actions ranking.

Also, the size of these factors is an important aspect: as expected, with more agents comprised into each factor, the resulting approximation is more reliable because the agents into each factor are able to share information and thus better coordinate. This is reflected by both learning approaches, but it is even more apparent with the mixture of experts one, with the factorizations with 3 agents per factor usually achieving smaller reconstruction error and a better ranking of actions than their counterparts with only 2 agents for each. However, factors that are too large (with a size very similar to that of the entire team) do not always result in a better representation, but instead can present some of the difficulties associated with joint learners. This suggests that we can find an optimal tradeoff between the totally independent learners and the full joint learner extremes that is capable of achieving a reliable representation in a reasonable training time. Of course, depending on the intended use of such a learned representation, such a tradeoff may differ: for example, if we are only interested in selecting an optimal joint action from this reconstruction after the training process (i.e. with a factored centralized joint Q-function agent for the entire team of agents) a smaller factorization with fewer factors and agents per factor, that is faster to train and still able to correctly rank some of the joint actions, may suffice. Conversely, if we are approximating the critic of an actor-critic method, in which the values of the selected actions are in turn influencing the policies of the agents, we may prefer a bigger factorization with a lower reconstruction error.

Generally speaking, we can observe how the value of coordination is well captured by the factored Q-function learning approach, that usually produce good approximations and reduces the training time with respect to a joint learner, allowing to learn even games in which there is no underlying factorization and complete team coordination is required, like the Dispersion Game. This becomes even more important with larger systems, as now more agents have to coordinate and thus more accurate representation of such coordination requirements is needed. However, when these requirements are less tight, also the mixture

of experts learning approach is showing benefits, being faster to train and not requiring inter-factor communication during the training phase to optimize its objective. Overall, we showed how the choice of a suitable factorization to efficiently learn in a multi-agent system is a difficult decision that needs to be taken considering the problem structure and requirements. While many aspects can influence the learning outcome, our results have five main takeaways:

- There are some problematic examples, like the Platonia Dilemma, where all types of factorization with small factors result in selecting the worst possible joint action. Given that only joint learners (and certain factorizations with larger factors to some extent) seem to be able to address such problems, currently no scalable deep reinforcement learning methods for dealing with such systems seem to exist. We hypothesize that this is due to an imbalance in the frequency with which each local action for the agents leads to the optimal reward value. In the Platonia Dilemma, each agent is more frequently experiencing the positive reward if it does not send the telegram itself and leave this action to someone else. However, if all the agents do this kind of reasoning, no-one is sending the telegram, and the resulting reward is not the optimal one. Breaking this tie is possible when we behave greedily (as the first agent correctly sending the telegram will keep sending it more frequently), but learning a complete and correct representation of the entire Q-function remains challenging.
- Beyond those, "complete factorizations" of modest factor size coupled with the factored Q-function learning approach yield near-perfect reconstructions and rankings of the actions, also for non-factored action-value functions. Moreover, these methods scale much better than joint learners: for a given training time, we see that these complete factorizations already outperform fully joint learners on modestly sized problems, resulting in a correct ranking of the actions and a low reconstruction error. This is a compelling property that renders these methods more suited for large multi-agent systems with a large joint action-space and justify the recent interest in factored methods from the research community.
- For many problems with less tight coordination requirements such as Aloha and the Generalized Firefighting Problem, random overlapping factors also achieve excellent performance, comparable to those of more computationally complex methods like joint learners and complete factorizations. This suggests that such approaches are a promising direction forward for scalable deep MARL in many problem settings.
- Factorizations with the mixture of experts approach usually perform somewhat worse than the corresponding factored Q-function approaches, as these are less able to capture the coordination requirements of certain problems. However, in some cases they perform better or comparably (Dispersion Game, Generalized Firefighting), in which M2R and M3R still outperform F1 (i.e., VDNs). This is promising, because the mixture of experts learning approach does not require any exchange of information between the neural networks, thus potentially facilitating learning in settings with communication constraints, and making it easier to parallelize across on multiple CPUs/GPUs.
- Our results show that, when facing larger multi-agent systems, factored representations retain many of their benefits and can still represent the action-value function correctly in many settings (or at least identify most if not all of the optimal joint actions), while both independent learners and centralized approaches tend to quickly deteriorate, resulting in wrong or incomplete representations. The reasons however are different: while independent learners still decompose the entire function into small components that can easily be learned, these fail in representing the value of coordinated decisions

when more agents are comprised into the system. On the other hand instead, a joint learner that is representing the centralized action-value function as a single component (and thus does not introduce any approximation), is now struggling to learn a correct representation because of the exponential number of joint actions. Factored representations instead learn small components easily, but take into account the value of coordination into each component, allowing for better final representations. This is an extremely desirable property that once more points out how these methods deserve attention as holding a great potential.

These observations also shed some light on the performance of independent learners in MARL, as used by many modern deep MARL algorithms [35, 40]: while these can outperform joint learners on large problems, the degree of independence and the final outcome is hard to predict and is affected by different factors. Designing algorithms that are able to overcome these difficulties should be a primary focus of MARL research.

5 Discussion

The aim of our analysis was to investigate the learning capabilities of factored representations and compare them to both independent learners and joint learners, to assess eventual benefits of such representations both in terms on learning speed and final accuracy of the reconstructed representations. In order to do so, we consider different aspects of these learned representations $\hat{Q}$: we consider the optimality of the greedy joint action, which is important when using $\hat{Q}$ to select actions. We also consider the distance to the optimal value $\Delta Q = |Q - \hat{Q}|$, since verifying the optimality of the greedy action requires bounding ΔQ. Although our approach focus on centralized learning, a correct representation of the joint action-value function can be used in the CTDE framework [22] to learn improved decentralized policies. Indeed, minimising ΔQ is important for deriving good policy gradients in actor-critic architectures (for example when computing the counterfactual baseline in [9], requiring very accurate estimates of many sub-optimal Q-values) and for sequential value estimation in any approach that relies on bootstrapping (such as Q-learning [44]) or message passing of local payoff values (like the max-sum algorithm [20]), where such values are used to update other values and thus need to be as accurate as possible.

Our analysis is focused on cooperative one-shot games. We chose these because, although they are a simpler setting than standard sequential problems, they still capture many of the aspects that can be problematic in MARL, like the exponentially large number of joint actions. Also, the shared reward observed by all the agents depends on the joint action of the whole team, and thus seems non-stationary from an agent perspective. Therefore, all of the problems that arise from these conditions are directly translated into our setting, and by removing the fostering effect of states, we are better able to analyse how the different methods can tackle these issues. Moreover, we mainly focused on using a stationary uniform sampling mechanism to select joint actions, thus not directly considering the exploration-exploitation tradeoff usually faced in MARL problems. There are multiple reasons for this: on one hand, using such a simple mechanism makes our comparison easier, as we are interested in the reconstruction of the entire action-value function (and not only the greedy action). On the other hand, if a network architecture is not suited to learn and represent accurate action-values under our stationary uniform sampling, it is extremely unlikely that the same is going to perform better under a more complex, time-changing policy as in

sequential MARL. This is validated by our preliminary results with a Boltzmann policy, in which none of the method is achieving better accuracy than under the stationary uniform mechanism. However, while good performance in such one-shot settings does not necessarily imply good performance in the sequential setting, Q-value-based approaches aim to transform the sequential MARL problem to precisely the one-shot decision making problem that we investigate, as the sequential problem simply becomes a one-shot maximization over the action-values. This implies that any limitations we find are likely to directly transfer to such approaches.

