Decentralized Mean Field Games

Sriram Ganapathi Subramanian [1, 3], **Matthew E. Taylor** [2, 4], **Mark Crowley** [1], **Pascal Poupart** [1, 3]

[1] University of Waterloo, Waterloo, Ontario, Canada
[2] University of Alberta, Edmonton, Alberta, Canada
[3] Vector Institute, Toronto, Ontario, Canada
[4] Alberta Machine Intelligence Institute (Amii), Edmonton, Alberta, Canada
s2ganapa@uwaterloo.ca, matthew.e.taylor@ualberta.ca, mcrowley@uwaterloo.ca, ppoupart@uwaterloo.ca

Abstract

Multiagent reinforcement learning algorithms have not been widely adopted in large scale environments with many agents as they often scale poorly with the number of agents. Using mean field theory to aggregate agents has been proposed as a solution to this problem. However, almost all previous methods in this area make a strong assumption of a centralized system where all the agents in the environment learn the same policy and are effectively indistinguishable from each other. In this paper, we relax this assumption about indistinguishable agents and propose a new mean field system known as *Decentralized Mean Field Games*, where each agent can be quite different from others. All agents learn independent policies in a decentralized fashion, based on their local observations. We define a theoretical solution concept for this system and provide a fixed point guarantee for a Q-learning based algorithm in this system. A practical consequence of our approach is that we can address a 'chicken-and-egg' problem in empirical mean field reinforcement learning algorithms. Further, we provide Q-learning and actor-critic algorithms that use the decentralized mean field learning approach and give stronger performances compared to common baselines in this area. In our setting, agents do not need to be clones of each other and learn in a fully decentralized fashion. Hence, for the first time, we show the application of mean field learning methods in fully competitive environments, large-scale continuous action space environments, and other environments with heterogeneous agents. Importantly, we also apply the mean field method in a ride-sharing problem using a real-world dataset. We propose a decentralized solution to this problem, which is more practical than existing centralized training methods.

1 Introduction

Most multiagent reinforcement learning (MARL) algorithms are not tractable when applied to environments with many agents (infinite in the limit) as these algorithms are exponential in the number of agents (Busoniu, Babuska, and De Schutter 2006). One exception is a class of algorithms that use the mean field theory (Stanley 1971) to approximate the many agent setting to a two agent setting, where the second agent is a mean field distribution of all agents representing the average effect of the population. This makes MARL algorithms tractable since, effectively, only two agents are being modelled. Lasry and Lions (2007) introduced the framework of a *mean field game* (MFG), which incorporates the mean field theory in MARL. In MARL, the mean field can be a population state distribution (Huang 2010) or action distribution (Yang et al. 2018) of all the other agents in the environment.

MFGs have three common assumptions. First, each agent does not have access to the local information of the other agents. However, it has access to accurate global information regarding the mean field of the population. Second, all agents in the environment are independent, homogeneous, and indistinguishable. Third, all agents maintain interactions with others only through the mean field. These assumptions (especially the first two) severely restrict the potential of using mean field methods in real-world environments. The first assumption is impractical, while the second assumption implies that all agents share the same state space, action space, reward function, and have the same objectives. Further, given these assumptions, prior works use centralized learning methods, where all agents learn and update a shared centralized policy. These two assumptions are only applicable to cooperative environments with extremely similar agents. Theoretically, the agent indices are omitted since all agents are interchangeable (Lasry and Lions 2007). We will relax the first two assumptions. In our case, the agents are not interchangeable and can each formulate their own policies during learning that differs from others. Also, we will not assume the availability of the immediate global mean field. Instead, agents only have local information and use modelling techniques (similar to opponent modelling common in MARL (Hernandez-Leal, Kartal, and Taylor 2019)) to effectively model the mean field during the training process. We retain the assumption that each agent's impact on the environment is infinitesimal (Huang 2010), and hence agents calculate best responses only to the mean field. Formulating best responses to each individual agent is intractable and unnecessary (Lasry and Lions 2007).

The solution concepts proposed by previous mean field methods have been either the centralized Nash equilibrium (Yang et al. 2018) or a closely related mean field equilibrium (Lasry and Lions 2007). These solution concepts are centralized as they require knowledge of the current policy of all other agents or other global information, even in non-cooperative environments. Verifying their existence is infeasible in many practical environments (Neumann 1928).

This work presents a new kind of mean field system, *Decentralized Mean Field Games* (DMFGs), which uses a decentralized information structure. This new formulation of the mean field system relaxes the assumption of agents' indistinguishability and makes mean field methods applicable to numerous real-world settings. Subsequently, we provide a decentralized solution concept for learning in DMFGs, which will be more practical than the centralized solution concepts considered previously. We also provide a fixed point guarantee for a Q-learning based algorithm in this system.

A 'chicken-and-egg' problem exists in empirical mean field reinforcement learning algorithms where the mean field requires agent policies, yet the policies cannot be learned without the global mean field (Yang and Wang 2020). We show that our formulation can address this problem, and we provide practical algorithms to learn in DMFGs. We test our algorithms in different types of many agent environments. We also provide an example of a ride-sharing application that simulates demand and supply based on a real-world dataset.

2 Background

Stochastic Game: An N-player stochastic (Markovian) game can be represented as a tuple $\langle \mathcal{S}, \mathcal{A}^1, \ldots, \mathcal{A}^N, r^1, \ldots, r^N, p, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}^j$ represents the action space of the agent $j \in \{1, \ldots, N\}$, and $r^j : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathcal{R}$ represents the reward function of j. Also, $p : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Omega(\mathcal{S})$ represents the transition probability that determines the next state given the current state and the joint action of all agents. Here $\Omega(\mathcal{S})$ is a probability distribution over the state space. In the stochastic game, agents aim to maximize the expected discounted sum of rewards, with discount factor $\gamma \in [0, 1)$.

Each agent j in the stochastic game formulates a policy, which is denoted by $\pi^j : \mathcal{S} \to \Omega(\mathcal{A}^j)$ where the joint policy is represented as $\boldsymbol{\pi} = [\pi^1, \ldots, \pi^N]$ for all $s \in \mathcal{S}$. Given an initial state s, the value function of the agent j under the joint policy $\boldsymbol{\pi}$ can be represented as $v^j(s|\boldsymbol{\pi}) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[r_t^j | s_0 = s, \boldsymbol{\pi}]$. The solution concept is the *Nash equilibrium* (NE) (Hu and Wellman 2003). This is represented by a joint policy $\boldsymbol{\pi}_* = [\pi_*^1, \ldots, \pi_*^N]$, such that, for all π^j, $v^j(s|\pi_*^j, \boldsymbol{\pi}_*^{-j}) \geq v^j(s|\pi^j, \boldsymbol{\pi}_*^{-j})$, for all agents $j \in \{1, \ldots, N\}$ and all states $s \in \mathcal{S}$. The notation $\boldsymbol{\pi}_*^{-j}$ represents the joint policy of all agents except the agent j.

Mean Field Game: MFG was introduced as a framework to solve the stochastic game when the number of agents N is very large (Lasry and Lions 2007; Huang et al. 2006). In this setting, calculating the best response to each individual opponent is intractable, so each agent responds to the aggregated state distribution $\mu_t \triangleq \lim_{N \to \infty} \frac{\sum_{j=1, j \neq i} \mathbf{1}(s_t^j)}{N}$, known as the mean field. Let $\boldsymbol{\mu} \triangleq \{\mu_t\}_{t=0}^{\infty}$. MFG assumes that all agents are identical (homogeneous), indistinguishable, and interchangeable (Lasry and Lions 2007). Given this assumption, the environment changes to a single-agent stochastic control problem where all agents share the same policy (Saldi, Basar, and Raginsky 2018). Hence, $\pi^1 = \cdots = \pi^N = \boldsymbol{\pi}$. The theoretical formulation focuses on a representative player, and the solution of this player (optimal policy) obtains the solution for the entire system. The value function of a representative agent can be given as $V(s, \boldsymbol{\pi}, \boldsymbol{\mu}) \triangleq \mathbb{E}_{\boldsymbol{\pi}}\big[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, \mu_t) | s_0 = s\big]$, where s and a are the state and action of the representative agent, respectively. The transition dynamics is represented as $P(s_t, a_t, \mu_t)$, where the dependence is on the mean field distribution. All agents share the same reward function $r(s_t, a_t, \mu_t)$. The action comes from the policy $\pi_t(a_t|s_t, \mu_t)$.

The solution concept for the MFG is the mean field equilibrium (MFE). A tuple $(\boldsymbol{\pi}_{MFG}^*, \boldsymbol{\mu}_{MFG}^*)$ is a mean field equilibrium if, for any policy $\boldsymbol{\pi}$, an initial state $s \in \mathcal{S}$ and given mean field $\boldsymbol{\mu}_{MFG}^*$, $V(s, \boldsymbol{\pi}_{MFG}^*, \boldsymbol{\mu}_{MFG}^*) \geq V(s, \boldsymbol{\pi}, \boldsymbol{\mu}_{MFG}^*)$. Additionally, $\boldsymbol{\mu}_{MFG}^*$ is the mean field obtained when all agents play the same $\boldsymbol{\pi}_{MFG}^*$ at each $s \in \mathcal{S}$.

