Supplementary material to 'Networked Communication for Mean-Field Games with Function Approximation and Empirical Mean-Field Estimation'

A EXPERIMENTS

Experiments were conducted on a Linux-based machine with 2 x Intel Xeon Gold 6248 CPUs (40 physical cores, 80 threads total, 55 MiB L3 cache). We use the JAX framework to accelerate and vectorise our code. Random seeds are set in our code in a fixed way dependent on the trial number to allow easy replication of experiments.

A.1 Games

We conduct numerical tests with five games. All are defined by the agents' reward/transition functions, and chosen for being particularly amenable to intuitive and visualisable understanding of whether the agents are learning behaviours that are appropriate and explainable for the respective objective functions. In all cases, rewards are normalised in [0,1] after they are computed.

Cluster. This is the inverse of the 'exploration' game in [21], where in our case agents are encouraged to gather together by the reward function $R(s_t^i, a_t^i, \hat{\mu}_t) = \log(\hat{\mu}_t(s_t^i))$. That is, agent i receives a reward that is logarithmically proportional to the fraction of the population that is co-located with it at time t. We give the population no indication where they should cluster, agreeing this themselves over time.

Agree on a single target. Unlike in the above 'cluster' game, the agents are given options of locations at which to gather, and they must reach consensus among themselves. If the agents are co-located with one of a number of specified targets $\phi \in \Phi$ (in our experiments we place one target in each of the four corners of the grid), and other agents are also at that target, they get a reward proportional to the fraction of the population found there; otherwise they receive a penalty of -1. In other words, the agents must coordinate on which of a number of mutually beneficial points will be their single gathering place. Define the magnitude of the distances between x, y at t as $dist_t(x, y)$. The reward function is given by $R(s_t^i, a_t^i, \hat{\mu}_t) = r_{targ}(r_{collab}(\hat{\mu}_t(s_t^i)))$, where

$$r_{targ}(x) = \begin{cases} x & \text{if } \exists \phi \in \Phi \text{ s.t. } \mathrm{dist}_t(s_t^i, \phi) = 0 \\ -1 & \text{otherwise,} \end{cases}$$

$$r_{collab}(x) = \begin{cases} x & \text{if } \hat{\mu}_t(s_t^i) > 1/N \\ -1 & \text{otherwise.} \end{cases}$$

Evade shark in shoal. Define the magnitude of the horizontal and vertical distances between x, y at t as $dist_t^h(x, y)$ and $dist_t^v(x, y)$ respectively. The state s_t^i now consists of agent i's position x_t^i and a 'shark's' position ϕ_t. At each time step, the shark steps towards the most populated grid point according to the empirical mean-field distribution i.e. $x_t^* = \arg\max_{x \in \mathcal{S}} \hat{\mu}_t(x)$. A horizontal step is taken if $dist_t^h(\phi_t, x_t^*) \geq dist_t^v(\phi_t, x_t^*)$, otherwise a vertical step is taken. As well as featuring a non-stationary distribution, we add 'common noise' to the environment, with the shark in a random direction with probability 0.01. Such noise that affects the local states of all agents in the same way, making the evolution of the distribution stochastic, makes population-independent policies sub-optimal [20]. Agents are rewarded more for being further from the shark, and also for clustering with other agents. The reward function is given by

$$R(s_t^i, a_t^i, \hat{\mu}_t) = dist_t^h(\phi_t, x_t^i) + dist_t^v(\phi_t, x_t^i) + \mathrm{norm}_{dist}(\log(\hat{\mu}_t(x_t^i))),$$

where $\mathrm{norm}_{dist}(\cdot)$ indicates that the final term is normalised to have the same maximum and minimum values as the total combined vertical and horizontal distance.

Push object to edge. This is similar to the task presented in [9]. As before, define the magnitude of the horizontal and vertical distances between x, y at t as $dist_t^h(x, y)$ and $dist_t^v(x, y)$ respectively. The state s_t^i consists of agent i's position x_t^i and the object's position ϕ_t. The number of agents in the positions surrounding the object at time t generates a probability field around the object, such that the object is most likely to move in the direction away from the side with the most agents. As such, if agents are equally distributed around the object, it will be equally likely to move in any direction, but if they coordinate on choosing the same side, they can 'push' it in a certain direction. If Edges = $\{\text{edge}^1, \ldots, \text{edge}^4\}$ are the grid edges, the closest edge to the object at time t is given by $\text{edge}_t^* = \arg\min_{\text{edge} \in \text{Edges}} \left(\min(dist_t^h(\phi_t, \text{edge}), dist_t^h(\phi_t, \text{edge}) \right)$. Agents are rewarded for how close they are to the object, and for how close the object is to the edge of the grid, i.e. they must coordinate on which side of the object from which to 'push' it, to ensure it moves to the grid's edge. The reward function is given by