Obviously, the opposite is not immediately true: good performance on this setting does not directly imply also good performance on general MARL problems. However, if a given representation is able to correctly capture the value of coordination and deal with the joint action set size in one-shot games, we can hypothesize that these benefits are likely to hold in sequential problems as well, where the same requirements usually brings similar challenges to our setting and thus can be tackled with similar solutions. Neural networks are known to be very good at dealing with large input spaces, but it is not so well known how these can deal with large output ones, like those resulting from exponential joint action sets. Therefore, our analysis moves an important step in this direction in the field of MARL, and has to be considered as an initial step toward a proper understanding of action-value functions in multi-agent settings. Of course, a direct investigation in full sequential settings is an interesting future direction, with key questions that are orthogonal to ours, like the presence of an input state of the exploration-exploitation tradeoff. Nonetheless, our results can still help taking informed decision in such problems as well, as they give us interesting insights and takeaways that practitioners of the field may take into account when taking informed decisions in designing multi-agent systems.

6 Related work

Recently, many works have applied deep reinforcement learning techniques to multi-agent systems, achieving great performance. Gupta et al. [16] compare the performance of many standard deep reinforcement learning algorithms (like DQN, DDPG and TRPO) using a variety of learning schemes (joint learners, fully independent learners, etc.) on both discrete and continuous tasks, assessing and comparing their performance. Tampuu et al. [41] present a variation of DQN capable of dealing with both competitive and cooperative settings in Pong. Applications of techniques to enhance the learning process have also been investigated: Palmer et al. [34] apply leniency to independent double DQN learners in a coordinated gridworld problem, while Foerster et al. [8] propose a novel approach to stabilize the experience replay buffer in DQNs by conditioning on when the samples were collected, thereby easing the non-stationarity of independent learners. Communication between agents has also been explored: Sukhbaatar et al. [39] investigate the emergence of a communication mechanism that can be directly learned through backpropagation. However, none of these works compare alternative representations of Q-values for such networks. Another interesting line of work is that of agent definition: Chung et al. [5] analyse how different agents definitions in a multi-agent system affect how efficiently these can be learned and modelled.

Close to our idea of factorization are [13, 20, 21]. Guestrin et al. [13] present different reinforcement learning algorithms, based either on Q-learning, policy iteration or direct policy search, that exploit a coordination graph representation of the problem and the variable

elimination algorithm [12] to avoid enumerating all the possible joint actions. Kok and Vlassis [20] investigate the max-sum algorithm, an approximation of variable elimination, in which agents exchange messages with an estimate of their local payoff function over a coordination graph used by other agents to update their own estimate until they all converge to a stable value. They then also introduce a modified version of Q-learning updates based on either an agent-based decomposition or an edge-based decomposition of the coordination graph. Kok et al. [21] instead present a method to learn a coordination graph structure for the agents to use. It works by maintaining statistics of a state future returns and the incoming transitions, and then splitting the state into a factored representation if these returns are significantly different. More recently, Boehmer et al. [2] propose Deep Coordination Graphs, published after our original work [4] and extending on it, in which different factors of a coordination graph are modelled with a single neural network through parameter sharing and uses a learned embedding of their input to differentiate. The results presented show the effectiveness of such a factored representation in cooperative sequential problems, but not a systematic analysis of the learned factored Q-functions.

Another line of work addressing the problem of the exponentially large joint action space exploits the paradigm of centralized learning with decentralized execution. Foerster et al. [9] present COMA, an architecture based on the actor-critic framework with multiple independent actors but a single centralized critic used to efficiently estimate both Q-values and a counterfactual baseline to tackle the credit assignment problem and guide the agents through the learning process under partial observability. On the other hand, Lowe et al. [26] propose MADDPG, expanding DDPG [24] to work in multi-agent systems by maintaining a different centralized critic network for every actor that estimate Q-values for each agents, considering the actions and observations of other agents, and apply their approach on both cooperative and competitive continuous tasks. However, these works still represents $Q(s, a)$ monolithically, and thus can experience scalability issues. Sunehag et al. [40] address the problem with value decomposition networks, training a set of independent agents collectively by using a value decomposition method that represents the original Q-function as the sum of local terms depending only on agent-wise information, while Rashid et al. [35] extend this idea in QMIX by representing the Q-function using a monotonic non-linear combination of individual Q-values, weighted with an additional mixing network conditioned on the global state signal, so that the maximization step can still be performed in an efficient way. While such mixing networks may lead to more accurate Q-values in the sequential domain, our investigation shows that for many coordination problems, individual Q-components may not suffice. Son et al. [38] propose QTRAN, an algorithm that transform a joint action-value function into another one with the same optimal joint action that can be easily factorized in individual components so that a monotonicity constraint is respected and thus action selection can be performed locally. Mahajan et al. [27] investigate the effect of poor exploration in value-based methods and propose MAVEN, an hybrid approach that improve value-based methods by conditioning them on a shared latent variable controlled by a hierarchical policy to improve over standard exploration methods, resulting in extended temporal exploration and improved performance.

7 Conclusions

In this work, we investigated how well neural networks can represent action-value functions arising in cooperative multi-agent systems. This is an important question since accurate representations can enable taking (near-) optimal actions in value-based

approaches using algorithms like variable elimination or max-sum, thus avoiding the maximization over an exponential number of joint actions, and computing good gradient estimates in actor-critic methods. In this paper, we focused on one-shot games as the simplest setting that captures the exponentially large joint action space of multi-agent systems. We compared a number of existing and new action-value network factorizations and learning approaches.

Our results highlight the difficulty of compactly representing action values in problems that require tight coordination, but indicate that using "higher-order" factorizations with multiple agents in each factor can improve the accuracy of these representations substantially. We also demonstrate that there are non-trivial coordination problems, some without a factored structure, that can be tackled quite well with simpler factorizations. Intriguingly, random overlapping factors perform very well in several settings. There are also settings where the mixtures of experts approach, with its low communication requirements and amenability to parallelization, is competitive in terms of the final reconstructions.

While our results emphasize the dependence of appropriate architectural choices on the problem at hand, our analysis also shows general trends that can help in the design of novel algorithms and improve general performance, highlighting how the use of factored action-value functions can be a viable way to obtain good representations without incurring in excessive costs.

As future work, we intend to extend this investigation to more complex settings and networks types. For example, investigating the learning power of factorizations using DQNs [29], a standard value-based choice when modelling agents with neural networks, to sequential multi-agent problems [25] seems a natural direction to try and assess what deep reinforcement learning algorithms are really learning and which problems and situations cause them to fail, while also an investigation of recurrent neural networks [17] as a tool to deal with partial observability in the popular setting of repeated games [33] can be beneficial for the multi-agent community.