Mean Field Reinforcement Learning (MFRL): MFRL, introduced by Yang et al. (2018), is another approach for learning in stochastic games with a large number of agents. Here, the empirical mean action is used as the mean field, which each agent then uses to update its Q-function and Boltzmann policy. Each agent is assumed to have access to the global state, and it takes local action from this state. In MFRL, the Q-function of the agent j is updated as,

$$\begin{aligned} &Q^j(s_t, a_t^j, \mu_t^a) \\ &\quad = (1 - \alpha) Q^j(s_t, a_t^j, \mu_t^a) + \alpha[r_t^j + \gamma v^j(s_{t+1})] \end{aligned} \tag{1}$$

where

$$v^j(s_{t+1}) = \sum_{a_{t+1}^j} \pi^j(a_{t+1}^j | s_{t+1}, \mu_t^a) Q^j(s_{t+1}, a_{t+1}^j, \mu_t^a) \tag{2}$$

$$\mu_t^a = \frac{1}{\mathcal{N}} \sum_j a_t^j, a_t^j \sim \pi^j(\cdot | s_t, \mu_{t-1}^a) \tag{3}$$

$$\pi^j(a_t^j | s_t, \mu_{t-1}^a) = \frac{\exp(-\hat{\beta} Q^j(s_t, a_t^j, \mu_{t-1}^a)}{\sum_{a_t^{j'} \in A^j} \exp(-\hat{\beta} Q^j(s_t, a_t^{j'}, \mu_{t-1}^a))} \tag{4}$$

where s_t is the global old state, s_{t+1} is the global resulting state, r_t^j is the reward of j at time t, v^j is the value function of j, $\mathcal{N}$ is the total number of agents, and $\hat{\beta}$ represents the Boltzmann parameter. The action a^j is assumed to be discrete and represented using one-hot encoding. Like stochastic games, MFRL uses the NE as the solution concept. The global mean field μ_t^a captures the action distribution of all agents. To address this global limitation, Yang et al. (2018) specify mean field calculation over certain neighbourhoods for each agent. However, an update using Eq. 3 requires each agent to have access to all other agent policies or the global mean field to use the centralized concept (NE). We omit an expectation in Eq. 2 since Yang et al. (2018) guaranteed that their updates will be greedy in the limit with infinite exploration (GLIE).

From Eq. 3 and Eq. 4, it can be seen that the current action depends on the mean field and the mean field depends on the current action. To resolve this 'chicken-and-egg' problem, Yang et al. (2018) simply use the previous mean field action to decide the current action, as in Eq. 4. This can lead to a loss of performance since the agents are formulating best responses to the previous mean field action μ_{t-1}^a, while they are expected to respond to the current mean field action μ_t^a.

3 Related Work

MFGs were first proposed in Huang, Caines, and Malhamé (2003), while a comprehensive development of the system and principled application methods were given later in Lasry and Lions (2007). Subsequently, learning algorithms were proposed for this framework. Subramanian and Mahajan (2019) introduce a restrictive form of MFG (known as stationary MFG) and provide a model-free policy-gradient (Sutton et al. 1999; Konda and Tsitsiklis 1999) algorithm along with convergence guarantees to a local Nash equilibrium. On similar lines, Guo et al. (2019) provide a model-free Q-learning algorithm (Watkins and Dayan 1992) for solving MFGs, also in the stationary setting. The assumptions in these works are difficult to verify in real-world environments. Particularly, Guo et al. (2019) assume the presence of a game engine (simulator) that accurately provides mean field information to all agents at each time step, which is not practical in many environments. Further, other works depend on fictitious play updates for the mean field parameters (Hadikhanloo and Silva 2019; Elie et al. 2020), which involves the strong assumption that opponents play stationary strategies. All these papers use the centralized setting for the theory and the experiments.

Prior works (Gomes, Mohr, and Souza 2010; Adlakha, Johari, and Weintraub 2015; Saldi, Basar, and Raginsky 2018) have established the existence of a (centralized) mean field equilibrium in the discrete-time MFG under a discounted cost criterion, in finite and infinite-horizon settings. Authors have also studied the behaviour of iterative algorithms and provided theoretical analysis for learning of the non-stationary (centralized) mean field equilibrium in infinite-horizon settings (Więcek and Altman 2015; Więcek 2020; Anahtarci, Kariksiz, and Saldi 2019). We provide similar guarantees in the decentralized setting with possibly heterogeneous agents.

Yang et al. (2018) introduces MFRL that uses a mean field approximation through the empirical mean action, and provides two practical algorithms that show good performance in large MARL settings. The approach is model-free and the algorithms do not need strong assumptions regarding the nature of the environment. However, they assume access to global information that needs a centralized setting for both theory and experiments. MFRL has been extended to multiple types (Subramanian et al. 2020) and partially observable environments (Subramanian et al. 2021). However, unlike us, Subramanian et al. (2020) assumes that agents can be divided into a finite set of types, where agents within a type are homogeneous. Subramanian et al. (2021) relaxes the assumption of global information in MFRL, however, it still uses the Nash equilibrium as the solution concept. Further, that work contains some assumptions regarding the existence of conjugate priors, which is hard to verify in real-world environments. Additional related work is provided in Appendix L.

4 Decentralized Mean Field Game

The DMFG model is specified by $\langle \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, p, \boldsymbol{R}, \mu_0 \rangle$, where $\boldsymbol{\mathcal{S}} = \mathcal{S}^1 \times \cdots \times \mathcal{S}^N$ represents the state space and $\boldsymbol{\mathcal{A}} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$ represents the joint action space. Here, $\mathcal{S}^j$ represents the state space of an agent $j \in \{1, \ldots, N\}$ and $\mathcal{A}^j$ represents the action space of j. As in MFGs, we are considering the infinite population limit of the game, where the set of agents N, satisfy $N \to \infty$. Similar to the MFG formulation in several prior works (Lasry and Lions 2007; Huang et al. 2006; Saldi, Basar, and Raginsky 2018), we will specify that both the state and action spaces are Polish spaces. Particularly, $\mathcal{A}^j$ for all agents j, is a compact subset of a finite dimensional Euclidean space $\Re^d$ with the Euclidean distance norm $||\cdot||$. Since all agents share the same environment, for simplicity, we will also assume that the state spaces of all the agents are the same $\mathcal{S}^1 = \cdots = \mathcal{S}^N = \mathcal{S}$ and are locally compact. Since the state space is a complete separable metric space (Polish space), it is endowed with a metric d_X. The transition function $p : \mathcal{S} \times \mathcal{A}^j \times \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{S})$ determines the next state of any j given the current state and action of j, and the probability distribution of the state in the system (represented by the mean field). The reward function is represented as a set $\boldsymbol{R} = \{R^1, \ldots, R^N\}$, where, $R^j : \mathcal{S} \times \mathcal{A}^j \times \mathcal{P}(\mathcal{S}) \to [0, \infty)$ is the reward function of j.

Recall that a DMFG has two major differences as compared to MFG and MFRL. 1) DMFG does not assume that the agents are indistinguishable and homogeneous (agent indices are retained). 2) DMFG does not assume that each agent can access the global mean field of the system. However, each agents' impact on the environment is infinitesimal, and therefore all agents formulate best responses only to the mean field of the system (no per-agent modelling is required).

As specified, in DMFG, the transition and reward functions for each agent depends on the mean field of the environment, represented by $\boldsymbol{\mu} \triangleq (\mu_t)_{t \geq 0}$, with the initial mean field represented as μ_0. For DMFG, mean field can either correspond to the state distribution $\mu_t \triangleq \lim_{N \to \infty} \frac{\sum_{j=1}^{N} \mathbf{1}(s_t^j)}{N}$ or the action distribution μ_t^a as in Eq. 3. Without loss of generality we use the mean field as the state distribution μ_t (represented by $\mathcal{P}(\mathcal{S})$), as done in prior works (Lasry and Lions 2007; Elliott, Li, and Ni 2013). However, our setting and theoretical results will hold for the mean field as action distribution μ_t^a as well.

