# Scaling Multi-Agent RL through Mean Field Games and Incremental Learning

**Max Balsells** [1] *, **Aleix Seguí** [1] *, **Artur Żółkowski** [1] *

[1] ETH Zürich
mbalsells@ethz.ch, asegui@ethz.ch, azolkowski@ethz.ch

## Abstract

Various Reinforcement Learning (RL) algorithms rely on learning state-action value functions to learn an optimal policy. This framework can be easily extended to a Multi-Agent RL (MARL) setting by considering joint actions, making it the most common approach to this new scenario. However, such a setting presents challenges due to the exponential growth of the action space, the need for decentralized policies for real-world applications, and dealing with non-stationary environments during the learning process. This work aims to study the performance of different MARL methods when scaling the number of agents. We also propose two approaches to tackle the issue of scalability: a) a new algorithm **MFQMIX**, which combines different techniques for Q-value factorization, and b) using **Incremental Learning**, i.e. slowly increasing the number of agents in the environment. We show that MFQMIX outperforms all baselines when trained in a non-stationary setting against each other, but its performance stalls when increasing the number of agents. Nonetheless, Incremental Learning successfully improves scalability, allowing agents to learn in more than twice as crowded environments.

## Introduction

Multi-Agent Reinforcement Learning (MARL) is a field that has gained some traction recently (Busoniu, Babuska, and De Schutter 2008; Zhang, Yang, and Başar 2021) for the multiple applications it offers. From transmit power control (Nasir and Guo 2019) to UAV swarms (Chen, Chang, and Zhang 2020), MARL has great potential to tackle many emerging problems. Prior work has shown great results in complex environments with a limited number of agents (Bansal et al. 2017; Vinyals et al. 2019; Lowe et al. 2017). However, the performance of these algorithms deteriorates with large populations, hindering their possible applications.

We focus our work on obtaining good policies even with a large population in a cooperative setting. In particular, we work in environments consisting of two teams competing against each other, where the agents within one team act cooperatively. However, we will be modeling only the agents of one of the teams, that is, the other agents are understood as part of the environment, which is why we consider this to be a cooperative setting. Additionally, our work is framed in a setting where we only have a single common team reward (i.e. all agents share and observe the same reward), since it's a very general framework and the most suitable for real use cases. Notice that depending on whether the other team is being modeled by a fixed policy or a dynamic one, we will be working in a stationary or a non-stationary setting respectively. Even though we will be focusing mostly on stationary settings, we will conduct some experiments on a non-stationary setting, to better tell apart which algorithms are more performant than the others, and to make sure that algorithms are able to adapt to changes in the environment, which is a desirable characteristic, since most real-life scenarios may require it.

When having direct feedback for each agent (i.e. each agent observes its own contribution towards the shared team reward), some prior work has shown great results even with many agents (Yang et al. 2018; Subramanian et al. 2020). Among these methods, those leveraging Mean Field Games (MFG) theory (Jovanovic and Rosenthal 1988; Guo et al. 2019), seem to perform better (Yang and Wang 2020). This framework makes a pass to the limit and characterizes all agents through a mean field distribution, reducing the multi-agent problem to a two-player game: that is, an arbitrary agent and an agent with an action distribution given by the mean-field action (Carmona, Delarue et al. 2018). However, such methods have some strong assumptions about the environment, such as considering all agents are similar to each other. And crucially, their performance doesn't scale well when dealing with a single joint team reward among all agents: i.e. when the agents cannot observe their individual contributions, but are just given feedback of the whole team's performance, which is the scenario that concerns us.

In this work, we study the scalability of the most recent work in terms of performance when having joint feedback for the team. We propose a new algorithm MFQMIX, that combines MFG techniques with the best-performing algorithms on MARL algorithms for a small number of agents, to tackle the issue of scalability. We also propose using Incremental Learning by slowly increasing the number of agents interacting in the environment during the learning process, which we show helps mitigate the degradation in performance that scaling the number of agents produces.

*These authors contributed equally.