Complete results

Table 14 presents the accuracy of the investigated representations using various measures, both in terms of action ranking and reconstruction error, as well as evaluating the action selection that these representations result in:

- To evaluate the reconstruction error, we compute the mean square error over all the joint actions (1st column);
- We also compute the same measure but restricted only to those actions that are optimal in the original action-value function (2nd column);
- We assess how many optimal actions are correctly identified by the reconstructions (3rd column);
- The value loss (regret) obtained by following the represented value functions when doing decision making (4th column);
- We also provide a different version of this regret, that we call *Boltzmann value loss* which expresses the value loss obtained by the expected reward accrued by defining a softmax distribution over all the joint actions (5th column). This gives an indication of value loss amongst all good actions; and

Table 14 Accuracy results with respect to both action ranking and reconstruction error for the different games. Best (bold) and worst (italic) performances for each game are highlighted

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
Dispersion Game $n = 6$ **(64 joint actions, 20 optimal)**							
Joint	**0.00 ± 0.0**	0.00 ± 0.0	**20 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**64 ± 0**	1.00 ± 0.0
F1	*0.62 ± 0.0*	0.88 ± 0.0	*5 ± 2*	1.70 ± 1.0	0.49 ± 0.0	*21 ± 2*	0.00 ± 0.0
F2R	*0.52 ± 0.0*	0.82 ± 0.0	*8 ± 0*	0.00 ± 0.0	0.38 ± 0.0	*24 ± 1*	0.27 ± 0.0
F3R	0.41 ± 0.0	0.72 ± 0.0	10 ± 1	0.60 ± 0.5	0.31 ± 0.0	31 ± 3	0.39 ± 0.0
F2C	**0.09 ± 0.0**	0.14 ± 0.0	**20 ± 0**	0.00 ± 0.0	0.16 ± 0.0	**64 ± 0**	1.00 ± 0.0
F3C	**0.09 ± 0.0**	0.14 ± 0.0	**20 ± 0**	0.00 ± 0.0	0.16 ± 0.0	**64 ± 0**	1.00 ± 0.0
F2O	0.41 ± 0.0	0.64 ± 0.0	10 ± 1	0.60 ± 0.5	0.32 ± 0.0	36 ± 2	0.44 ± 0.0
F3O	**0.19 ± 0.0**	0.30 ± 0.0	13 ± 1	0.20 ± 0.4	0.20 ± 0.0	47 ± 3	0.70 ± 0.0
M1	*0.62 ± 0.0*	0.88 ± 0.0	*6 ± 1*	1.30 ± 0.8	0.49 ± 0.0	*24 ± 2*	0.02 ± 0.0
M2R	*0.56 ± 0.0*	0.82 ± 0.0	*8 ± 0*	0.00 ± 0.0	0.46 ± 0.0	*24 ± 1*	0.27 ± 0.0
M3R	0.46 ± 0.0	0.73 ± 0.0	10 ± 1	0.40 ± 0.5	0.39 ± 0.0	31 ± 2	0.39 ± 0.0
M2C	*0.55 ± 0.0*	0.81 ± 0.0	**20 ± 0**	0.00 ± 0.0	0.46 ± 0.0	**64 ± 0**	1.00 ± 0.0
M3C	0.43 ± 0.0	0.68 ± 0.0	**20 ± 0**	0.00 ± 0.0	0.40 ± 0.0	**64 ± 0**	1.00 ± 0.0
M2O	*0.56 ± 0.0*	0.81 ± 0.0	10 ± 2	0.50 ± 0.5	0.46 ± 0.0	36 ± 4	0.46 ± 0.0
M3O	0.44 ± 0.0	0.69 ± 0.0	11 ± 1	0.30 ± 0.5	0.40 ± 0.0	40 ± 3	0.58 ± 0.1
Dispersion Game (sparse) $n = 6$ **(64 joint actions, 20 optimal)**							
Joint	**0.00 ± 0.0**	0.00 ± 0.0	**20 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**64 ± 0**	1.00 ± 0.0
F1	*1.93 ± 0.0*	4.25 ± 0.0	*7 ± 1*	1.50 ± 1.5	1.77 ± 0.0	*37 ± 1*	0.02 ± 0.0
F2R	*1.83 ± 0.0*	3.95 ± 0.0	*8 ± 0*	0.00 ± 0.0	1.64 ± 0.0	40 ± 0	0.13 ± 0.0
F3R	1.72 ± 0.0	3.60 ± 0.0	11 ± 1	1.80 ± 1.5	1.55 ± 0.0	45 ± 1	0.31 ± 0.0
F2C	1.41 ± 0.0	2.25 ± 0.0	**20 ± 0**	0.00 ± 0.0	1.32 ± 0.0	**64 ± 0**	1.00 ± 0.0
F3C	1.41 ± 0.0	2.26 ± 0.0	**20 ± 0**	0.00 ± 0.0	1.32 ± 0.0	**64 ± 0**	1.00 ± 0.0
F2O	1.72 ± 0.0	3.53 ± 0.0	10 ± 1	1.50 ± 1.5	1.56 ± 0.0	45 ± 3	0.30 ± 0.1
F3O	1.52 ± 0.0	2.71 ± 0.1	12 ± 1	0.60 ± 1.2	1.41 ± 0.0	49 ± 2	0.45 ± 0.1
M1	*1.93 ± 0.0*	4.27 ± 0.0	*6 ± 1*	2.10 ± 1.4	1.77 ± 0.0	*37 ± 2*	0.01 ± 0.1