In the DMFG, each agent j will not have access to the true mean field of the system and instead use appropriate techniques to actively model the mean field through exploration. The agent j holds an estimate of the actual mean field represented by μ_t^j. Let $\boldsymbol{\mu}^j \triangleq (\mu_t^j)_{t \geq 0}$. Let, $\mathcal{M}$ be used to denote the set of mean fields $\{\boldsymbol{\mu^j} \in \mathcal{P}(\mathcal{S})\}$. A Markov policy for an agent j is a stochastic kernel on the action space $\mathcal{A}^j$ given the immediate local state (s_t^j) and the agent's current estimated mean field μ_t^j, i.e. $\pi_t^j : \mathcal{S} \times \mathcal{P}(\mathcal{S}) \to \mathcal{P}(\mathcal{A}^j)$, for each $t \geq 0$. Alternatively, a non-Markov policy will depend on the entire state-action history of game play. We will use Π^j to denote a set of all policies (both Markov and non-Markov) for the agent j. Let s_t^j represent the state of an agent j at time t and a_t^j represent the action of j at t. Then an agent j tries to maximize the objective function given by the following equation (where r^j denotes the immediate reward obtained by the agent j and $\beta \in [0, 1)$ denotes the discount factor),

$$J_{\boldsymbol{\mu}}^j(\pi^j) \triangleq \mathbb{E}^{\pi^j}[\textstyle\sum_{t=0}^{\infty} \beta^t r^j(s_t^j, a_t^j, \mu_t)]. \tag{5}$$

In line with prior works (Yang et al. 2018; Subramanian et al. 2020), we assume that each agent's sphere of influence is restricted by its neighbourhood, where it conducts

exploration. Using this assumption, we assert that, after a finite t, the mean field estimate will accurately reflect the true mean field for j, in its neighbourhood denoted as $\mathcal{N}^j$. This assumption specifies that agents have full information in their neighbourhood, and they can use modelling techniques to obtain accurate mean field information within the neighbourhood (also refer to flocking from Perrin et al. (2021)).

Assumption 1. *There exists a finite time T and a neighbourhood $\mathcal{N}^j$, such that for all $t > T$, the mean field estimate of an agent $j \in 1, \dots, N$ satisfies ($\forall s^j \in \mathcal{N}^j$) $\boldsymbol{\mu}^j(s^j) = \boldsymbol{\mu}(s^j)$. Also, $\forall s^j \in \mathcal{N}^j$, we have, $p(\cdot|s_t^j, a_t^j, \mu_t^j) = p(\cdot|s_t^j, a^j, \mu_t)$ and $r^j(\cdot|s_t^j, a_t^j, \mu_t^j) = r^j(\cdot|s_t^j, a_t^j, \mu_t)$.*

Let us define a set $\Phi : \mathcal{M} \to 2^{\Pi}$ as $\Phi(\boldsymbol{\mu^j}) = \{\pi^j \in \Pi^j : \pi^j \text{ is optimal for } \boldsymbol{\mu^j}\}$. Conversely, for j, we define a mapping $\Psi : \boldsymbol{\Pi} \to \mathcal{M}$ as, given a policy $\pi^j \in \Pi^j$, the mean field state estimate $\boldsymbol{\mu}^j \triangleq \Psi(\pi^j)$ can be constructed as,

$$\mu_{t+1}^j(\cdot) = \int_{\mathcal{S}\times\mathcal{A}^j} p(\cdot|s_t^j, a_t^j, \mu_t)\mathcal{P}^{\pi^j}(a_t^j|s_t^j, \mu_t^j)\mu_t^j(s_t^j). \tag{6}$$

Here $\mathcal{P}^{\pi^j}$ is a probability measure induced by π^j. Later (in Theorem 1) we will prove that restricting ourselves to Markov policies is sufficient in a DMFG, and hence $\mathcal{P}^{\pi^j} = \pi^j$.

Now, we can define the *decentralized mean field equilibrium* (DMFE) which is the solution concept for this game.

Definition 1. *The decentralized mean field equilibrium of an agent j is represented as a pair $(\pi_*^j, \mu_*^j) \in \Pi^j \times \mathcal{M}$ if $\pi_*^j \in \Phi(\mu_*^j)$ and $\mu_*^j = \Psi(\pi_*^j)$. Here π_*^j is the best response to μ_*^j and μ_*^j is the mean field estimate of j when it plays π_*^j.*

The important distinction between DMFE and centralized concepts, such as NE and MFE, is that DMFE does not rely on the policy information of other agents. MFE requires all agents to play the same policy, and NE requires all agents to have access to other agents' policies. DMFE has no such constraints. Hence, this decentralized solution concept is more practical than NE and MFE. In Appendix F, we summarize the major differences between the DMFG, MFG and MFRL.

5 Theoretical Results

We provide a set of theorems that will first guarantee the existence of the DMFE in a DMFG. Further, we will show that a simple Q-learning update will converge to a fixed point representing the DMFE. We will borrow relevant results from prior works in centralized MFGs in our theoretical guarantees. Particularly, we aim to adapt the results and proof techniques in works by Saldi, Basar, and Raginsky (2018), Lasry and Lions (2007), and Anahtarci, Kariksiz, and Saldi (2019) to the decentralized setting. The statements of all our theorems are given here, while the proofs are in Appendices A – E.

Similar to an existing result from centralized MFG (Lasry and Lions 2007), in the DMFG, restricting policies to only Markov policies would not lead to any loss of optimality. We use Π_M^j to denote the set of Markov policies for the agent j.

Theorem 1. *For any mean field, $\boldsymbol{\mu} \in \mathcal{M}$, and an agent $j \in \{1, \dots, N\}$, we have,*

$$\sup_{\pi^j \in \Pi^j} J_{\boldsymbol{\mu}}^j(\pi^j) = \sup_{\pi^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\pi^j). \tag{7}$$

Next, we show the existence of a DMFE under a set of assumptions similar to those previously used in the centralized MFG (Lasry and Lions 2007; Saldi, Basar, and Raginsky 2018). The assumptions pertain to bounding the reward function and imposing restrictions on the nature of the mean field (formal statements in Appendix B). We do not need stronger assumptions than those previously considered for the MFGs.

Theorem 2. *An agent $j \in \{1, \dots, N\}$ in the DMFG admits a decentralized mean field equilibrium $(\pi_*^j, \mu_*^j) \in \Pi^j \times \mathcal{M}$.*

We use $\mathcal{C}$ to denote a set containing bounded functions in $\mathcal{S}$. Now, we define a decentralized mean field operator (H),

$$H : \mathcal{C} \times \mathcal{P}(\mathcal{S}) \ni (Q^j, \boldsymbol{\mu}^j) \to (H_1(Q^j, \boldsymbol{\mu}^j), H_2(Q^j, \boldsymbol{\mu}^j)) \in \mathcal{C} \times \mathcal{P}(\mathcal{S}) \tag{8}$$

where

$$H_1(Q^j, \boldsymbol{\mu}^j)(s_t^j, a_t^j) \triangleq r^j(s_t^j, a_t^j, \mu_t) + \beta \int_{\mathcal{S}} Q_{\max_{a^j}}^j(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j|s_t^j, a_t^j, \mu_t) \tag{9}$$

$$H_2(Q^j, \boldsymbol{\mu}^j)(\cdot) \triangleq \int_{\mathcal{S}\times\mathcal{A}^j} p\Big(\cdot\,|s_t^j, \pi^j(s_t^j, Q_t^j, \mu_t^j), \mu_t\Big)\mu_t^j(s) \tag{10}$$

for an agent j. Here, π^j is a maximiser of the operator H_1.

For the rest of the theoretical results, we consider a set of assumptions different from those needed for Theorem 2. Here we assume that the reward and transition functions are Lipschitz continuous with a suitable Lipschitz constant. The Lipschitz continuity assumption is quite common in the mean field literature (Yang et al. 2018; Lasry and Lions 2007; Subramanian et al. 2020). We also consider some further assumptions regarding the nature of gradients of the value function (formal statements in Appendix C). All assumptions are similar to those considered before for the analysis in the centralized MFGs (Lasry and Lions 2007; Anahtarci, Kariksiz, and Saldi 2019; Huang 2010). First, we provide a theorem regarding the nature of operator H. Then, we provide another theorem showing that H is a contraction.

Theorem 3. *The decentralized mean field operator H is well-defined, i.e., this operator maps $\mathcal{C} \times \mathcal{P}(\mathcal{S})$ to itself.*

Theorem 4. *Let $\mathcal{B}$ represent the space of bounded functions in $\mathcal{S}$. Then the mapping $H : \mathcal{C} \times \mathcal{P}(\mathcal{S}) \to \mathcal{C} \times \mathcal{P}(\mathcal{S})$ is a contraction in the norm of $\mathcal{B}(\mathcal{S})$.*

Since H is a contraction, by the Banach fixed point theorem (Shukla, Balasubramanian, and Pavlović 2016), we can obtain that fixed point using the Q iteration algorithm given by Alg. 1. We provide this result in the next theorem.

Theorem 5. *Let the Q-updates in Algorithm 1 converge to (Q_*^j, μ_*^j) for an agent $j \in \{1, \dots, N\}$. Then, we can construct a policy π_*^j from Q_*^j using the relation,*

$$\pi_*^j(s^j) = \arg\max_{a^j \in \mathcal{A}^j} Q_*^j(s^j, a^j, \mu_*^j). \tag{11}$$

Then the pair (π_*^j, μ_*^j) is a DMFE.