$$R(s_t^i, a_t^i, \hat{\mu}_t) = dist_t^h(\phi_t, x_t^i) + dist_t^v(\phi_t, x_t^i) + dist_t^h(\phi_t, \text{edge}_t^*) + dist_t^v(\phi_t, \text{edge}_t^*).$$

Disperse. This is similar to the 'exploration' tasks in [21], [34] and other MFG works. In our version agents are rewarded for being located in more sparsely populated areas but only if they are stationary. The reward function is given by $R(s_t^i, a_t^i, \hat{\mu}_t) = r_{stationary}(-\hat{\mu}_t(s_t^i))$, where

$$r_{stationary}(x) = \begin{cases} x & \text{if } a_t^i \text{ is 'remain stationary'} \\ -1 & \text{otherwise.} \end{cases}$$

A.2 Experimental Metrics

To give as informative results as possible about both performance and proximity to the MFNE, we provide two metrics for each experiment. Both metrics are plotted with mean and standard deviation, computed over the ten trials (each with a random seed) of the system evolution in each setting.

A.2.1 Exploitability. Works on MFGs most commonly use the *exploitability* metric to evaluate how close a given policy π is to a NE policy π^* [1, 3, 20, 21, 24, 27, 34]. The metric usually assumes that all agents are following the same policy π, and quantifies how much an agent can benefit by deviating from π by measuring the difference between the return given by π and that of a *BR* policy with respect to the distribution generated by π:

Definition 9 (Exploitability of π). *The exploitability Ex of policy π is given by:*

$$Ex(\pi) = V(BR(I(\pi)), I(\pi)) - V(\pi, I(\pi)).$$

If π has a large exploitability then an agent can significantly improve its return by deviating from π, meaning that π is far from π^*, whereas an exploitability of 0 implies that $\pi = \pi^*$. Prior works conducting empirical testing have generally focused on the centralised setting, so this classical definition, as well as most evaluations, only consider exploitability when all agents are following a single policy π_k. However, [3] notes that purely independent agents, as well as networked agents, may have divergent policies $\pi_k^i \neq \pi_k^j$ $\forall i, k \in 1, \ldots, N$, as in our own setting. We therefore are interested in the 'exploitability' of the population's joint policy $\boldsymbol{\pi} := (\pi^1, \ldots, \pi^N) \in \Pi^N$.

Since we do not have access to the exact *BR* policy as in some related works [21, 34], we must instead approximate the exploitability, similarly to [3, 26]. We freeze the policy of all agents apart from a deviating agent, for which we store its current policy and then conduct 50 k loops of policy improvement. To approximate the expectations in Def. 9, we take the best return of the deviating agent across 10 additional k loops, as well as the mean of all the other agents' returns across these same 10 loops. (While the policies of all non-deviating agents is π_k in the centralised case, if the non-deviating agents do not share a single policy, then this method is in fact approximating the exploitability of their joint policy $\boldsymbol{\pi}_k^{-d}$, where d is the deviating agent.) We then revert the agent back to its stored policy, before learning continues for all agents as per the main algorithm. Due to the expensive computations required for this metric, we evaluate it every second k iteration of the main algorithm for Figs. 1, 2, 5, 6 and 7, and every fourth iteration for the population-dependent experiments.

The exploitability metric has a number of limitations in our setting. Our approximation takes place via MOMD policy improvement steps (as in the main algorithm) for an independent, deviating agent while the policies of the rest of the population are frozen. As such, the quality of our approximation is limited by the number of policy improvement/expectation rounds, which must be restricted for the sake of running speed of the experiments. Moreover, since one of the findings of our paper is that networked agents can improve their policies faster than independent or centralised agents, especially when non-linear function approximation is used, it is arguably unsurprising that approximating the *BR* by an independently deviating agent sometimes gives an unclear and noisy metric. This includes the exploitability going below zero, which should not be possible if the policies and distributions are computed exactly.