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
M2R	*1.88 ± 0.0*	4.13 ± 0.0	*8 ± 0*	0.00 ± 0.0	1.73 ± 0.0	40 ± 0	0.13 ± 0.0
M3R	1.77 ± 0.0	3.88 ± 0.0	10 ± 1	1.50 ± 1.5	1.66 ± 0.0	44 ± 3	0.27 ± 0.1
M2C	*1.87 ± 0.0*	4.11 ± 0.0	**20 ± 0**	0.00 ± 0.0	1.73 ± 0.0	**64 ± 0**	1.00 ± 0.0
M3C	1.74 ± 0.0	3.83 ± 0.0	**20 ± 0**	0.00 ± 0.0	1.66 ± 0.0	**64 ± 0**	1.00 ± 0.0
M2O	*1.87 ± 0.0*	4.13 ± 0.0	10 ± 1	1.20 ± 1.5	1.73 ± 0.0	44 ± 2	0.26 ± 0.1
M3O	1.75 ± 0.0	3.84 ± 0.0	12 ± 1	0.60 ± 1.2	1.67 ± 0.0	47 ± 2	0.38 ± 0.1
Platonia Dilemma $n = 6$ **(64 joint actions, 6 optimal)**							
Joint	**0.00 ± 0.0**	0.03 ± 0.1	**6 ± 0**	0.00 ± 0.0	0.01 ± 0.0	**64 ± 0**	1.00 ± 0.0
F1	2.22 ± 0.0	15.55 ± 0.1	5 ± 0	6.00 ± 0.0	4.19 ± 0.0	62 ± 0	0.82 ± 0.0
F2R	2.11 ± 0.0	14.17 ± 0.1	5 ± 0	6.00 ± 0.0	4.09 ± 0.0	62 ± 0	0.82 ± 0.0
F3R	2.00 ± 0.0	12.90 ± 0.1	5 ± 0	6.00 ± 0.0	4.04 ± 0.0	62 ± 0	0.82 ± 0.0
F2C	1.69 ± 0.0	8.92 ± 0.1	5 ± 0	6.00 ± 0.0	4.39 ± 0.0	62 ± 0	0.82 ± 0.0
F3C	1.69 ± 0.0	8.97 ± 0.1	5 ± 0	6.00 ± 0.0	4.52 ± 0.0	62 ± 0	0.82 ± 0.0
F2O	2.00 ± 0.0	12.83 ± 0.1	5 ± 0	6.00 ± 0.0	3.94 ± 0.0	62 ± 1	0.78 ± 0.1
F3O	1.78 ± 0.0	10.10 ± 0.4	5 ± 0	6.00 ± 0.0	4.15 ± 0.1	62 ± 0	0.82 ± 0.0
M1	*2.80 ± 0.0*	27.01 ± 0.0	5 ± 0	6.00 ± 0.0	5.15 ± 0.0	62 ± 0	0.82 ± 0.0
M2R	2.53 ± 0.0	23.93 ± 0.0	5 ± 0	6.00 ± 0.0	4.93 ± 0.0	62 ± 0	0.82 ± 0.0
M3R	2.28 ± 0.0	20.51 ± 0.1	5 ± 0	6.00 ± 0.0	4.64 ± 0.0	62 ± 0	0.82 ± 0.0
M2C	2.52 ± 0.0	23.90 ± 0.0	5 ± 0	6.00 ± 0.0	4.92 ± 0.0	62 ± 0	0.82 ± 0.0
M3C	2.25 ± 0.0	20.54 ± 0.0	5 ± 0	6.00 ± 0.0	4.62 ± 0.0	62 ± 0	0.82 ± 0.0
M2O	2.54 ± 0.0	23.93 ± 0.1	*4 ± 0*	6.00 ± 0.0	4.92 ± 0.0	*61 ± 1*	0.69 ± 0.1
M3O	2.28 ± 0.0	20.60 ± 0.1	*4 ± 0*	6.00 ± 0.0	4.61 ± 0.0	*61 ± 1*	0.71 ± 0.1
Climb Game $n = 6$ **(729 joint actions, 1 optimal)**							
Joint	**0.17 ± 0.1**	18.45 ± 4.9	0 ± 0	2.70 ± 0.9	1.52 ± 0.3	727 ± 1	1.00 ± 0.0
F1	0.58 ± 0.0	52.29 ± 0.1	0 ± 0	3.00 ± 0.0	2.16 ± 0.0	726 ± 0	0.98 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
F2R	0.52 ± 0.0	40.95 ± 0.0	0 ± 0	3.00 ± 0.0	2.06 ± 0.0	726 ± 0	0.98 ± 0.0
F3R	0.44 ± 0.0	36.51 ± 0.2	0 ± 0	3.00 ± 0.0	1.92 ± 0.0	726 ± 0	0.98 ± 0.0
F2C	$\mathbf{0.25 \pm 0.0}$	7.86 ± 0.1	$\mathbf{1 \pm 0}$	0.00 ± 0.0	1.40 ± 0.0	$\mathbf{729 \pm 0}$	1.00 ± 0.0
F3C	$\mathbf{0.17 \pm 0.0}$	70.77 ± 0.7	0 ± 0	3.00 ± 0.0	0.96 ± 0.0	726 ± 0	0.98 ± 0.0
F2O	0.45 ± 0.0	30.83 ± 0.1	0 ± 0	3.00 ± 0.0	1.94 ± 0.0	726 ± 0	0.98 ± 0.0
F3O	0.30 ± 0.0	28.89 ± 1.9	0 ± 0	3.00 ± 0.0	1.54 ± 0.0	726 ± 0	0.98 ± 0.0
M1	*0.71 ± 0.0*	35.91 ± 0.0	0 ± 0	3.00 ± 0.0	2.36 ± 0.0	726 ± 0	0.98 ± 0.0
M2R	0.63 ± 0.0	35.77 ± 0.0	0 ± 0	3.00 ± 0.0	2.30 ± 0.0	726 ± 0	0.98 ± 0.0
M3R	0.53 ± 0.0	35.34 ± 0.1	0 ± 0	3.00 ± 0.0	2.20 ± 0.0	726 ± 0	0.98 ± 0.0
M2C	0.62 ± 0.0	35.77 ± 0.0	0 ± 0	3.00 ± 0.0	2.30 ± 0.0	726 ± 0	0.98 ± 0.0
M3C	0.51 ± 0.0	35.30 ± 0.1	0 ± 0	3.00 ± 0.0	2.20 ± 0.0	726 ± 0	0.98 ± 0.0
M2O	0.63 ± 0.0	35.74 ± 0.0	0 ± 0	3.00 ± 0.0	2.30 ± 0.0	726 ± 0	0.98 ± 0.0
M3O	0.52 ± 0.0	35.31 ± 0.1	0 ± 0	3.00 ± 0.0	2.20 ± 0.0	726 ± 0	0.98 ± 0.0
Penalty Game $n = 6$ **(729 joint actions, 2 optimal)**							
Joint	1.60 ± 0.4	18.21 ± 4.3	$\mathbf{1 \pm 0}$	0.90 ± 1.4	3.24 ± 0.1	$\mathbf{727 \pm 1}$	1.00 ± 0.0
F1	2.18 ± 0.0	63.71 ± 0.1	0 ± 0	3.00 ± 0.0	3.31 ± 0.0	*722 ± 0*	0.91 ± 0.0
F2R	2.00 ± 0.0	65.95 ± 0.3	0 ± 0	3.00 ± 0.0	3.33 ± 0.0	723 ± 0	0.92 ± 0.0
F3R	1.75 ± 0.0	66.27 ± 1.0	0 ± 0	3.00 ± 0.0	3.31 ± 0.0	723 ± 0	0.94 ± 0.0
F2C	$\mathbf{1.29 \pm 0.0}$	79.66 ± 0.0	0 ± 0	6.00 ± 0.0	3.30 ± 0.0	*722 ± 0*	0.92 ± 0.0
F3C	$\mathbf{0.54 \pm 0.0}$	82.77 ± 0.0	0 ± 0	3.00 ± 0.0	2.06 ± 0.0	724 ± 0	0.97 ± 0.0
F2O	1.82 ± 0.0	68.81 ± 0.3	0 ± 0	6.00 ± 0.0	3.32 ± 0.0	723 ± 0	0.95 ± 0.0
F3O	$\mathbf{1.27 \pm 0.0}$	73.48 ± 1.4	0 ± 0	6.00 ± 0.0	3.29 ± 0.0	723 ± 0	0.95 ± 0.0
M1	*2.71 ± 0.0*	45.27 ± 0.1	0 ± 0	3.00 ± 0.0	3.66 ± 0.0	*722 ± 0*	0.91 ± 0.0
M2R	2.43 ± 0.0	49.11 ± 0.2	0 ± 0	3.00 ± 0.0	3.54 ± 0.0	723 ± 0	0.93 ± 0.0
M3R	2.09 ± 0.0	52.23 ± 0.8	0 ± 0	3.00 ± 0.0	3.43 ± 0.0	723 ± 0	0.94 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
M2C	2.41 ± 0.0	49.12 ± 0.1	0 ± 0	3.00 ± 0.0	3.54 ± 0.0	724 ± 0	0.97 ± 0.0
M3C	2.02 ± 0.0	52.45 ± 0.2	0 ± 0	3.00 ± 0.0	3.43 ± 0.0	724 ± 0	0.97 ± 0.0
M2O	2.43 ± 0.0	49.11 ± 0.1	0 ± 0	3.00 ± 0.0	3.54 ± 0.0	724 ± 0	0.97 ± 0.0
M3O	2.06 ± 0.0	52.56 ± 0.5	0 ± 0	3.00 ± 0.0	3.43 ± 0.0	723 ± 2	0.96 ± 0.0
Generalized Firefighting $n = 6$ **(8192 joint actions, 779 optimal)**							
Joint	1.29 ± 2.5	4.96 ± 7.2	*656 ± 123*	61.60 ± 68.2	46.42 ± 48.4	6,893 ± 1,475	0.85 ± 0.2
F1	0.16 ± 0.0	0.20 ± 0.0	700 ± 7	26.20 ± 7.3	6.38 ± 0.1	6,236 ± 38	0.88 ± 0.0
F2R	0.12 ± 0.0	0.15 ± 0.0	722 ± 19	16.80 ± 8.5	4.78 ± 1.4	6,777 ± 383	0.91 ± 0.0
F3R	**0.09 ± 0.0**	0.11 ± 0.1	743 ± 25	11.10 ± 10.2	3.42 ± 1.7	7,288 ± 558	0.94 ± 0.0
F2C	**0.00 ± 0.0**	0.00 ± 0.0	**779 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**8,192 ± 0**	1.00 ± 0.0