## Preliminaries

This work deals with mixed cooperative-competitive games, where two teams compete against each other. However, we only model one of the teams at a time, understanding the other one as part of the environment. Hence, formally, we work in a strictly cooperative setting. From the perspective of the modeled team, a game can be represented by a tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}, P, R, \mu, \gamma, n \rangle$, where $\mathcal{S}$ represents the state space, $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^n$ represents the action space, in which $\mathcal{A}^i$ denotes the set of actions of the $i$-th agent (which in our case is symmetric for all agents: $\mathcal{A}^i = \mathcal{A}^1$), $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ represents the transition probabilities, $R : \mathcal{S} \times \mathcal{A} \to R$ the rewards, $\mu \in \Delta(\mathcal{S})$ the initial state distribution, $n$ the number of agents of the team and $\gamma \in [0, 1)$ the discount factor. Let $r_t = R(s_t, \mathbf{a}_t)$ denote the reward of the team at timestep $t$. Our objective is to find a policy $\pi = \pi^1 \times \cdots \times \pi^n : \mathcal{S} \to \Delta(\mathcal{A})$, where $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$, that maximizes the expected discounted joint reward, that is,

$$\arg\max_\pi E\left[r | s_0 \sim \mu, \mathbf{a}_t \sim \pi(\cdot|s_t), s_{t+1} \sim P(s_t, \mathbf{a}_t)\right]$$

where $r = \sum_{t=0}^{\infty} \gamma^t r_t$. Notice that we require the policy to allow a factorization in terms of independent agent policies $\pi^i$, because we want to obtain decentralized policies, i.e. each agent should decide its action on its own, given the current state. A joint policy has associated a joint action-value function $Q^\pi(s, \mathbf{a}) = E\left[r | s_0 = s, \mathbf{a}_0 = \mathbf{a}, P, \pi\right]$. Notice that such a Q-value function allows us to find the best possible joint action $\mathbf{a}$ by considering $\arg\max_{\mathbf{a} \in \mathcal{A}} Q(s, \mathbf{a})$. However, evaluating this expression takes exponential time with respect to the number of agents, and does not allow us to obtain decentralized policies. Most algorithms (Sunehag et al. 2017; Rashid et al. 2020; Peng et al. 2021) tackle this issue by assuming the existence of a factorization of the joint Q function into individual Q-value functions $Q^{\pi^i}(s, a^i)$ such that the joint Q-function can be obtained by a (potentially non-linear) combination of the individual ones. We will compare different approaches used to factorize the joint state-action values and propose a new method that leverages techniques from Mean Field Games in order to find a factorization of the joint Q-function that approximates it better.

## Related Work

Most prior work aims to decompose the joint state-action value function into independent ones, i.e. $Q(s, \mathbf{a}) = f(Q^1(s, a^1), \ldots, Q^n(s, a^n))$. One of the first such methods was VDN (Sunehag et al. 2017), which factorizes $Q(s, \mathbf{a})$ as a sum of $Q^i(s, a^i)$, where each $Q^i(s, a^i)$ is represented by a neural network (NN). More recent work (Rashid et al. 2020) shows that we can factorize $Q$ using a non-linear function $f$, and that, as long as $f$ is monotonically increasing, the joint action maximizing the Q-function is the one formed by maximizing each $Q^i$ function independently, meaning we can extract optimal decentralized policies from it. In (Rashid et al. 2020) the authors model $f$ also using a NN: $f_\theta$, but guaranteeing its monotonicity by generating its weights through a hyper-network. Finally, some of the most recent work (Peng et al. 2021) shows that by modeling $f$ as a non-monotonic function, i.e. without said restriction, and by just taking the maximum action for each agent, one can get similar and even in some cases better results.

Some other works are framed in a different setting: each agent has an individual reward function, meaning each agent $i$ gets an individual feedback $r_t^i$. In this setting, the individual rewards give rise to independent Q-value functions $Q^i(s, \mathbf{a}) = E\left[r^i\right]$ where $r^i$ denotes the discounted total reward for agent $i$. However, methods assuming direct feedback on each agent also require breaking down the joint action on each individual Q-function in order to get a decentralized and good-performing policy. Some of the most successful recent work (Yang et al. 2018; Subramanian et al. 2020), leverages the theory on MFG to propose a completely different way to factorize it. In (Yang et al. 2018) the authors show that we can approximate $Q^i(s, \mathbf{a}) \approx Q^i(s, a^i, \overline{a}^i)$, where $\overline{a}^i$ denotes the mean action taken by the local neighborhood $N_i$ of agent $i$. Assuming that agents outside this neighborhood don't contribute to $r_t^i$, such an approximation theoretically guarantees the convergence of the method, and that a Nash Equilibrium will be reached.