Hence, from the above results, we have proved that a DMFG admits a DMFE, and an iterative algorithm using Q-updates can arrive at this equilibrium. This provides a fixed point for Alg. 1, to which the Q-values converge.

Algorithm 1: Q-learning for DMFG

1: For each agent $j \in \{1, \ldots, N\}$, start with initial Q-function Q_0^j and the initial mean field state estimate μ_0^j
2: **while** $(Q_n^j, \mu_n^j) \neq (Q_{n-1}^j, \mu_{n-1}^j)$ **do**
3: $\quad (Q_{n+1}^j, \mu_{n+1}^j) = H(Q_n^j, \mu_n^j)$
4: **end while**
5: Return the fixed point (Q_*^j, μ_*^j) of H

6 Algorithms

We will apply the idea of decentralized updates to the model-free MFRL framework. We modify the update equations in MFRL (Section 2) and make them decentralized, where agents only observe their local state and do not have access to the immediate global mean field. Our new updates are:

$$\begin{aligned} &Q^j(s_t^j, a_t^j, \mu_t^{j,a}) \\ &\quad = (1-\alpha)Q^j(s_t^j, a_t^j, \mu_t^{j,a}) + \alpha[r_t^j + \gamma v^j(s_{t+1}^j)] \end{aligned} \tag{12}$$

where

$$v^j(s_{t+1}^j) = \sum_{a_{t+1}^j} \pi^j(a_{t+1}^j | s_{t+1}^j, \mu_{t+1}^{j,a}) Q^j(s_{t+1}^j, a_{t+1}^j, \mu_{t+1}^{j,a}) \tag{13}$$

$$\mu_t^{j,a} = f^j(s_t^j, \hat{\mu}_{t-1}^{j,a}) \tag{14}$$

$$\text{and } \pi^j(a_t^j | s_t^j, \mu_t^{j,a}) = \frac{\exp(-\hat{\beta} Q^j(s_t^j, a_t^j, \mu_t^{j,a}))}{\sum_{a_t^{j'} \in A^j} \exp(-\hat{\beta} Q^j(s_t^j, a_t^{j'}, \mu_t^{j,a}))} \tag{15}$$

Here, s_t^j is the local state and $\mu_t^{j,a}$ is the mean field action estimate for the agent j at time t and $\hat{\mu}_{t-1}^{j,a}$ is the observed local mean field action of j at $t-1$. Other variables have the same meaning as Eq. 1 – Eq. 4. In Eq. 14, the mean field estimate for j is updated using a function of the current state and the previous local mean field. Opponent modelling techniques commonly used in MARL (Hernandez-Leal, Kartal, and Taylor 2019) can be used here. We use the technique of He and Boyd-Graber (2016), that used a neural network to model the opponent agent(s). In our case, we use a fully connected neural network (2 Relu layers of 50 nodes and an output softmax layer) to model the mean field action. The network takes the current state and previous mean field action as inputs and outputs the estimated current mean field. This network is trained using a mean square error between the estimated mean field action from the network (Eq. 14) and the observed local mean field (local observation of actions of other agents $\hat{\mu}_t^{j,a}$) after action execution. The policy in Eq. 15 does not suffer from the 'chicken-and-egg' problem of Eq. 4 since it depends on the current mean field estimate, unlike MFRL which used the previous global mean field in Eq. 4.

We provide a neural network-based Q-learning implementation for our update equations, namely *Decentralized Mean Field Game Q-learning* (DMFG-QL), and an actor-critic implementation, *Decentralized Mean Field Game Actor-Critic* (DMFG-AC). Detailed description of the algorithms are in Appendix H (see Algs. 2 and 3). A complexity analysis is in Appendix K, and hyperparameter details are in Appendix J.

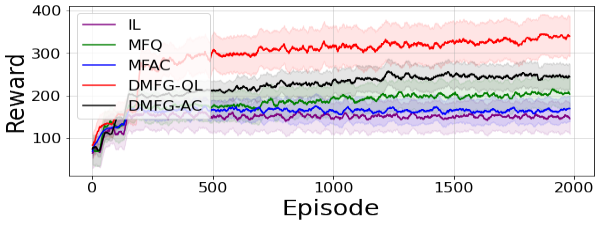
7 Experiments and Results

In this section we study the performance of our algorithms. The code for experiments has been open-sourced (Subramanian 2021). We provide the important elements of our domains here, while the complete details are in Appendix I.

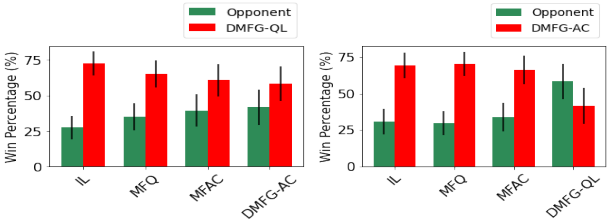
The first five domains belong to the MAgent environment (Zheng et al. 2018). We run the experiments in two phases, training and execution. Analogous to experiments conducted in previous mean field studies (Yang et al. 2018; Subramanian et al. 2020), all agents train against other agents playing the same algorithm for 2000 games. This is similar to multiagent training using self-play (Shoham, Powers, and Grenager 2003). The trained agents then enter into an execution phase, where the trained policies are simply executed. The execution is run for 100 games, where algorithms may compete against each other. We consider three baselines, independent Q-learning (IL) (Tan 1993), mean field Q-learning (MFQ) (Yang et al. 2018), and mean field actor-critic (MFAC) (Yang et al. 2018). Each agent in our implementations learns in a decentralized fashion, where it maintains its own networks and learns from local experiences. This is unlike centralized training in prior works (Yang et al. 2018; Guo et al. 2019). We repeat experiments 30 times, and report the mean and standard deviation. Wall clock times are given in Appendix M.

First, we consider the mixed cooperative-competitive Battle game (Zheng et al. 2018). This domain consists of two teams of 25 agents each. Each agent is expected to cooperate with agents within the team and compete against agents of the other team to win the battle. For the training phase, we plot the cumulative rewards per episode obtained by the agents of the first team for each algorithm in Fig. 1(a). The performance of the second team is also similar (our environment is not zero-sum). From the results, we see that DMFG-QL performs best while the others fall into a local optimum and do not get the high rewards. The DMFG-AC algorithm comes second. It has been noted previously (Yang et al. 2018) that Q-learning algorithms often perform better compared to their actor-critic counterparts in mean field environments. MFQ and MFAC (using the previous mean field information) performs poorly compared to DMFG-QL and DMFG-AC (using the current estimates). Finally, IL loses out to others due to its independent nature. In execution, one team trained using one algorithm competes against another team from a different algorithm. We plot the percentage of games won by each algorithm in a competition against DMFG-QL and DMFG-AC. A game is won by the team that kills more of its opponents. The performances are in Fig. 1(b) and (c), where DMFG-QL performs best. In Appendix G, we show that DFMG-QL can accurately model the true mean field in the Battle game.

The second domain is the heterogeneous Combined Arms environment. This domain is a mixed setting similar to Battle except that each team consists of two different types of agents, ranged and melee, with distinct action spaces. Each team has 15 ranged and 10 melee agents. This environment is different from those considered in Subramanian et al. (2020), which formulated each team as a distinct type, where agents within a team are homogeneous. The ranged agents are faster and attack further, but can be killed quickly. The melee agents are slower but are harder to kill. We leave out MFQ and MFAC



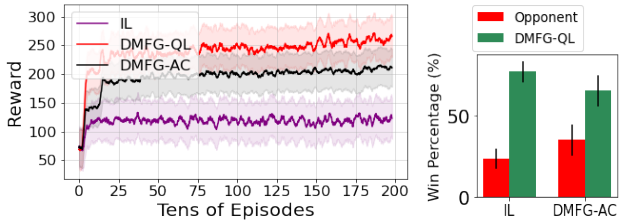
(a) Training



(b) Execution vs. DMFG-QL (c) Execution vs. DMFG-AC

Figure 1: Battle results. In (a) solid lines show average and shaded regions represent standard deviation. In (b) and (c) bars are average and black lines represent standard deviation.

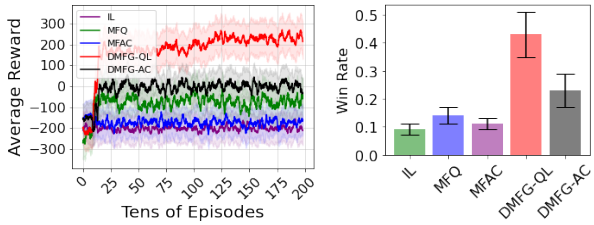
for this experiment, since both these algorithms require the presence of fully homogeneous agents. The experimental procedure is the same as in Battle. From the results we see that DMFG-QL performs best in both phases (see Fig. 2).