F3C	**0.00 ± 0.0**	0.00 ± 0.0	**779 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**8,192 ± 0**	1.00 ± 0.0
F2O	**0.09 ± 0.0**	0.10 ± 0.0	747 ± 18	8.00 ± 7.1	3.75 ± 1.1	7,333 ± 382	0.95 ± 0.0
F3O	**0.03 ± 0.0**	0.03 ± 0.0	**778 ± 4**	0.20 ± 0.6	1.14 ± 0.9	**8,149 ± 130**	1.00 ± 0.0
FTF	**0.00 ± 0.0**	0.00 ± 0.0	**779 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**8,192 ± 0**	1.00 ± 0.0
M1	*3.55 ± 0.0*	8.35 ± 0.0	700 ± 6	27.80 ± 6.4	163.84 ± 0.1	6,220 ± 30	0.88 ± 0.0
M2R	1.85 ± 0.1	4.92 ± 0.2	718 ± 12	20.60 ± 5.4	124.61 ± 0.6	6,602 ± 301	0.90 ± 0.0
M3R	1.09 ± 0.2	2.81 ± 0.2	739 ± 33	14.60 ± 13.8	88.58 ± 2.3	7,097 ± 703	0.93 ± 0.0
M2C	1.82 ± 0.0	4.90 ± 0.0	**777 ± 0**	0.00 ± 0.0	124.42 ± 0.1	**7,826 ± 0**	0.97 ± 0.0
M3C	0.85 ± 0.0	2.58 ± 0.0	**778 ± 1**	0.00 ± 0.0	88.09 ± 0.1	**8,151 ± 4**	1.00 ± 0.0
M2O	1.97 ± 0.1	5.08 ± 0.1	741 ± 13	11.70 ± 5.0	127.21 ± 1.9	*5,628 ± 249*	0.84 ± 0.0
M3O	1.73 ± 1.2	4.96 ± 3.3	738 ± 55	8.70 ± 14.0	97.67 ± 10.9	*5,867 ± 682*	0.85 ± 0.1
MTF	*2.60 ± 0.0*	6.14 ± 0.0	**779 ± 0**	0.00 ± 0.0	133.79 ± 0.1	**8,177 ± 2**	1.00 ± 0.0
Aloha $n = 6$ **(64 joint actions, 2 optimal)**							
Joint	1.13 ± 0.0	0.00 ± 0.0	**2 ± 0**	0.00 ± 0.0	0.08 ± 0.0	51 ± 1	0.88 ± 0.0
F1	4.78 ± 0.0	50.93 ± 0.1	0 ± 0	6.00 ± 0.0	4.04 ± 0.0	27 ± 1	0.67 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
F2R	4.05 ± 0.4	35.00 ± 7.0	0 ± 0	5.00 ± 1.3	3.69 ± 0.4	*22 ± 4*	0.70 ± 0.0
F3R	3.16 ± 0.5	20.64 ± 4.6	0 ± 0	4.20 ± 1.4	3.23 ± 0.9	*26 ± 4*	0.74 ± 0.0
F2C	0.91 ± 0.0	0.14 ± 0.0	**2 ± 0**	0.00 ± 0.0	-0.04 ± 0.0	42 ± 0	0.89 ± 0.0
F3C	**0.07 ± 0.0**	0.14 ± 0.0	**2 ± 0**	0.00 ± 0.0	0.22 ± 0.0	**64 ± 0**	1.00 ± 0.0
F2O	3.27 ± 0.3	20.63 ± 3.0	0 ± 0	4.40 ± 1.2	3.24 ± 0.5	*23 ± 4*	0.74 ± 0.0
F3O	1.46 ± 0.3	3.55 ± 1.3	**1 ± 1**	0.80 ± 1.3	1.19 ± 0.4	29 ± 5	0.83 ± 0.0
FTF	**0.00 ± 0.0**	0.00 ± 0.0	**2 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**64 ± 0**	1.00 ± 0.0
M1	*8.26 ± 0.0*	50.84 ± 0.1	0 ± 0	6.00 ± 0.0	5.47 ± 0.0	27 ± 1	0.67 ± 0.0
M2R	6.52 ± 0.2	44.17 ± 1.5	0 ± 0	5.00 ± 1.3	4.57 ± 0.1	*25 ± 4*	0.70 ± 0.0
M3R	4.53 ± 0.5	31.35 ± 3.8	0 ± 0	4.00 ± 2.2	3.41 ± 0.6	28 ± 5	0.77 ± 0.1
M2C	6.51 ± 0.0	44.65 ± 0.1	0 ± 0	6.00 ± 0.0	4.59 ± 0.0	28 ± 0	0.76 ± 0.0
M3C	4.56 ± 0.0	33.45 ± 0.1	0 ± 0	6.00 ± 0.0	3.62 ± 0.0	36 ± 1	0.86 ± 0.0
M2O	6.63 ± 0.4	44.51 ± 1.1	0 ± 0	5.20 ± 1.0	4.62 ± 0.2	*22 ± 5*	0.66 ± 0.0
M3O	4.71 ± 0.3	33.36 ± 1.2	0 ± 0	5.20 ± 1.0	3.65 ± 0.2	*25 ± 4*	0.74 ± 0.0
MTF	3.20 ± 0.0	23.88 ± 0.1	0 ± 0	2.00 ± 0.0	2.78 ± 0.0	36 ± 1	0.88 ± 0.0
Dispersion Game $n = 9$ **(512 joint actions, 252 optimal)**							
Joint	*0.93 ± 0.5*	1.74 ± 0.9	162 ± 30	0.00 ± 0.0	0.25 ± 0.1	307 ± 77	0.49 ± 0.2
F1	0.74 ± 0.0	0.53 ± 0.0	*124 ± 1*	1.10 ± 0.9	0.47 ± 0.0	*191 ± 3*	0.02 ± 0.0
F2R	0.66 ± 0.0	0.53 ± 0.0	143 ± 2	0.00 ± 0.0	0.40 ± 0.0	206 ± 3	0.19 ± 0.0
F3R	0.57 ± 0.0	0.51 ± 0.0	171 ± 2	0.40 ± 0.5	0.32 ± 0.0	279 ± 3	0.37 ± 0.0
F2C	**0.06 ± 0.0**	0.03 ± 0.0	**252 ± 0**	0.00 ± 0.0	0.09 ± 0.0	**512 ± 0**	1.00 ± 0.0
F3C	**0.06 ± 0.0**	0.03 ± 0.0	**252 ± 0**	0.00 ± 0.0	0.09 ± 0.0	**512 ± 0**	1.00 ± 0.0
F2O	0.57 ± 0.0	0.49 ± 0.0	159 ± 4	0.20 ± 0.4	0.32 ± 0.0	254 ± 9	0.35 ± 0.0
F3O	0.36 ± 0.0	0.33 ± 0.0	178 ± 4	0.10 ± 0.3	0.21 ± 0.0	305 ± 14	0.54 ± 0.0
M1	0.74 ± 0.0	0.53 ± 0.0	*124 ± 1*	0.90 ± 0.8	0.47 ± 0.0	*192 ± 4*	0.01 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
M2R	0.70 ± 0.0	0.52 ± 0.0	142 ± 2	0.00 ± 0.0	0.45 ± 0.0	204 ± 5	0.19 ± 0.0
M3R	0.64 ± 0.0	0.48 ± 0.0	170 ± 1	0.30 ± 0.5	0.42 ± 0.0	276 ± 4	0.37 ± 0.0
M2C	0.70 ± 0.0	0.51 ± 0.0	**252 ± 0**	0.00 ± 0.0	0.45 ± 0.0	**512 ± 0**	1.00 ± 0.0
M3C	0.63 ± 0.0	0.46 ± 0.0	**252 ± 0**	0.00 ± 0.0	0.42 ± 0.0	**512 ± 0**	1.00 ± 0.0
M2O	0.70 ± 0.0	0.51 ± 0.0	159 ± 3	0.10 ± 0.3	0.45 ± 0.0	254 ± 10	0.35 ± 0.0
M3O	0.63 ± 0.0	0.47 ± 0.0	172 ± 4	0.10 ± 0.3	0.42 ± 0.0	289 ± 12	0.48 ± 0.0
Dispersion Game $n = 12$ **(4096 joint actions, 924 optimal)**							
Joint	*19.48 ± 0.7*	31.72 ± 0.9	*210 ± 8*	2.30 ± 1.0	0.80 ± 0.0	*1,108 ± 34*	0.00 ± 0.0
F1	1.17 ± 0.0	1.82 ± 0.0	*186 ± 39*	2.40 ± 1.3	0.80 ± 0.0	*1,137 ± 37*	0.01 ± 0.0
F2R	1.08 ± 0.0	1.75 ± 0.0	303 ± 14	0.00 ± 0.0	0.70 ± 0.0	1,635 ± 17	0.18 ± 0.0
F3R	0.99 ± 0.0	1.67 ± 0.0	351 ± 7	0.90 ± 0.7	0.63 ± 0.0	*1,364 ± 20*	0.28 ± 0.0
F2C	**0.17 ± 0.0**	0.37 ± 0.0	*924 ± 0*	0.00 ± 0.0	0.27 ± 0.0	**4,096 ± 0**	1.00 ± 0.0
F3C	**0.20 ± 0.0**	0.43 ± 0.1	*774 ± 194*	0.00 ± 0.0	0.27 ± 0.0	**3,711 ± 510**	0.92 ± 0.1
F2O	0.99 ± 0.0	1.64 ± 0.0	297 ± 8	0.60 ± 0.5	0.64 ± 0.0	1,403 ± 56	0.28 ± 0.0
F3O	0.74 ± 0.0	1.31 ± 0.0	344 ± 13	0.70 ± 0.6	0.49 ± 0.0	1,673 ± 40	0.43 ± 0.0
M1	1.17 ± 0.0	1.83 ± 0.0	*187 ± 30*	2.20 ± 1.5	0.80 ± 0.0	*1,134 ± 36*	0.00 ± 0.0
M2R	1.14 ± 0.0	1.80 ± 0.0	303 ± 4	0.00 ± 0.0	0.78 ± 0.0	1,633 ± 6	0.18 ± 0.0
M3R	1.09 ± 0.0	1.75 ± 0.0	350 ± 9	1.00 ± 0.6	0.75 ± 0.0	*1,363 ± 24*	0.28 ± 0.0
M2C	1.14 ± 0.0	1.80 ± 0.0	**813 ± 69**	0.00 ± 0.0	0.78 ± 0.0	**3,831 ± 246**	0.95 ± 0.0