Moreover, in coordination games (the setting for all tasks apart from the 'disperse' game), agents benefit by following the same behaviour as others, and so a deviating agent generally stands to gain less from a *BR* policy than it might in the non-coordination games on which many other works focus. For example, the return of a best-responding agent in the 'push object' game still depends on the extent to which other agents coordinate on which direction in which to push the box, meaning it cannot significantly increase its return by deviating. This means that the downward trajectory of the exploitability metric is less clear in our plots than in other works. This is likely why the approximated exploitability gets lower in the non-coordination 'disperse' task in Fig. 7 than in the other tasks. Given the limitations presented by approximating exploitability, we also provide the second metric to indicate the progress of learning.

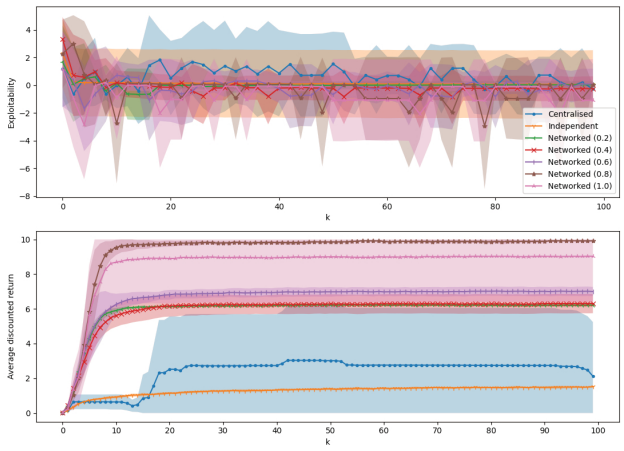

Figure 5: 'Target agreement' task, population-independent policies, 50x50 grid.

A.2.2 Average Discounted Return. We record the average discounted return of the agents' policies π_k^i during the M iterations - this allows us to observe that settings that converge to similar exploitability values may not have similar average agent returns, suggesting that some algorithms are better than others at finding not just NE, but preferable NE. See for example Figs. 1 and 5, where the networked agents converge to similar exploitability as the independent and centralised agents, but receive higher average returns.

A.3 Hyperparameters

See Table 1 for our hyperparameter choices. We can group our hyperparameters into those controlling the size of the experiment, those controlling the size of the Q-network, those controlling the number of iterations of each loop in the algorithms and those affecting the learning/policy updates or policy adoption.

In our experiments we generally want to demonstrate that our communication-based algorithms outperform the centralised and independent architectures by allowing policies that are estimated to be better performing to proliferate through the population, such that convergence occurs within fewer iterations and computationally faster, even when the Q-function is poorly approximated and/or the mean-field is poorly estimated, as is likely to be the case in real-world scenarios. Moreover we want to show that there is a benefit even to a small amount of communication, so that communication rounds themselves do not excessively add to time complexity. As such, we generally select hyperparameters at the lowest end of those we tested during development, to show that our algorithms are particularly successful given what might otherwise be considered 'undesirable' hyperparameter choices.

A.4 Additional Experiments

We provide additional experiments on large grids in Figs. 5, 6 and 7.

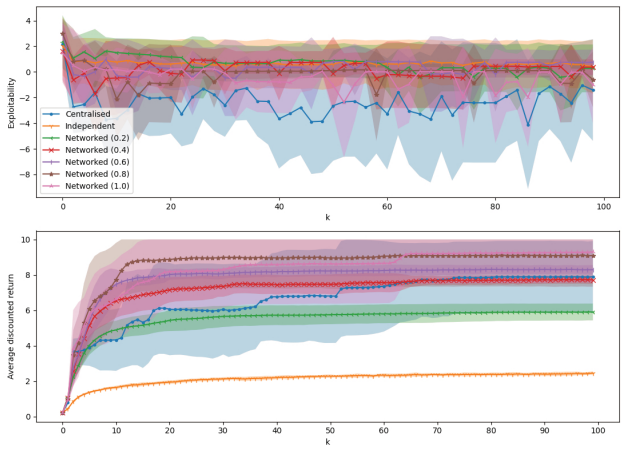

Figure 6: 'Cluster' task, population-independent policies, 50x50 grid.