(a) Training (b) Execution

Figure 2: Combined Arms results

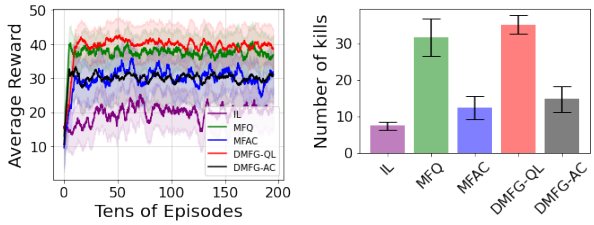
Next is the fully competitive Gather environment. This contains 30 agents trying to capture limited food. All the agents compete against each other for capturing food and could resort to killing others when the food becomes scarce. We plot the average rewards obtained by each of the five algorithms in the training phase (Fig. 3(a)). DMFG-QL once again obtains the maximum performance. In competitive environments, actively formulating the best responses to the current strategies of opponents is crucial for good performances. Predictably, the MFQ and MFAC algorithms (relying on previous information) lose out. For execution, we sample (at random) six agents from each of the five algorithms to make a total of 30. We plot the percentage of games won by each algorithm in a total of 100 games. A game is determined to have been won by the agent obtaining the most rewards. Again, DMFG-QL shows the best performance during execution (Fig. 3(b)).



(a) Training (b) Execution

Figure 3: Gather results

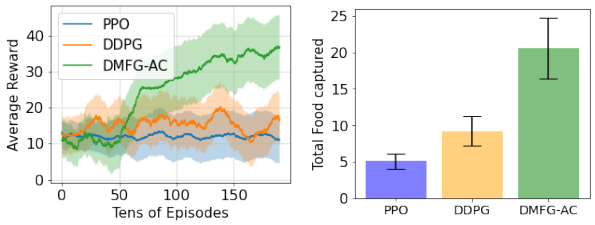
The next domain is a fully cooperative Tiger-Deer environment. In this environment, a team of tigers aims to kill deer. The deer are assumed to be part of the environment moving randomly, while the tigers are agents that learn to coordinate with each other to kill the deer. At least two tigers need to attack a deer in unison to gain large rewards. Our environment has 20 tigers and 101 deer. In the training phase, we plot the average reward obtained by the tigers (Fig. 4(a)). The performance of MFQ almost matches that of DMFG-QL and the performance of DMFG-AC matches MFAC. In a cooperative environment, best responses to actively changing strategies of other agents are not as critical as in competitive environments. Here all agents aid each other and using the previous time information (as done in MFQ and MFAC) does not hurt performance as much. For execution, a set of 20 tigers from each algorithm execute their policy for 100 games. We plot the average number of deer killed by the tigers for each algorithm. DMFG-QL gives the best performance (Fig. 4(b)).



(a) Training (b) Execution

Figure 4: Tiger-Deer results

The next domain is the continuous action Waterworld domain, first introduced by Gupta, Egorov, and Kochenderfer (2017). This is also a fully cooperative domain similar to Tiger-Deer, where a group of 25 pursuer agents aim to capture a set of food in the environment while actively avoiding poison. The action space corresponds to a continuous thrust variable. We implement DMFG-AC in this domain, where the mean field is a mixture of Dirac deltas of actions taken by all agents. The experimental procedure is the same as Tiger-Deer. For this continuous action space environment, we use proximal policy optimization (PPO) (Schulman et al. 2017) and deep deterministic policy gradient (DDPG) (Lillicrap et al. 2016), as baselines. We see that DMFG-AC obtains the



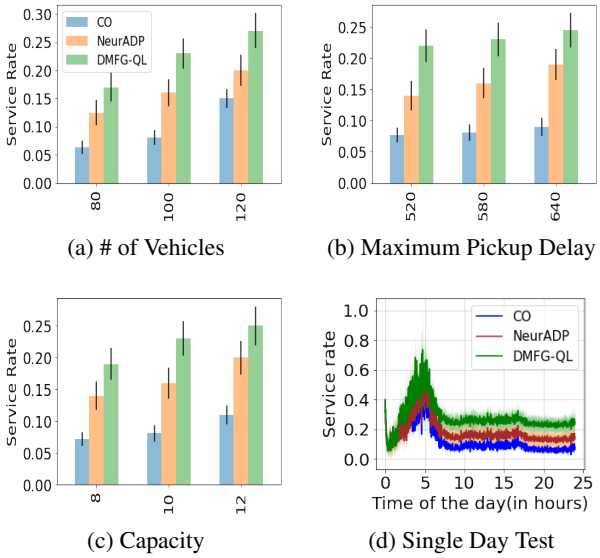
(a) Training (b) Execution

Figure 5: Waterworld results

best performance in both phases (refer to Fig. 5(a) and (b)).

Our final environment is a real-world *Ride-pool Matching Problem* (RMP) introduced by Alonso-Mora et al. (2017). This problem pertains to improving the efficiency of vehicles satisfying ride requests as part of ride-sharing platforms such as UberPool. In our environment, ride requests come from the open source New York Yellow Taxi dataset (NYYellowTaxi 2016). The road network (represented as a grid with a finite set of nodes or road intersections) contains a simulated set of vehicles (agents) that aim to serve the user requests. Further details about this domain are in Appendix I. We consider two baselines in this environment. The first is the method from Alonso-Mora et al. (2017), which used a constrained optimization (CO) approach to match ride requests to vehicles. This approach is hard to scale and is myopic in assigning requests (it does not consider future rewards). The second baseline is the Neural Approximate Dynamic Programming (NeurADP) method from Shah, Lowalekar, and Varakantham (2020), which used a (centralized) DQN algorithm to learn a value function for effective mapping of requests. This approach assumes all agents are homogenous (i.e., having the same capacity and preferences), which is impractical. To keep comparisons fair, we consider a decentralized version of NeurADP as our baseline. Finally, we implement DMFG-QL for this problem where the mean-field corresponds to the distribution of ride requests at every node in the environment.

Similar to prior approaches (Lowalekar, Varakantham, and Jaillet 2019; Shah, Lowalekar, and Varakantham 2020), we use the service rate (total percentage of requests served) as the comparison metric. We train NeurADP and DMFG-QL using a set of eight consecutive days of training data and test all the performances in a previously unseen test set of six days. The test results are reported as an average (per day) of performances in the test set pertaining to three different hyperparameters. The first is the capacity of the vehicle (c) varied from 8 to 12, the second is the maximum allowed waiting time (τ) varied from 520 seconds to 640 seconds, and the last is the number of vehicles (N), varied from 80 to 120. The results in Figs. 6(a-c) show that DMFG-QL outperforms the baselines in all our test cases. The mean field estimates in DMFG-QL help predict the distribution of ride requests in the environment, based on which agents can choose ride requests strategically. If agents choose orders that lead to destinations with a high percentage of requests, they will be able to serve more requests in the future. Thus, DMFG-QL outperforms



(a) # of Vehicles (b) Maximum Pickup Delay

(c) Capacity (d) Single Day Test

Figure 6: Results for the ride-sharing experiment. For (a), (b) and (c), we start with a prototypical configuration of c=10, τ = 580, and N = 100, and then vary the different parameters. Figures (a), (b) and (c) share the same legend given in (a).

the NeurADP method (which does not maintain a mean field). We note that the service rate for all algorithms is low in this study (10% – 30%), since we are considering fewer vehicles compared to prior works (Shah, Lowalekar, and Varakantham 2020) due to the computational requirements of being decentralized. In practice our training is completely parallelizable and this is not a limitation of our approach. Also, from Fig. 6(c), an increase in the number of vehicles, increases the service rate. In Fig. 6(d) we plot the performance of the three algorithms for a single test day (24 hours — midnight to midnight). During certain times of the day (e.g., 5 am), the ride demand is low, and all approaches satisfy a large proportion of requests. However, during the other times of the day, when the demand is high, the DMFG-QL satisfies more requests than the baselines, showing its relative superiority.

8 Conclusion

In this paper, we relaxed two strong assumptions in prior work on using mean field methods in RL. We introduced the DMFG framework, where agents are not assumed to have global information and are not homogeneous. All agents learn in a decentralized fashion, which contrasts with centralized procedures in prior work. Theoretically, we proved that the DMFG will have a suitable solution concept, DMFE. Also, we proved that a Q-learning based algorithm will find the DMFE. Further, we provided a principled method to address the 'chicken-and-egg' problem in MFRL, and demonstrated performances in a variety of environments (including RMP).

For future work, we would like to extend our theoretical analysis to the function approximation setting and analyze the convergence of policy gradient algorithms. Empirically, we could consider other real-world applications like autonomous driving and problems on demand and supply optimization.

9 Acknowledgements

Resources used in preparing this research at the University of Waterloo were provided by the province of Ontario and the government of Canada through CIFAR, NRC, NSERC and companies sponsoring the Vector Institute. Part of this work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, CIFAR; Compute Canada; and NSERC.