M3C	1.08 ± 0.0	1.74 ± 0.0	**920 ± 5**	0.00 ± 0.0	0.75 ± 0.0	**4,089 ± 10**	1.00 ± 0.0
M2O	1.14 ± 0.0	1.80 ± 0.0	291 ± 8	0.40 ± 0.5	0.78 ± 0.0	1,348 ± 37	0.27 ± 0.0
M3O	1.08 ± 0.0	1.74 ± 0.0	329 ± 12	0.60 ± 0.5	0.75 ± 0.0	1,560 ± 57	0.39 ± 0.0
Generalized Firefighting $n = 9$ **(524.288 joint actions, 17.682 optimal)**							
Joint	*69.98 ± 0.9*	142.27 ± 1.3	*2,556 ± 50*	3,808.40 ± 602	8,339.16 ± 1.3	*94,559 ± 522*	0.02 ± 0.0
F1	0.25 ± 0.0	0.31 ± 0.0	14,887 ± 90	319.00 ± 103.	64.62 ± 1.0	333,466 ± 1,06	0.86 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
F2R	0.68 ± 0.2	0.86 ± 0.3	11,986 ± 54	621.40 ± 64.6	325.70 ± 51.0	257,663 ± 12,168	0.78 ± 0.0
F3R	0.17 ± 0.0	0.20 ± 0.1	15,934 ± 47	157.80 ± 74.6	38.63 ± 12.7	393,625 ± 30,1	0.91 ± 0.0
F2C	**0.00 ± 0.0**	0.00 ± 0.0	**17,682 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**524,288 ± 0**	1.00 ± 0.0
F3C	**0.00 ± 0.0**	0.00 ± 0.0	**17,682 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**524,288 ± 0**	1.00 ± 0.0
F2O	0.18 ± 0.0	0.21 ± 0.0	15,761 ± 2816	85.70 ± 40.5	40.58 ± 10.1	385,288 ± 20,999	0.90 ± 0.0
F3O	**0.09 ± 0.00**	0.10 ± 0.0	**16,939 ± 41**	51.40 ± 38.7	22.72 ± 8.6	**464,893 ± 26,1**	0.96 ± 0.0
FTF	**0.00 ± 0.0**	0.00 ± 0.0	**17,682 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**524,288 ± 0**	1.00 ± 0.0
M1	6.55 ± 0.0	17.68 ± 0.0	14,848 ± 73	331.80 ± 62.7	2,030.74 ± 0.9	333,711 ± 767	0.86 ± 0.0
M2R	4.54 ± 0.2	13.09 ± 0.2	12,444 ± 535	196.60 ± 121.1	2,718.26 ± 1128	65,765 ± 10,895	0.80 ± 0.0
M3R	2.77 ± 0.2	9.05 ± 0.2	15,596 ± 42	209.40 ± 59.9	1,425.89 ± 9.9	371,664 ± 25,3	0.89 ± 0.0
M2C	4.30 ± 0.0	12.83 ± 0.0	**17,680 ± 0**	0.00 ± 0.0	1,721.18 ± 0.9	**465,019 ± 2**	0.95 ± 0.0
M3C	2.71 ± 0.0	8.99 ± 0.0	**17,680 ± 0**	0.00 ± 0.0	1,422.88 ± 0.7	**465,090 ± 2**	0.95 ± 0.0
M2O	4.37 ± 0.1	12.87 ± 0.2	15,439 ± 432	331.40 ± 72.6	1,723.51 ± 1621	67,534 ± 15,640	278 ± 0.0
M3O	2.96 ± 0.1	9.24 ± 0.1	**16,783 ± 30**	54.00 ± 23.9	1,431.61 ± 23	**298,182 ± 26,3**	0.82 ± 0.0
MTF	5.30 ± 0.0	14.34 ± 0.0	**17,682 ± 0**	0.00 ± 0.0	1,765.64 ± 1.0	**512,022 ± 810**	0.99 ± 0.0
Aloha $n = 9$ **(512 joint actions, 1 optimal)**							
Joint	4.86 ± 1.6	47.77 ± 9.2	0 ± 0	4.40 ± 1.7	4.47 ± 0.6	215 ± 17	0.71 ± 0.0
F1	6.14 ± 0.0	130.68 ± 0.6	0 ± 0	10.00 ± 0.0	6.62 ± 0.0	127 ± 1	0.64 ± 0.0
F2R	6.97 ± 0.4	123.35 ± 26.5	0 ± 0	9.00 ± 1.8	6.73 ± 0.3	*100 ± 11*	0.58 ± 0.0
F3R	5.05 ± 0.5	87.35 ± 15.5	0 ± 0	8.00 ± 2.2	5.97 ± 0.6	133 ± 7	0.68 ± 0.0
F2C	2.25 ± 0.0	9.03 ± 0.2	**1 ± 0**	0.00 ± 0.0	2.06 ± 0.0	187 ± 1	0.82 ± 0.0
F3C	1.32 ± 0.0	18.60 ± 0.3	0 ± 0	2.00 ± 0.0	1.47 ± 0.0	365 ± 1	0.88 ± 0.0
F2O	4.98 ± 0.2	82.30 ± 7.7	0 ± 0	6.60 ± 1.8	5.71 ± 0.4	135 ± 9	0.69 ± 0.0
F3O	3.95 ± 0.4	49.17 ± 9.6	0 ± 0	7.00 ± 2.2	5.01 ± 0.7	138 ± 8	0.73 ± 0.0
FTF	**0.00 ± 0.0**	0.00 ± 0.0	**1 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**512 ± 0**	1.00 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
M1	*11.93 ± 0.0*	145.97 ± 0.1	0 ± 0	10.00 ± 0.0	9.68 ± 0.0	126 ± 1	0.64 ± 0.0
M2R	*10.45 ± 0.2*	140.15 ± 5.7	0 ± 0	9.00 ± 1.6	8.85 ± 0.1	*105 ± 11*	0.60 ± 0.0
M3R	8.71 ± 0.3	125.24 ± 6.2	0 ± 0	7.80 ± 0.6	7.85 ± 0.2	131 ± 7	0.69 ± 0.0
M2C	*10.31 ± 0.0*	138.52 ± 0.1	0 ± 0	10.00 ± 0.0	8.79 ± 0.0	140 ± 1	0.68 ± 0.0
M3C	8.68 ± 0.0	126.24 ± 0.2	0 ± 0	10.00 ± 0.0	7.86 ± 0.0	152 ± 1	0.72 ± 0.0
M2O	*10.48 ± 0.2*	138.60 ± 5.8	0 ± 0	9.60 ± 0.8	8.86 ± 0.1	*114 ± 10*	0.62 ± 0.0
M3O	8.86 ± 0.4	124.02 ± 12.3	0 ± 0	9.40 ± 1.8	7.87 ± 0.3	125 ± 12	0.63 ± 0.0
MTF	6.25 ± 0.0	85.95 ± 1.1	0 ± 0	2.00 ± 0.0	6.10 ± 0.0	191 ± 5	0.84 ± 0.0
Aloha $n = 12$ **(4096 joint actions, 2 optimal)**							
Joint	*23.98 ± 0.6*	135.36 ± 2.5	0 ± 0	7.40 ± 3.1	12.12 ± 0.3	*700 ± 13*	0.42 ± 0.0
F1	7.30 ± 0.0	231.09 ± 0.3	0 ± 0	12.00 ± 0.0	7.96 ± 0.0	861 ± 4	0.64 ± 0.0
F2R	6.84 ± 0.2	199.45 ± 11.1	0 ± 0	10.40 ± 1.2	7.67 ± 0.2	851 ± 33	0.65 ± 0.0
F3R	6.56 ± 0.3	182.17 ± 15.5	0 ± 0	9.80 ± 2.1	7.59 ± 0.3	860 ± 47	0.66 ± 0.0
F2C	2.18 ± 0.0	10.36 ± 0.1	**2 ± 0**	0.00 ± 0.0	2.06 ± 0.0	1,380 ± 5	0.83 ± 0.0
F3C	**0.81 ± 0.0**	12.45 ± 0.2	**1 ± 0**	2.00 ± 0.0	1.10 ± 0.0	**2,488 ± 10**	0.91 ± 0.0
F2O	6.32 ± 0.3	169.46 ± 13.1	0 ± 0	10.20 ± 1.7	7.29 ± 0.3	864 ± 34	0.67 ± 0.0
F3O	4.71 ± 0.5	92.22 ± 18.6	0 ± 0	7.60 ± 2.3	5.94 ± 0.7	954 ± 67	0.73 ± 0.0
FTF	**0.00 ± 0.0**	0.00 ± 0.0	**2 ± 0**	0.00 ± 0.0	0.00 ± 0.0	**4,096 ± 0**	1.00 ± 0.0
M1	*15.37 ± 0.0*	230.65 ± 0.4	0 ± 0	12.00 ± 0.0	12.45 ± 0.0	845 ± 26	0.63 ± 0.0
M2R	13.85 ± 0.1	225.18 ± 2.3	0 ± 0	10.80 ± 1.3	11.61 ± 0.1	838 ± 22	0.65 ± 0.0
M3R	12.38 ± 0.2	217.04 ± 5.0	0 ± 0	10.00 ± 1.8	10.76 ± 0.1	853 ± 34	0.66 ± 0.0
M2C	13.83 ± 0.0	225.33 ± 0.2	0 ± 0	12.00 ± 0.0	11.60 ± 0.0	908 ± 7	0.67 ± 0.0
M3C	12.27 ± 0.0	214.92 ± 0.2	0 ± 0	12.00 ± 0.0	10.69 ± 0.0	969 ± 6	0.71 ± 0.0
M2O	13.94 ± 0.2	225.44 ± 1.5	0 ± 0	11.60 ± 0.8	11.61 ± 0.1	*671 ± 58*	0.56 ± 0.0
M3O	12.56 ± 0.3	214.01 ± 3.2	0 ± 0	11.40 ± 0.9	10.85 ± 0.2	*752 ± 87*	0.61 ± 0.0