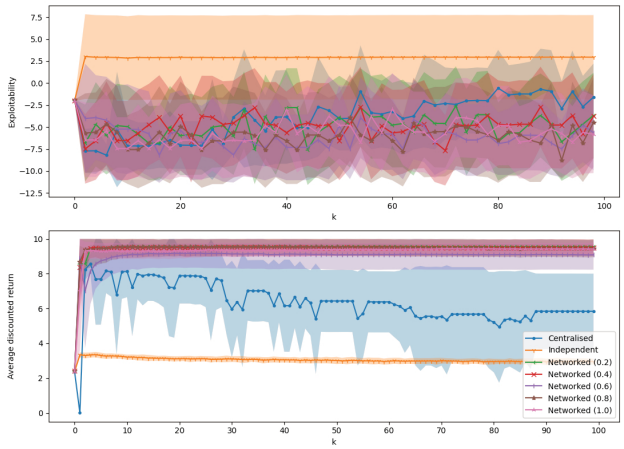

Figure 7: 'Disperse' task, population-independent policies, 100x100 grid.

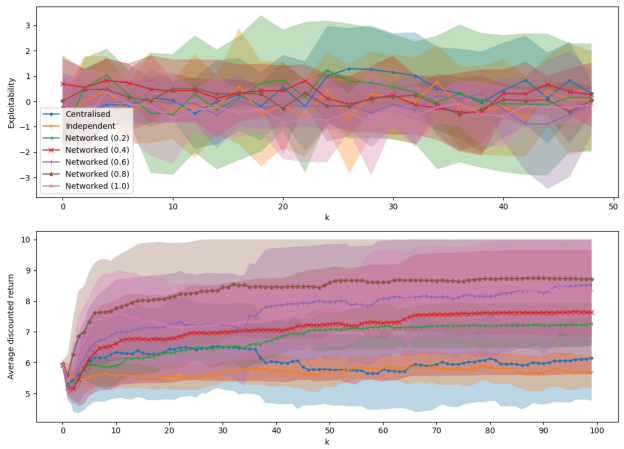

Figure 8: 'Push object' task, population-dependent policies with global observability of mean field, 10x10 grid.

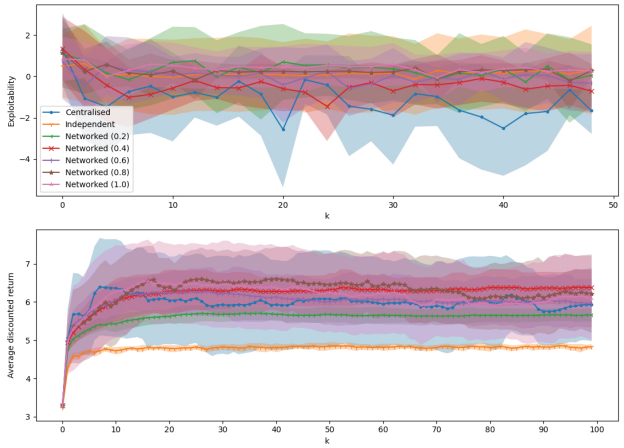

Figure 9: 'Evade' task, population-dependent policies with global observability of mean field, 10x10 grid.

In the 'target agreement' task in Fig. 5, the networked agents generally have lower exploitability than both centralised and independent agents, and significantly outperform the other architectures in terms of average return. As before, the margin by which the networked agents can outperform the centralised agents is much greater than in [3], showing that the benefits of the communication scheme are even greater in non-tabular settings.

In the 'cluster' task in Fig. 6, the networked agents obtain significantly higher return than the independent agents. While centralised agents have the lowest exploitability, networked agents of almost all communication radii outperform them in terms of average return, indicating that the communication scheme helps populations reach better performing equilibria.

In the 'disperse' task in Fig. 7, networked agents significantly outperform independent and centralised agents in terms of average return. They also outperform centralised agents in terms of exploitability, and significantly outperform independent agents in terms of exploitability. The fact that this happens in this non-coordination, competitive game shows that agents do have an incentive to communicate with each other even if they are self-interested.

B ADDITIONAL REMARKS ON MEAN-FIELD ESTIMATION ALGORITHMS

In our Algs. 2 and 3, agents share their local *counts* with neighbours on the communication network $\mathcal{G}_t^{comm}$, and only after the C_e communication rounds do they complete their estimated distribution by distributing the uncounted agents along their vectors. An alternative would be for each agent to immediately form a local *estimate* from their local count obtained via $\mathcal{G}_t^{obs}$ or $\mathcal{G}_t^{vis}$, which is only then communicated and updated via the communication network. However, we take the former approach to avoid poor local estimations spreading through the network and leading to widespread inaccuracies. Information that is certain (the count) is

spread as widely as possible, before being locally converted into an estimate of the total mean field. The same would be the case in our extension proposed in Sec. C for averaging noisy counts, i.e. only the counts would be averaged, with the estimates completed by distributing the remaining agents after the C_e communication rounds.