References

Adlakha, S.; Johari, R.; and Weintraub, G. Y. 2015. Equilibria of dynamic games with many players: Existence, approximation, and market structure. *Journal of Economic Theory*, 156: 269–316.

Alonso-Mora, J.; Samaranayake, S.; Wallar, A.; Frazzoli, E.; and Rus, D. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of National Academy of Science USA*, 114(3): 462–467.

Anahtarci, B.; Kariksiz, C. D.; and Saldi, N. 2019. Value Iteration Algorithm for Mean-field Games. *arXiv preprint arXiv:1909.01758*.

Bartle, R. G. 2014. *The elements of integration and Lebesgue measure*. John Wiley & Sons.

Busoniu, L.; Babuska, R.; and De Schutter, B. 2006. Multi-agent reinforcement learning: A survey. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, 1–6. IEEE.

Cardaliaguet, P.; and Hadikhanloo, S. 2017. Learning in mean field games: the fictitious play. *ESAIM: Control, Optimisation and Calculus of Variations*, 23(2): 569–591.

Carmona, R.; Laurière, M.; and Tan, Z. 2019a. Linear-quadratic mean-field reinforcement learning: convergence of policy gradient methods. *arXiv preprint arXiv:1910.04295*.

Carmona, R.; Laurière, M.; and Tan, Z. 2019b. Model-free mean-field reinforcement learning: mean-field MDP and mean-field Q-learning. *arXiv preprint arXiv:1910.12802*.

Elie, R.; Pérolat, J.; Laurière, M.; Geist, M.; and Pietquin, O. 2020. On the Convergence of Model Free Learning in Mean Field Games. In *AAAI*, 7143–7150.

Elliott, R.; Li, X.; and Ni, Y.-H. 2013. Discrete time mean-field stochastic linear-quadratic optimal control problems. *Automatica*, 49(11): 3222–3233.

Fu, Z.; Yang, Z.; Chen, Y.; and Wang, Z. 2019. Actor-Critic Provably Finds Nash Equilibria of Linear-Quadratic Mean-Field Games. In *ICLR*.

Gomes, D. A.; Mohr, J.; and Souza, R. R. 2010. Discrete time, finite state space mean field games. *Journal de mathématiques pures et appliquées*, 93(3): 308–328.

Guo, X.; Hu, A.; Xu, R.; and Zhang, J. 2019. Learning mean-field games. In *NeurIPS*, 4966–4976.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS*, 66–83. Springer.

Hadikhanloo, S.; and Silva, F. J. 2019. Finite mean field games: fictitious play and convergence to a first order continuous mean field game. *Journal de Mathématiques Pures et Appliquées*, 132: 369–397.

Hajek, B.; and Raginsky, M. 2019. Statistical learning theory. *Lecture Notes*, 387.

He, H.; and Boyd-Graber, J. L. 2016. Opponent Modeling in Deep Reinforcement Learning. In *ICML*, volume 48, 1804–1813. JMLR.org.

Hernandez-Leal, P.; Kartal, B.; and Taylor, M. E. 2019. A survey and critique of multiagent deep reinforcement learning. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)*, 33(6): 750–797.

Hernández-Lerma, O.; and Lasserre, J. B. 2012. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media.

Hinderer, K. 1970. Decision models. In *Foundations of Non-stationary Dynamic Programming with Discrete Time Parameter*, 78–83. Springer.

Hu, J.; and Wellman, M. P. 2003. Nash Q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov): 1039–1069.

Huang, M. 2010. Large-population LQG games involving a major player: the Nash certainty equivalence principle. *SIAM Journal on Control and Optimization*, 48(5): 3318–3353.

Huang, M.; Caines, P. E.; and Malhamé, R. P. 2003. Individual and mass behaviour in large population stochastic wireless power control problems: centralized and Nash equilibrium solutions. In *42nd IEEE International Conference on Decision and Control*. IEEE.

Huang, M.; Malhamé, R. P.; Caines, P. E.; et al. 2006. Large population stochastic dynamic games: closed-loop McKean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems*, 6(3): 221–252.

Kakutani, S. 1941. A generalization of Brouwer's fixed point theorem. *Duke mathematical journal*, 8(3): 457–459.

Kizilkale, A. C.; and Caines, P. E. 2013. Mean Field Stochastic Adaptive Control. *IEEE Trans. Autom. Control.*, 58(4): 905–920.

Konda, V. R.; and Tsitsiklis, J. N. 1999. Actor-Critic Algorithms. In *NeurIPS*. The MIT Press.

Lasry, J.-M.; and Lions, P.-L. 2007. Mean field games. *Japanese journal of mathematics*, 2(1): 229–260.

Lattimore, T.; and Szepesvári, C. 2020. *Bandit algorithms*. Cambridge University Press.

Li, M.; Qin, Z.; Jiao, Y.; Yang, Y.; Wang, J.; Wang, C.; Wu, G.; and Ye, J. 2019. Efficient Ridesharing Order Dispatching with Mean Field Multi-Agent Reinforcement Learning. In *The World Wide Web Conference, WWW 2019*. ACM.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *ICLR*.

Lowalekar, M.; Varakantham, P.; and Jaillet, P. 2019. ZAC: A Zone Path Construction Approach for Effective Real-Time Ridesharing. In *ICAPS*, 528–538. AAAI Press.

Mguni, D.; Jennings, J.; and de Cote, E. M. 2018. Decentralised Learning in Systems With Many, Many Strategic Agents. In *AAAI*, 4686–4693. AAAI Press.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529.

Nash, J. F. 1951. *Non-Cooperative Games*. Princeton University Press.

Neumann, J. v. 1928. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1): 295–320.

NYYellowTaxi. 2016. New York yellow taxi dataset. http://www.nyc.gov/html/tlc/html/about/triprecorddata.shtml. [Online; accessed 19-June-2021].

Perrin, S.; Laurière, M.; Pérolat, J.; Geist, M.; Élie, R.; and Pietquin, O. 2021. Mean Field Games Flock! The Reinforcement Learning Way. In *IJCAI*, 356–362.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley. ISBN 978-0-47161977-2.

Ropke, S.; and Cordeau, J. 2009. Branch and Cut and Price for the Pickup and Delivery Problem with Time Windows. *Transportation Science*, 43(3): 267–286.

Saldi, N.; Basar, T.; and Raginsky, M. 2018. Discrete-time risk-sensitive mean-field games. *arXiv preprint arXiv:1808.03929*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. In *arXiv preprint arXiv:1707.06347*.

Shah, S.; Lowalekar, M.; and Varakantham, P. 2020. Neural Approximate Dynamic Programming for On-Demand Ride-Pooling. In *AAAI*, 507–515. AAAI Press.

Shoham, Y.; Powers, R.; and Grenager, T. 2003. Multi-agent reinforcement learning: a critical survey. Technical report, Technical report, Stanford University.

Shukla, S.; Balasubramanian, S.; and Pavlović, M. 2016. A generalized Banach fixed point theorem. *Bulletin of the Malaysian Mathematical Sciences Society*, 39(4): 1529–1539.

Stanley, H. E. 1971. Phase transitions and critical phenomena. Clarendon.

Subramanian, J.; and Mahajan, A. 2019. Reinforcement Learning in Stationary Mean-field Games. In *AAMAS*, 251–259. IFAAMAS.

Subramanian, S. G. 2021. Decentralized Mean Field Games. https://github.com/Sriram94/DMFG.

Subramanian, S. G.; Poupart, P.; Taylor, M. E.; and Hegde, N. 2020. Multi Type Mean Field Reinforcement Learning. In *AAMAS*. IFAAMAS.

Subramanian, S. G.; Taylor, M. E.; Crowley, M.; and Poupart, P. 2021. Partially Observable Mean Field Reinforcement Learning. In *AAMAS*. ACM.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NIPS*, 1057–1063. The MIT Press.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*.

Tanaka, K.; Hori, T.; and Wang, H. O. 2003. A multiple Lyapunov function approach to stabilization of fuzzy control systems. *IEEE Transactions on fuzzy systems*, 11(4): 582–589.

Terry, J. K.; Black, B.; Jayakumar, M.; Hari, A.; Santos, L.; Dieffendahl, C.; Williams, N. L.; Lokesh, Y.; Sullivan, R.; Horsch, C.; and Ravi, P. 2020. PettingZoo: Gym for Multi-Agent Reinforcement Learning. In *arXiv preprint arXiv:2009.14471*.

Wang, Z.; Qin, Z. T.; Tang, X.; Ye, J.; and Zhu, H. 2018. Deep Reinforcement Learning with Knowledge Transfer for Online Rides Order Dispatching. In *ICDM*, 617–626. IEEE Computer Society.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine Learning*, 8(3-4): 279–292.