Table 14 (continued)

Model	Mean square error	MSE on optimal actions	Optimal actions found	Value loss	Boltzmann value loss	Correctly ranked	Kendall τ
MTF	9.64 ± 0.0	167.68 ± 0.6	0 ± 0	4.00 ± 0.0	8.82 ± 0.0	1,254 ± 14	0.83 ± 0.0

– Finally, we compute the number of correctly ranked actions (accounting for ties where needed) and the corresponding Kendall τ-b coefficient [19] between the computed ranking and the original one (6th and 7th columns respectively).

A low value on the first two columns indicates that a representation is close to the original action value function: especially when the value of second column is low in combination with the value of the third one, the learned representation tends to correctly reconstruct and identify the optimal joint actions. The fourth and fifth columns tell us how reliably these representations can be used to perform decision making (i.e. when applying a greedy policy with respect to the joint action-value function during the evaluation phase) under two different kinds of policy. In fact, if the regret is low, even if the representation is not accurate, the model can still be used to pick actions for the team resulting in high rewards. Finally, the last two columns point out if the reconstructed values of the joint actions match the same hierarchy that they have in the original action-value function: even if the representation is not accurate in terms of mean squared error, a correct ranking of the actions points out that it has the same structure of the original one, and thus the two are similar except for the magnitude of the values itself. All of these measures are interlinked and express how similar a learned representation is to the original, but they all highlight different aspects that can help us analyse the benefits and drawbacks of these models.