C LIMITATIONS AND FUTURE WORK

Our work follows the gold standard in MFGs by presenting experiments on grid world toy environments, albeit we show our algorithms are able to handle much larger and more complex games than prior work. Nevertheless future work lies in moving from these environments to real-world settings. In Sec. 6 we give theoretical results showing that our networked algorithm can outperform a centralised alternative. We leave more general analysis, such proof of convergence and sample guarantees in the function approximation setting, for future work.

Alg. 3 assumes that if a state s' is connected to s on the visibility graph $\mathcal{G}_t^{vis}$, an agent in s is able to *accurately* count all the agents in s', i.e. it either counts the exact total or cannot observe the state at all. We assume this for simplicity but it is not inherently the case, since a real-world agent may have only noisy observations even of others located nearby, due to imperfect sensors. We suggest two ways to deal with this. Firstly, if agents share unique IDs as in Alg. 2, then when communicating their vectors of collected IDs with each other via $\mathcal{G}_t^{comm}$, agents would gain the most accurate picture possible of all the agents that have been observed in a given state. However, as we note above, there are various reasons why sharing IDs might be undesirable, including privacy and scalability. If instead only counts are taken, and if the noise on each agents' count is assumed to be independent and, for example, subject to a Gaussian distribution, the algorithm can easily be updated such that communicating agents compute averages of their local and received counts to improve their accuracy, rather than simply using communication to fill in counts for previously unobserved states. (Note that we can also consider the original case without noise to involve averaging, since averaging identical values equates to using the original value). Since the algorithm is intended to aid in local estimation of the mean-field distribution, which is inherently approximate due to the uniform method for distributing the uncounted agents, we are not concerned with reaching exact consensus between agents on the communicated counts, so we do not require repeated averaging to ensure asymptotic convergence.

We may wish to consider more sophisticated methods for distributing the uncounted agents across states, in place of the current uniform distribution. Such choices may be domain-specific based on knowledge of a particular environment. For example, one might use the counts to perform Bayesian updates on a specific prior, where this prior may relate to the estimated mean-field distribution at the previous time step $t-1$. If agents seek to learn to predict the *evolution* of the mean field based on their own policy or by learning a model, the Bayesian prior may also be based on forward prediction from the estimated mean-field distribution at $t-1$. Future work lies in conducting experiments in all of these more general settings.

[25] notes that in grid-world settings such as those in our experiments, passing the (estimated or true global) mean-field distribution as a flat vector to the Q-network ignores the geometric structure of the problem. They therefore propose to create an embedding of the distribution by first passing the vector to a convolutional neural network, essentially treating the categorical distribution as an image. This technique is also followed in [34] (for their additional experiments, but not in the main body of their paper). As future work, we can test whether such a method improves the performance of our algorithms.

REFERENCES

[1] Talal Algumaei, Ruben Solozabal, Reda Alami, Hakim Hacid, Merouane Debbah, and Martin Takáč. 2023. Regularization of the policy updates for stabilizing Mean Field Games. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 361–372.

[2] Berkay Anahtarci, Can Deha Kariksiz, and Naci Saldi. 2023. Q-learning in regularized mean-field games. *Dynamic Games and Applications* 13, 1 (2023), 89–117.

[3] Patrick Benjamin and Alessandro Abate. 2023. Networked communication for decentralised agents in mean-field games. *arXiv preprint arXiv:2306.02766* (2023).

[4] Pierre Cardaliaguet, François Delarue, Jean-Michel Lasry, and Pierre-Louis Lions. 2015. The master equation and the convergence problem in mean field games. arXiv:1509.02505 [math.AP] https://arxiv.org/abs/1509.02505

[5] René Carmona, François Delarue, and Daniel Lacker. 2016. Mean Field Games with Common Noise. *The Annals of Probability* 44, 6 (2016), 3740–3803. http://www.jstor.org/stable/44072057

[6] Yufan Chen, Lan Wu, Renyuan Xu, and Ruixun Zhang. 2024. Periodic Trading Activities in Financial Markets: Mean-field Liquidation Game with Major-Minor Players. *arXiv preprint arXiv:2408.09505* (2024).