Więcek, P. 2020. Discrete-time ergodic mean-field games with average reward on compact spaces. *Dynamic Games and Applications*, 10(1): 222–256.

Więcek, P.; and Altman, E. 2015. Stationary anonymous sequential games with undiscounted rewards. *Journal of optimization theory and applications*, 166(2): 686–710.

Xu, Z.; Li, Z.; Guan, Q.; Zhang, D.; Li, Q.; Nan, J.; Liu, C.; Bian, W.; and Ye, J. 2018. Large-Scale Order Dispatch in On-Demand Ride-Hailing Platforms: A Learning and Planning Approach. In *KDD*. ACM.

Yang, J.; Ye, X.; Trivedi, R.; Xu, H.; and Zha, H. 2017. Deep Mean Field Games for Learning Optimal Behavior Policy of Large Populations. *CoRR*, abs/1711.03156.

Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; and Wang, J. 2018. Mean Field Multi-Agent Reinforcement Learning. In *ICML*. PMLR.

Yang, Y.; and Wang, J. 2020. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. *CoRR*, abs/2011.00583.

Yin, H.; Mehta, P. G.; Meyn, S. P.; and Shanbhag, U. V. 2014. Learning in Mean-Field Games. *IEEE Trans. Autom. Control.*, 59(3): 629–644.

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 *AAAI*.

A Proof of Theorem 1

For the proof, we will follow the idea of using discounted occupancy measures, as discussed in Puterman (1994). Let us consider an agent $j \in \{1, \ldots, N\}$. Let $\boldsymbol{\mu} \in \mathcal{M}$ and $\pi^j \in \Pi^j$ be arbitrary. Then from the Ionescu-Tulcea theorem (see Theorem 3.3 in Lattimore and Szepesvári (2020)), a mean field distribution $\boldsymbol{\mu}$ on $\mathcal{S}$ and a policy π^j defines a unique probability measure $\mathcal{P}^{\pi^j}$ on $\mathcal{S} \times \mathcal{A}^j$.

We will begin with a definition of a discounted occupancy measure for some mean field $\boldsymbol{\mu}$. Consider an arbitrary policy $\pi^j \in \Pi^j$. Then, the discounted occupancy measure $\nu^{\pi^j}_{\boldsymbol{\mu}} \in \mathcal{F}(\mathcal{S} \times \mathcal{A})$ induced by this policy on the set $\mathcal{S} \times \mathcal{A}^j$ with mean field $\boldsymbol{\mu}$ is given by the equation

$$\nu^{\pi^j}(s^j, a^j|\mu) = \sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\pi^j}(s^j_t = s^j, a^j_t = a^j|\mu_t = \mu). \tag{16}$$

The discounted occupancy measure assigns a measure to the given state-action pair, which is the infinite discounted sum of the process that has a mean field of $\boldsymbol{\mu}$ and follows the policy π^j, with the agent j hitting at various times the state s^j and taking the action a^j at that state.

The discounted measure has a property that the value function can be represented as a product of the immediate reward and the discounted occupancy measure. To prove this, consider (from Eq. 5),

$$\begin{aligned}
&J^j_{\boldsymbol{\mu}}(\pi^j) = \mathbb{E}^{\pi^j}[\textstyle\sum_{t=0}^{\infty} \beta^t r^j(s^j_t, a^j_t, \mu_t)] \\
&= \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} \textstyle\sum_{t=0}^{\infty} \beta^t \mathbb{E}^{\pi^j}[r^j(s^j_t, a^j_t, \mu_t) \\
&\qquad\qquad \mathcal{I}(s^j = s^j_t, a^j = a^j_t, \mu_t = \mu)] \\
&\overset{1}{=} \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} \textstyle\sum_{t=0}^{\infty} \beta^t \mathbb{E}^{\pi^j}[r^j(s^j, a^j, \mu) \\
&\qquad\qquad \mathcal{I}((s^j_t = s^j, a^j_t = a^j, \mu_t = \mu)] \\
&\overset{2}{=} \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} r^j(s^j, a^j, \mu) \\
&\qquad \textstyle\sum_{t=0}^{\infty} \beta^t \mathbb{E}^{\pi^j}[\mathcal{I}(s^j_t = s^j, a^j_t = a^j, \mu_t = \mu)] \\
&= \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} r^j(s^j, a^j, \mu) \\
&\qquad \textstyle\sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\pi^j}(s^j_t = s^j, a^j_t = a^j|\mu_t = \mu) \\
&= \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} r^j(s^j, a^j, \mu) \nu^{\pi^j}(s^j, a^j|\mu)
\end{aligned} \tag{17}$$

Here the notation $\mathcal{I}(x)$ is an indicator function, which equals 1 if the x is true and 0 if x is false. The sum of all indicators for all the state-action pairs at a given t equals 1. The step (1) uses this property of the indicator function, to drop the time indices of the reward function. The step (2) is from Lebesgue's dominated convergence theorem (Bartle 2014).

Now we state a lemma needed for our proof.

Lemma 1. *For an agent $j \in \{1, \ldots, N\}$ and policy π^j, given mean field $\boldsymbol{\mu}$, there exists a Markov policy $\hat{\pi}^j \in \Pi^j_M$ such that*

$$\nu^{\hat{\pi}^j}(s^j, a^j|\mu) = \nu^{\pi^j}(s^j, a^j|\mu) \tag{18}$$

Proof. For a mean field $\boldsymbol{\mu}$, we define an occupancy measure over the state space $\tilde{\nu}^{\pi^j}(s^j|\mu) \triangleq \int_{a^j} \nu^{\pi^j}(s^j, a^j|\mu)$.

Now consider a Markov policy $\hat{\pi}^j$ as follows,

$$\begin{aligned}
&\hat{\pi}^j(a^j|s^j, \mu) = \tfrac{\nu^{\pi^j}(s^j, a^j|\mu)}{\tilde{\nu}^{\pi^j}(s^j|\mu)}, \quad \text{if } \tilde{\nu}^{\pi^j}(s^j, \mu) \neq 0 \\
&\text{and} \\
&\hat{\pi}^j(a^j|s^j, \mu) = \pi^j_0(a^j), \quad \text{if } \tilde{\nu}^{\pi^j}(s^j, \mu) = 0
\end{aligned} \tag{19}$$

Here, $\pi^j_0(a^j) \in \Pi^j_M$ is an arbitrary policy.
Consider,

$$\begin{aligned}
&\tilde{\nu}^{\hat{\pi}^j}(s^j|\mu) = \textstyle\int_{a^j} \nu^{\hat{\pi}^j}(s^j, a^j|\mu) \\
&= \textstyle\int_{a^j} \sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\hat{\pi}^j}(s^j_t = s^j, a^j_t = a^j|\mu_t = \mu) \\
&\overset{1}{=} \textstyle\sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\hat{\pi}^j}(s^j_t = s^j|\mu_t = \mu) \\
&= \nu_0(s^j) + \textstyle\sum_{t=1}^{\infty} \beta^t \mathcal{P}^{\hat{\pi}^j}(s^j_t = s^j|\mu_t = \mu) \\
&= \nu_0(s^j) + \beta \textstyle\sum_{t=1}^{\infty} \beta^{t-1} \int_{s'^j \in \mathcal{S}} \int_{a'^j \in \mathcal{A}^j} \\
&\mathcal{P}^{\hat{\pi}^j}[s^j_{t-1} = s'^j, a^j_{t-1} = a'^j|\mu_{t-1} = \mu'] p(s^j|s'^j, a'^j, \mu') \\
&= \nu_0(s^j) + \beta \textstyle\sum_{t=1}^{\infty} \beta^{t-1} \int_{s'^j \in \mathcal{S}} \\
&\qquad \mathcal{P}^{\hat{\pi}^j}[s_{t-1} = s'|\mu_{t-1} = \mu'] \\
&\qquad \int_{a'^j \in \mathcal{A}^j} \hat{\pi}^j(a'^j|s'^j, \mu') p(s^j|s'^j, a'^j, \mu') \\
&= \nu_0(s^j) \\
&+ \beta \int_{s'^j \in \mathcal{S}} \textstyle\sum_{t=1}^{\infty} \beta^{t-1} \mathcal{P}^{\hat{\pi}^j}[s^j_{t-1} = s'^j|\mu_{t-1} = \mu'] \\
&\int_{a'^j \in \mathcal{A}^j} \hat{\pi}^j(a'^j|s'^j, \mu') p(s^j|s'^j, a'^j, \mu_{t-1} = \mu') \\
&\overset{2}{=} \nu_0(s^j) + \beta \int_{s'^j \in \mathcal{S}} \tilde{\nu}^{\hat{\pi}^j}(s'^j|\mu') \\
&\qquad \int_{a'^j \in \mathcal{A}^j} \hat{\pi}^j(a'^j|s'^j, \mu') p(s^j|s'^j, a'^j, \mu')
\end{aligned} \tag{20}$$

Here $\nu_0(s^j)$ is the initial distribution of the state of the agent j. To obtain expression (2), see that this follows from expression (1).