## Experiment details

The experiments are run in general-sum games from Magent2 (Zheng et al. 2018). The environments feature two agent populations. In *Adversarial Pursuit*, a fixed number of *predators* earn rewards for tagging *prey*, while prey can move again after being tagged. Predators are penalized for attempting to tag but rewarded upon success. Preys are penalized only when tagged. In *Battle*, two symmetrical teams compete, earning rewards for tagging opponents. The algorithms are implemented using the MARL framework `XuanCe` (Liu et al. 2023).

We are running two sets of experiments. First, we compare each method against a random policy in Adversarial Pursuit. This allows for an easy comparison of the methods in a stationary setting, showing which ones scale and perform better overall. Secondly, we run a simulation between every pair of methods in Battle, where they learn at the same time. This reveals the algorithms' true performance in a non-stationary setting and which method performs better in a fair head-to-head.

Adversarial Pursuit is shown in Figure 1. The agents under an MFQ policy after convergence are shown. We can observe how the different predators learn to corner prey and

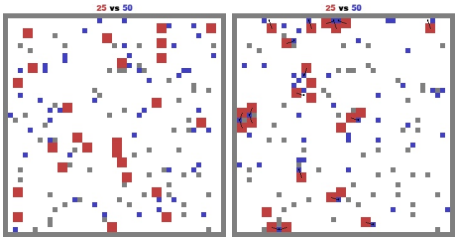


Figure 1: Adversarial pursuit environment. The beginning of the game is shown at the left, and an advanced step after convergence is shown at the right. Red are the predators, and blue are the prey (random policy).

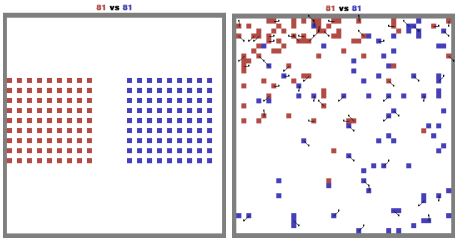

Figure 2: Battle environment example: The left shows the game at the start; the right depicts a later stage after convergence. Red represents MFQ, and blue a random policy.

tag them all the time, maximizing their reward. The environment Battle is shown in Figure 2. After convergence, the populations whose policy has been trained (red) tend to stick together in separate groups, with a small tagging fight along the separating plane.

## Methods

### Motivation

Current MARL algorithms struggle with large populations when dealing with a common reward for the whole team.

Figure 3 depicts the performance of some of the baselines, namely QMIX, MFQ, and IQL, in a joint reward setting. The plot shows that QMIX and MFQ scale better than IQL with the number of agents, and, in particular, MFQ converges much faster than QMIX when dealing with large populations. However, no method manages to obtain a good policy when the number of agents is 25 or higher (they just collapse to 0 reward, which is the equivalent of not tagging at all). That is, no algorithm scales well in a joint reward setting.

Additionally, Figure 4 compares the performance of MFQ with varying number of agents in two settings: shared team rewards and individual rewards based on each agent's contribution to the total team reward (sum of rewards). The results show a major performance drop in the joint rewards setting as agent numbers increase, highlighting the challenge of disentangling a shared reward into individual contributions. This motivates the search of better methods to disentangle team rewards as the population scales. Hence, in this work we:

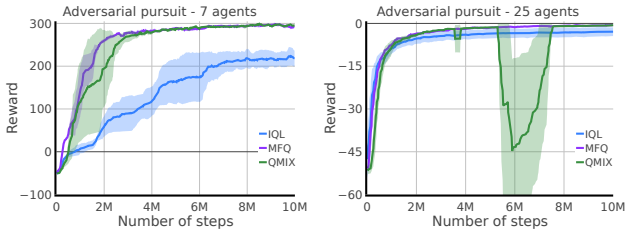


Figure 3: Baseline performance in Adversarial Pursuit across different population sizes. Predators learn policies using various algorithms, while prey follows a fixed random policy. The map size scales with the number of agents, and rewards are normalized.

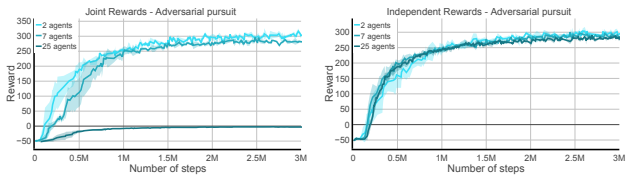


Figure 4: Training MFQ in Adversarial Pursuit: Left plot shows training with a shared team performance reward, while the right plot uses individual agent contributions as rewards. Rewards are normalized by the number of agents.