[7] Kai Cui, Christian Fabian, and Heinz Koeppl. 2023. Multi-Agent Reinforcement Learning via Mean Field Control: Common Noise, Major Agents and Approximation Properties. *arXiv preprint arXiv:2303.10665* (2023).

[8] Kai Cui and Heinz Koeppl. 2021. Approximately Solving Mean Field Games via Entropy-Regularized Deep Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1909–1917.

[9] Breno Cunha Queiroz and Daniel MacRae. 2024. Occlusion-based object transportation around obstacles with a swarm of miniature robots. *Swarm Intelligence* (2024), 1–29.

[10] A. E. Eiben and J. E. Smith. 2015. *What Is an Evolutionary Algorithm?* Springer Berlin Heidelberg, Berlin, Heidelberg, 25–48. https://doi.org/10.1007/978-3-662-44874-8_3

[11] Sriram Ganapathi Subramanian, Pascal Poupart, Matthew E Taylor, and Nidhi Hegde. 2020. Multi Type Mean Field Reinforcement Learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 411–419.

[12] Sriram Ganapathi Subramanian, Matthew E Taylor, Mark Crowley, and Pascal Poupart. 2021. Partially Observable Mean Field Reinforcement Learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*. 537–545.

[13] Xin Guo, Anran Hu, Renyuan Xu, and Junzi Zhang. 2023. A General Framework for Learning Mean-Field Games. *Mathematics of Operations Research* 48, 2 (2023), 656–686.

[14] Saeed Hadikhanloo. 2017. Learning in anonymous nonatomic games with applications to first-order mean field games. *arXiv preprint arXiv:1704.00378* (2017).

[15] Emma Hart, Andreas Steyven, and Ben Paechter. 2015. Improving Survivability in Environment-Driven Distributed Evolutionary Algorithms through Explicit Relative Fitness and Fitness Proportionate Communication. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (Madrid, Spain) *(GECCO '15)*. Association for Computing Machinery, New York, NY, USA, 169–176. https://doi.org/10.1145/2739480.2754688

[16] Anran Hu and Junzi Zhang. 2024. MF-OML: Online Mean-Field Reinforcement Learning with Occupation Measures for Large Population Games. *arXiv preprint arXiv:2405.00282* (2024). https://arxiv.org/abs/2405.00282

[17] Minyi Huang, Roland P. Malhamé, and Peter E. Caines. 2006. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems* 6, 3 (2006), 221 – 252.

[18] Jean-Michel Lasry and Pierre-Louis Lions. 2007. Mean Field Games. *Japanese Journal of Mathematics* 2, 1 (2007), 229–260.

[19] Mathieu Laurière. 2021. Numerical Methods for Mean Field Games and Mean Field Type Control. *Mean field games* 78, 221-282 (2021).

[20] Mathieu Laurière, Sarah Perrin, Matthieu Geist, and Olivier Pietquin. 2022. Learning Mean Field Games: A Survey. *arXiv preprint arXiv:2205.12944* (2022).

[21] Mathieu Laurière, Sarah Perrin, Sertan Girgin, Paul Muller, Ayush Jain, Theophile Cabannes, Georgios Piliouras, Julien Perolat, Romuald Elie, Olivier Pietquin, and Matthieu Geist. 2022. Scalable Deep Reinforcement Learning Algorithms for

Mean Field Games. In *Proceedings of the 39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 12078–12095. https://proceedings.mlr.press/v162/lauriere22a.html

[22] Behrang Monajemi Nejad, Sid Ahmed Attia, and Jorg Raisch. 2009. Max-consensus in a max-plus algebraic setting: The case of fixed communication topologies. In *2009 XXII International Symposium on Information, Communication and Automation Technologies*. 1–7. https://doi.org/10.1109/ICAT.2009.5348437

[23] Julien Perolat, Sarah Perrin, Romuald Elie, Mathieu Laurière, Georgios Piliouras, Matthieu Geist, Karl Tuyls, and Olivier Pietquin. 2021. Scaling up Mean Field Games with Online Mirror Descent. *arXiv preprint arXiv:2103.00623* (2021).

[24] Julien Pérolat, Sarah Perrin, Romuald Elie, Mathieu Laurière, Georgios Piliouras, Matthieu Geist, Karl Tuyls, and Olivier Pietquin. 2022. Scaling Mean Field Games by Online Mirror Descent. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems* (Virtual Event, New Zealand) *(AAMAS '22)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1028–1037.