For every method, mean values and standard errors across 10 runs are reported. For every game, we highlight the best (bold) and worst (italic) performances of the investigated methods on some of the proposed measures.

Acknowledgements This research made use of a GPU donated by NVIDIA. F.A.O. is funded by EPSRC First Grant EP/R001227/1. This project had received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 — INFLUENCE).

References

1. Amato, C., & Oliehoek, F. A. (2015). Scalable planning and learning for multiagent POMDPs. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, AAAI'15 (pp. 1995–2002). American Association for Artificial Intelligence.
2. Boehmer, W., Kurin, V., & Whiteson, S. (2020). Deep coordination graphs. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20 (pp. 980–991). PMLR.
3. Busoniu, L., Babuska, R., Schutter, D., & Bart. . (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics Part C (Applications and Reviews), 38,* 156–172.
4. Castellini, J., Oliehoek, F. A., Savani, R., & Whiteson, S. (2019). The representational capacity of action-value networks for multi-agent reinforcement learning - extended abstract. In *Proceedings of*

the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS'19 (pp. 1862–1864). International Foundation for Autonomous Agents and Multiagent Systems.
5. Chung, J. J., Mikliundefined, D., Sabattini, L., Tumer, K., & Siegwart, R. (2019). The impact of agent definitions and interactions on multiagent learning for coordination. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS'19 (pp. 1752–1760). International Foundation for Autonomous Agents and MultiAgent Systems.
6. Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th/10th AAAI Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI'98/IAAI'98 (pp. 746–752). American Association for Artificial Intelligence.
7. Foerster, J., Assael, I. A., de Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29*, NIPS'16 (pp. 2137–2145). Curran Associates, Inc.
8. Foerster, J. N., Nardelli, N., Farquhar, G., Torr, Philip H. S., Kohli, P., & Whiteson, S. (2017). Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17 (pp. 1146–1155). PMLR.
9. Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, AAAI'18 (pp. 2974–2982). American Association for Artificial Intelligence.
10. Ghavamzadeh, M., Mahadevan, S., & Makar, R. (2006). Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems, 13*(2), 197–229.
11. Grenager, T., Powers, R. A., & Shoham, Y. (2002). Dispersion games: General definitions and some specific learning results. In *Proceedings of the 18th AAAI Conference on Artificial Intelligence*, AAAI'02 (pp. 398–403). American Association for Artificial Intelligence.
12. Guestrin, C., Koller, D., & Parr, R. (2002). Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, NIPS'01 (pp. 1523–1530). Morgan Kaufmann Publishers Inc.
13. Guestrin, C., Lagoudakis, M. G., & Parr, R. (2002). Coordinated reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, ICML'02 (pp. 227–234). Morgan Kaufmann Publishers Inc.
14. Guestrin, C., Venkataraman, S., & Koller, D. (2002). Context-specific multiagent coordination and planning with factored MDPs. In *Proceedings of the 19th/10th AAAI Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI'02/IAAI'02. American Association for Artificial Intelligence.
15. Guestrin, C., Koller, D., Parr, R., & Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research, 19*(1), 399–468.
16. Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multi-Agent Systems* (pp. 66–83). Springer.
17. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.
18. Hofstadter, D. R. (1985). *Metamagical Themas: Questing for the essence of mind and pattern*. Basic Books, Inc.
19. Kendall, M., & Gibbons, J. D. (1990). *Rank Correlation Methods*, 5th edn. A Charles Griffin Title.
20. Kok, J. R., & Vlassis, N. (2006). Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research, 7,* 1789–1828.
21. Kok, J. R., 't Hoen, P. J., Bakker, B., & Vlassis, N. (2005). Utile coordination: Learning interdependencies among cooperative agents. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG)* (pp. 29–36).
22. Kraemer, L., & Banerjee, B. (2016). Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing, 190,* 82–94.
23. Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., & Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS'17 (pp. 464–473). International Foundation for Autonomous Agents and MultiAgent Systems.
24. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*, ICLR'16.

25. Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on International Conference on Machine Learning*, ICML'94 (pp. 157–163). Morgan Kaufmann Publishers Inc.
26. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30*, NIPS'17 (pp. 6379–6390). Curran Associates, Inc.
27. Mahajan, A., Rashid, T., Samvelyan, M., & Whiteson, S. (2019). MAVEN: multi-agent variational exploration. In *Advances in Neural Information Processing Systems 32*, NIPS'19 (pp. 7611–7622). Curran Associates, Inc.
28. Matignon, L., Laurent, G. J., Fort-Piat, L., & Nadine. (2012). Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems. *Knowledge Engineering Review, 27*(1), 1–31.
29. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.
30. Mnih, V., Badia, A. P., Mirza, M., Graves, A., Harley, T., Lillicrap, T. P., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *ICML'16* (pp. 1928–1937). PMLR.
31. Oliehoek, F. A. (2010). *Value-based planning for teams of agents in stochastic partially observable environments*. PhD thesis, Informatics Institute, University of Amsterdam.
32. Oliehoek, F. A., Whiteson, S., & Spaan, M. T. J. (2011). Exploiting agent and type independence in collaborative graphical Bayesian games. *CoRR*, abs/1108.0404.
33. Osborne, M. J., & Rubinstein, A. (1994). *A Course in Game Theory*. The MIT Press.
34. Palmer, G., Tuyls, K., Bloembergen, D., & Savani, R. (2018). Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS'18 (pp. 443–451). International Foundation for Autonomous Agents and MultiAgent Systems.
35. Rashid, T., Samvelyan, M., Schröder de Witt, C., Farquhar, G., Foerster, J. N., & Whiteson, S. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, ICML'18 (pp. 4292–4301). JMLR.org.
36. Rogers, A., Farinelli, A., Stranders, R., & Jennings, N. R. (2011). Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence, 175*(2), 730–759.
37. Shahrampour, S., Rakhlin, A., & Jadbabaie, A. (2017). Multi-armed bandits in multi-agent networks. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2786–2790).
38. Son, K., Kim, D., Kang, W. J., Hostallero, D., & Yi, Y. (2019). QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, ICML'19 (pp. 5887–5896). JMLR.org.
39. Sukhbaatar, S., Szlam, A., & Fergus, R. (2016). Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems 30*, NIPS'16 (pp. 2252–2260). Curran Associates, Inc.
40. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., & Graepel, T. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS'18 (pp. 2085–2087). International Foundation for Autonomous Agents and MultiAgent Systems.
41. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., et al. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE, 12*(4), 1–15.
42. Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*, ICML'93 (pp. 330–337). Morgan Kaufmann Publishers Inc.
43. Van der Pol, E., & Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. In *Workshop on Learning, Inference and Control of Multi-Agent Systems*, NIPS'16.
44. Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3), 279–292.
45. Wei, E., & Luke, S. (2016). Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research, 17*(84), 1–42.

46. Wunder, M., Littman, M. L., & Babes, M. (2010). Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *Proceedings of the 27th International Conference on Machine Learning*, ICML'10 (pp. 1167–1174). Omnipress.
47. Ye, D., Zhang, M., & Yang, Y. (2015). A multi-agent framework for packet routing in wireless sensor networks. *Sensors, 15*(5), 10026–10047.