- Develop MFQMIX, a new algorithm designed to improve scalability and performance in MARL by making a less harsh approximation of the joint Q function into individual Q functions.
- Use Incremental Learning by starting with smaller populations, where fewer agents make it easier to disentangle the team reward, and gradually expanding to larger populations.

### Mean Field QMIX

We propose a new algorithm MFQMIX, which leverages the use of MFG theory displayed by some previous work focused on individual agent rewards, to boost some of the best methods dealing with team rewards. In particular, we propose to make a less optimistic approximation than QMIX, by approximating $Q(s, \mathbf{a}) \approx f(Q^1(s, \mathbf{a}), \cdots, Q^n(s, \mathbf{a}))$, on the premise that, in most scenarios, we can decompose the joint reward into a reward based on each agent's performance, which might be influenced, to a certain extent, by other agents. Notice we keep the monotonicity constraint of QMIX. We can leverage the typical approximation in MFQ by considering the other agents as a distribution over actions. In our case, we further use this technique to approximate the joint Q-value function $Q(s, \mathbf{a}) \approx f(Q^1(s, \mathbf{a}), \cdots, Q^n(s, \mathbf{a})) \approx f(Q^1(s, a^1, \overline{a}^1), \cdots, Q^n(s, a^n, \overline{a}^n))$, where $\overline{a}^i$ denotes the action distribution over the agents in $N_i$: the neighborhood of agent $i$ (which consists of the 10% closest agents to agent $i$). In practice, we don't train separate neural networks for

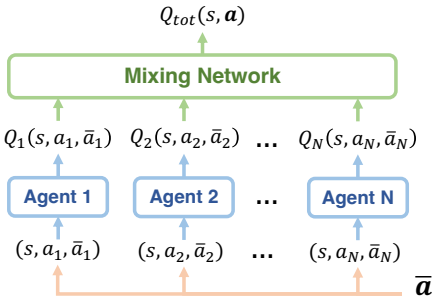


Figure 5: MFQMIX breaks down the joint Q-value by considering a combination of individual Q-functions together with the mean actions of their neighborhood.

each agent. Instead, we use a shared network with unique embeddings for each agent. This allows agents to develop distinct policies while benefiting from the shared learning history of other agents.

We will be studying a set of baselines from both settings: QMIX (Rashid et al. 2020), MFQ (Yang et al. 2018) and, IQL (Tan 1993), with which we aim to compare our method.

### Incremental Learning

We observed that performance declines in large populations likely because it's difficult to disentangle each agent's contribution to the team's reward. To address this, we propose Incremental Learning in multi-agent environments, gradually increasing the number of agents. Initially, inactive agents are excluded from training, simplifying reward attribution for smaller groups. This helps agents better understand the environment. As more agents are added, the earlier ones adapt more easily due to their established policies. Since all agents share some policy parameters (with specific embeddings), newly added agents also benefit from a better starting point than random initialization.

In our experiments, the number of agents and prey remain constant, but only a subset of agents $\mathcal{T}$ use our policy (the other ones act randomly), and it is this subset that is being used to train the policy. As training goes on, we increase the number of learning agents by including more agents in $\mathcal{T}$. We do so linearly, adding a batch of learning agents every few epochs, giving enough time for the network to converge with each new batch.

## Results

We ran all experiments using 5 different seeds, reporting the mean and standard deviation in each plot. Figure 7 shows the performance of MFQMIX compared to the baselines in the two described environments. In the upper plot, we observe that MFQMIX is still not able to achieve high rewards with large populations. In the non-stationary setting in the bottom, we compare QMIX, MFQ, and MFQMIX in a head-to-head battle. The reward difference is shown and reveals that MFQMIX is the most performant algorithm, achieving higher reward differences. The curves for QMIX and MFQMIX show unstable training due to the complex architecture on which they rely, as seen in (Hu et al. 2021).

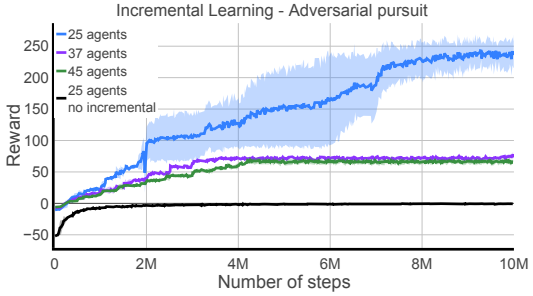


Figure 6: Incremental Learning during training allows agents to learn effectively in up to twice larger populations.