[25] Sarah Perrin, Mathieu Laurière, Julien Pérolat, Romuald Élie, Matthieu Geist, and Olivier Pietquin. 2022. Generalization in mean field games by learning master policies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9413–9421.

[26] Sarah Perrin, Mathieu Laurière, Julien Pérolat, Matthieu Geist, Romuald Élie, and Olivier Pietquin. 2021. Mean Field Games Flock! The Reinforcement Learning Way. In *IJCAI*.

[27] Sarah Perrin, Julien Pérolat, Mathieu Laurière, Matthieu Geist, Romuald Elie, and Olivier Pietquin. 2020. Fictitious Play for Mean Field Games: Continuous Time Analysis and Applications. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1107, 15 pages.

[28] Naci Saldi, Tamer Başar, and Maxim Raginsky. 2018. Markov–Nash Equilibria in Mean-Field Games with Discounted Cost. *SIAM Journal on Control and Optimization* 56, 6 (2018), 4256–4287. https://doi.org/10.1137/17M1112583 arXiv:https://doi.org/10.1137/17M1112583

[29] Javad Soleimani, Reza Farhangi, and Gunes Karabulut Kurt. 2024. Distributed Critic-Based Neuro-Fuzzy Learning in Swarm Autonomous Vehicles. In *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*. 1–6. https://doi.org/10.1109/VTC2024-Fall63153.2024.10757965

[30] Jayakumar Subramanian and Aditya Mahajan. 2019. Reinforcement Learning in Stationary Mean-Field Games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) *(AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 251–259.

[31] Sriram Ganapathi Subramanian, Matthew E Taylor, Mark Crowley, and Pascal Poupart. 2022. Decentralized Mean Field Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9439–9447.

[32] Noureddine Toumi, Roland Malhame, and Jerome Le Ny. 2024. A mean field game approach for a class of linear quadratic discrete choice problems with congestion avoidance. *Automatica* 160 (2024), 111420. https://doi.org/10.1016/j.automatica.2023.111420

[33] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. 2020. Munchausen Reinforcement Learning. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 4235–4246. https://proceedings.neurips.cc/paper_files/paper/2020/file/2c6a0bae0f071cbbf0bb3d5b11d90a82-Paper.pdf

[34] Zida Wu, Mathieu Laurière, Samuel Jia Cong Chua, Matthieu Geist, Olivier Pietquin, and Ankur Mehta. 2024. Population-aware Online Mirror Descent for Mean-Field Games by Deep Reinforcement Learning. *arXiv preprint arXiv:2403.03552* (2024).

[35] Qiaomin Xie, Zhuoran Yang, Zhaoran Wang, and Andreea Minca. 2021. Learning While Playing in Mean-Field Games: Convergence and Optimality. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 11436–11447. https://proceedings.mlr.press/v139/xie21g.html

[36] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 5571–5580. https://proceedings.mlr.press/v80/yang18d.html

[37] Batuhan Yardim, Semih Cayci, Matthieu Geist, and Niao He. 2023. Policy Mirror Ascent for Efficient and Independent Learning in Mean Field Games. In *International Conference on Machine Learning*. PMLR, 39722–39754.

[38] Batuhan Yardim, Artur Goldman, and Niao He. 2024. When is Mean-Field Reinforcement Learning Tractable and Relevant? *arXiv preprint arXiv:2402.05757* (2024).

[39] Batuhan Yardim and Niao He. 2024. Exploiting Approximate Symmetry for Efficient Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2408.15173* (2024).

[40] Bora Yongacoglu, Gürdal Arslan, and Serdar Yüksel. 2022. Independent Learning in Mean-Field Games: Satisficing Paths and Convergence to Subjective Equilibria. *arXiv preprint arXiv:2209.05703* (2022).

[41] Muhammad Aneeq Uz Zaman, Alec Koppel, Sujay Bhatt, and Tamer Basar. 2023. Oracle-free Reinforcement Learning in Mean-Field Games along a Single Sample Path. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 10178–10206.

[42] Sihan Zeng, Sujay Bhatt, Alec Koppel, and Sumitra Ganesh. 2024. A Single-Loop Finite-Time Convergent Policy Optimization Algorithm for Mean Field Games (and Average-Reward Markov Decision Processes). *arXiv e-prints* (2024), arXiv–2408.

[43] Chenyu Zhang, Xu Chen, and Xuan Di. 2024. Stochastic Semi-Gradient Descent for Learning Mean Field Games with Population-Aware Function Approximation. *arXiv preprint arXiv:2408.08192* (2024).

Table 1: Hyperparameters

Hyperparameter	Value	Comment
Trials	10	We run 10 trials with different random seeds for each experiment. We plot the mean and standard deviation for each metric across the trials.
Gridsize	10x10 / 50x50 / 100x100	Experiments with population-dependent policies are run on the 10x10 grid (Figs. 3, 4, 8 and 9), while experiments on large state spaces are run on 50x50 and 100x100 grids (Figs. 1, 2, 5, 6 and 7).
Population	500	We chose 500 for our demonstrations to show that our algorithm can handle large populations, indeed often larger than those demonstrated in other mean-field works, especially for grid-world environments, while also being feasible to simulate wrt. time and computation constraints [3, 7, 8, 11–13, 30, 31, 34, 36, 40].
Number of neurons in input layer	cf. comment	The agent's position is represented by two concatenated one-hot vectors indicating the agent's row and column. An additional two such vectors are added for the shark's/object's position in the 'evade' and 'push object' tasks. For population-dependent policies, the mean-field distribution is a flattened vector of the same size as the grid. As such, the input size in the 'evade' and 'push object' tasks is $[(4 \times \text{dimension}) + (\text{dimension}^2)]$; in the other settings it is $[2 \times \text{dimension}]$.
Neurons per hidden layer	cf. comment	We draw inspiration from common rules of thumb when selecting the number of neurons in hidden layers, e.g. it should be between the number of input neurons and output neurons / it should be 2/3 the size of the input layer plus the size of the output layer / it should be a power of 2 for computational efficiency. Using these rules of thumb as rough heuristics, we select the number of neurons per hidden layer by rounding the size of the input layer down to the nearest power of 2. The layers are all fully connected.
Hidden layers	2	We experimented with 2 and 3 hidden layers in the Q-networks. While 3 hidden layers gave similar or slighly better performance, we selected 2 for increased computational speed for conducting our experiments.
Activation function	ReLU	This is a common choice in deep RL.
K	100	K is chosen to be large enough to see at least one of the metrics converging.
M	50	We tested M in {50,100} and found that the lower value was sufficient to achieve convergence while minimising training time. It may be possible to converge with even smaller choices of M.
L	50	We tested L in {50,100} and found that the lower value was sufficient to achieve convergence while minimising training time. It may be possible to converge with even smaller choices of L.
E	20	We tested E in {20,50,100}, and choose the lowest value to show the benefit to convergence even from very few evaluation steps. It may be possible to reduce this value further and still achieve similar results.
C_p	1	As in [3], we choose this value to show the convergence benefits brought by even a single communication round, even in networks that may have limited connectivity; higher choices are likely to have even better performance.
C_e	1	Similar to C_p, we choose this value to show the ability of our algorithm to appropriately estimate the mean field even with only a single communication round, even in networks that may have limited connectivity.
γ	0.9	Standard choice across RL literature.
τ_q	0.03	We tested τ_q in {0.01,0.02,0.03,0.04,0.05}, as well as linearly decreasing τ_q from $0.05 \rightarrow 0$ as k increases. However, only 0.03 gave stable increase in return. Note that this is the value also chosen in [33].
$\lvert B \rvert$	32	This is a common choice of batch size that trades off noisy updates and computational efficiency.
cl	-1	We use the same value as in [33].
ν	$L-1$	We tested ν in $\{1, 4, 20, L-1\}$. We found that in our setting, updating $\theta' \leftarrow \theta$ once per k iteration s.t. $\theta'_{k+1,l} = \theta_{k,l}\ \forall l$ gave sufficient learning that was similar to the other potential choices of ν, so we do this for simplicity, rather than arbitrarily choosing a frequency to update θ' during each k loop. Setting the target to be the policy from the previous iteration is similar to the method in [21]. Whilst [34] updates the target within the L loops for stability, we do not find this to be a problem in our experiments.
Optimiser	Adam	As in [33], we use the Adam optimiser with initial learning rate 0.01.
τ_k	cf. comment	τ_k increases linearly from 0.001 to 1 across the K iterations. This is a simplification of the annealing scheme used in [3]. Further optimising the annealing process may lead to better results.