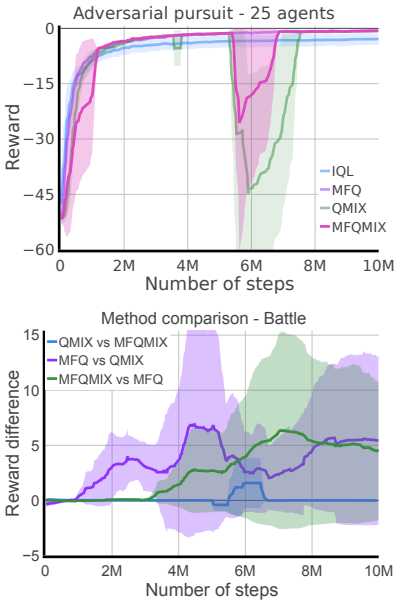


Figure 7: MFQMIX is tested in a stationary setting in *adversarial pursuit* (up). Moreover, MFQ, QMIX, and MFQMIX are run against each other (i.e. non stationary setting) in *battle* with 12 agents per team (down). Both teams have the same reward function and we study their differences.

Incremental Learning, on the other hand, allows us to tackle the scalability issue. Figure 6 shows how training using incremental learning breaks the 0-reward barrier, allowing it to scale up to double the number of agents. It is observed that, as the number of agents is larger, the reward per agent is not as great. The achieved policy is not as good as with a smaller number of agents, leaving room for improvement for a very large number of agents.

## Conclusion

In this work, we showed the problem of scalability in MARL, motivating the need to find better alternatives. We proposed new methods to tackle this issue.

- We characterized the problem of scalability in MARL and proposed two approaches to tackle it. A new algorithm MFQMIX, and a training method (incremental learning) and its benefits were experimentally shown.
- MFQMIX still struggles with scalability, though it is more performant than QMIX and MFQ in non-stationary settings. This result highlights the importance of the mean-field action as a contribution to the algorithms.
- Incremental Learning is experimentally found to greatly improve scalability, allowing the agents to learn in much large populations. This result is explained by its ability to better disentangle the reward between different agents.

Future work includes analyzing different Incremental Learning strategies, as well as a theoretical analysis of the reward decomposition.

## References


Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2017. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2): 156–172.

Carmona, R.; Delarue, F.; et al. 2018. *Probabilistic theory of mean field games with applications I-II*. Springer.

Chen, Y.-J.; Chang, D.-K.; and Zhang, C. 2020. Autonomous tracking using a swarm of UAVs: A constrained multi-agent reinforcement learning approach. *IEEE Transactions on Vehicular Technology*, 69(11): 13702–13717.

Guo, X.; Hu, A.; Xu, R.; and Zhang, J. 2019. Learning mean-field games. *Advances in neural information processing systems*, 32.

Hu, J.; Jiang, S.; Harding, S. A.; Wu, H.; and Liao, S.-w. 2021. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*.

Jovanovic, B.; and Rosenthal, R. W. 1988. Anonymous sequential games. *Journal of Mathematical Economics*, 17(1): 77–87.

Liu, W.; Cai, W.; Jiang, K.; Cheng, G.; Wang, Y.; Wang, J.; Cao, J.; Xu, L.; Mu, C.; and Sun, C. 2023. XuanCe: A Comprehensive and Unified Deep Reinforcement Learning Library. *arXiv preprint arXiv:2312.16248*.

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

Nasir, Y. S.; and Guo, D. 2019. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on Selected Areas in Communications*, 37(10): 2239–2250.

Peng, B.; Rashid, T.; Schroeder de Witt, C.; Kamienny, P.-A.; Torr, P.; Böhmer, W.; and Whiteson, S. 2021. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34: 12208–12221.

Rashid, T.; Samvelyan, M.; De Witt, C. S.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2020. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178): 1–51.

Subramanian, S. G.; Poupart, P.; Taylor, M. E.; and Hegde, N. 2020. Multi type mean field reinforcement learning. *arXiv preprint arXiv:2002.02513*.

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

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

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean field multi-agent reinforcement learning. In *International conference on machine learning*, 5571–5580. PMLR.

Yang, Y.; and Wang, J. 2020. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*.

Zhang, K.; Yang, Z.; and Başar, T. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control*, 321–384.

Zheng, L.; Yang, J.; Cai, H.; Zhou, M.; Zhang, W.; Wang, J.; and Yu, Y. 2018. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.