Now consider,

$$\tilde{\nu}^{\pi^j}(s^j|\mu) = \sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\pi^j}(s_t^j = s^j|\mu_t = \mu)$$

$$= \nu_0(s^j) + \sum_{t=1}^{\infty} \beta^t \mathcal{P}^{\pi^j}(s_t^j = s^j|\mu_t = \mu)$$

$$= \nu_0(s^j) + \beta \sum_{t=1}^{\infty} \beta^{t-1} \int_{s'^j \in \mathcal{S}} \int_{a'^j \in \mathcal{A}^j}$$

$$\mathcal{P}^{\pi^j}[s_{t-1}^j = s'^j, a_{t-1}^j = a'^j|\mu_{t-1} = \mu']p(s^j|s'^j, a'^j, \mu')$$

$$= \nu_0(s^j) + \beta \int_{s'^j \in \mathcal{S}} \int_{a'^j \in \mathcal{A}^j} \sum_{t=1}^{\infty} \beta^{t-1}$$

$$\mathcal{P}^{\pi^j}[s_{t-1}^j = s'^j, a_{t-1}^j = a'^j|\mu_{t-1} = \mu']p(s^j|s'^j, a'^j, \mu')$$

$$\stackrel{1}{=} \nu_0(s^j)$$

$$+\beta \int_{s'^j \in \mathcal{S}} \int_{a'^j \in \mathcal{A}^j} \nu^{\pi^j}(s'^j, a'^j|\mu')p(s^j|s'^j, a'^j, \mu')$$

$$\stackrel{2}{=} \nu_0(s^j) + \beta \int_{s'^j \in \mathcal{S}} \int_{a'^j \in \mathcal{A}^j}$$

$$\hat{\pi}^j(a'^j|s'^j, \mu')\tilde{\nu}^{\pi^j}(s'^j|\mu')p(s^j|s'^j, a'^j, \mu')$$

$$= \nu_0(s^j) + \beta \int_{s'^j \in \mathcal{S}} \tilde{\nu}^{\pi^j}(s'^j|\mu')$$

$$\int_{a'^j \in \mathcal{A}^j} \hat{\pi}^j(a'^j|s'^j, \mu')p(s^j|s'^j, a'^j, \mu') \tag{21}$$

The (1) is from Eq. 16 and (2) is from Eq. 19. From both Eq. 21 and Eq. 20, we find that both the discounted state occupation frequency can be recursively expressed as a term depending on the state occupation frequency at the previous time step and a few other terms that are the same for both the Eq. 20 and Eq. 21. Since the initial state distribution is the same for both policies, we can conclude that $\tilde{\nu}^{\hat{\pi}^j}(s^j|\mu) = \tilde{\nu}^{\pi^j}(s^j|\mu)$.

Now, consider

$$\nu^{\hat{\pi}^j}(s^j, a^j|\mu) = \sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\hat{\pi}^j}(s_t^j = s^j, a_t^j = a^j|\mu_t = \mu)$$

$$= \sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\hat{\pi}^j}(a_t^j = a^j|s_t^j = s^j, \mu_t = \mu)$$

$$\mathcal{P}^{\hat{\pi}^j}(s_t^j = s^j|\mu_t = \mu)$$

$$= \sum_{t=0}^{\infty} \beta^t \hat{\pi}^j(a_t^j = a^j|s_t^j = s^j, \mu_t = \mu)$$

$$\mathcal{P}^{\hat{\pi}^j}(s_t^j = s^j|\mu_t = \mu) \tag{22}$$

Also, from Eq. 19, we have

$$\begin{aligned}\nu^{\pi^j}(s^j, a^j|\mu) &= \hat{\pi}^j(a^j|s^j, \mu) \times \tilde{\nu}^{\pi^j}(s^j|\mu) \\ &= \hat{\pi}^j(a^j|s^j, \mu) \times \tilde{\nu}^{\hat{\pi}^j}(s^j|\mu) \\ &= \hat{\pi}^j(a^j|s^j, \mu) \sum_{t=0}^{\infty} \beta^t \mathcal{P}^{\hat{\pi}^j}(s_t^j = s^j|\mu_t = \mu) \\ &= \sum_{t=0}^{\infty} \beta^t \hat{\pi}^j(a^j|s^j, \mu)\mathcal{P}^{\hat{\pi}^j}(s_t^j = s^j|\mu_t = \mu)\end{aligned} \tag{23}$$

From Eq. 22 and Eq. 23 we find that $\nu^{\hat{\pi}^j}(s^j, a^j|\mu) = \nu^{\pi^j}(s^j, a^j|\mu)$, due to the property of the Markov policy $\hat{\pi}^j$. □

Theorem 1. *For any mean field, $\boldsymbol{\mu} \in \mathcal{M}$, and an agent $j \in \{1, \ldots, N\}$, we have,*

$$\sup_{\pi^j \in \Pi^j} J_{\boldsymbol{\mu}}^j(\pi^j) = \sup_{\pi^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\pi^j).$$

Proof. Now, we aim to show that $\sup_{\pi^j \in \Pi^j} J_{\boldsymbol{\mu}}^j(\pi^j) = \sup_{\pi^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\pi^j)$. Since $\Pi_M^j \subset \Pi^j$, we know that $\sup_{\pi^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\pi^j) \leq \sup_{\pi^j \in \Pi^j} J_{\boldsymbol{\mu}}^j(\pi^j)$. To show the equality we aim to prove $\sup_{\pi^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\pi^j) \geq \sup_{\pi^j \in \Pi^j} J_{\boldsymbol{\mu}}^j(\pi^j)$ as well.

To show this result we use Lemma 1. Consider a policy π^j and a Markov policy $\hat{\pi}^j$, such that $\nu_{\boldsymbol{\mu}}^{\hat{\pi}^j} = \nu_{\boldsymbol{\mu}}^{\pi^j}$. Now using the Eq. 17, we have

$$\begin{aligned}J_{\boldsymbol{\mu}}^j(\pi^j) &= \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} r^j(s^j, a^j, \mu)\nu^{\pi^j}(s^j, a^j|\mu) \\ &\stackrel{1}{=} \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} r^j(s^j, a^j, \mu)\nu^{\hat{\pi}^j}(s^j, a^j|\mu) \\ &\leq \sup_{\hat{\pi}^j \in \Pi_M^j} \int_{(s^j \in \mathcal{S}, a^j \in \mathcal{A}^j)} r^j(s_t^j, a_t^j, \mu_t)\nu^{\hat{\pi}^j}(s^j, a^j|\mu) \\ &= \sup_{\hat{\pi}^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\hat{\pi}^j)\end{aligned} \tag{24}$$

Here (1) is from Lemma 1. Now, from Eq. 24, and taking supermum on both sides, we get that $\sup_{\pi^j \in \Pi^j} J_{\boldsymbol{\mu}}^j(\pi^j) \leq \sup_{\pi^j \in \Pi_M^j} J_{\boldsymbol{\mu}}^j(\pi^j)$, which concludes our proof.

□

B Proof of Theorem 2

The proof of this theorem follows the Theorem 1 in Saldi, Basar, and Raginsky (2018). We aim to extend it to the decentralized setting as mentioned in Section 5.

Before starting the proof, we will state some assumptions,

Assumption 2. *The reward function for all agents $j \in \{1, \dots, N\}$ is bounded and continuous.*

Let us define a function $w : \mathcal{S} \to [1, \infty)$, such that there exists an increasing sequence of compact subsets $\{K_n\}_{n\geq 1}$ of $\mathcal{S}$ where

$$\lim_{n\to\infty} \inf_{s\in\mathcal{S}\setminus K_n} w(s) = \infty. \tag{25}$$

That is, the value of $w(s)$ gets close to infinity in this limit. Here, the w can be regarded as a continuous moment function (Saldi, Basar, and Raginsky 2018).

Assumption 3. *There exists a positive α such that the following holds (for all agents $j \in \{1, \dots, N\}$),*

$$\sup_{(a^j,\mu^j)\in\mathcal{A}^j\times P(\mathcal{S})} \int_{\mathcal{S}} w(s^j_{t+1}) p(s^j_{t+1}|s^j_t, a^j_t, \mu_t) \leq \alpha w(s^j_t) \tag{26}$$

Also, consider two different moment functions w and v, then α satisfies,

$$\sup_{(a^j,\mu^j)\in\mathcal{A}^j\times P(\mathcal{S})} \Big| \int_{\mathcal{S}} w(s^j_{t+1}) p(s^j_{t+1}|s^j_t, a^j_t, \mu_t) - \int_{\mathcal{S}} v(s^j_{t+1}) p(s^j_{t+1}|s^j_t, a^j_t, \mu_t)) \Big| \leq \alpha |w(s^j_t) - v(s^j_t)| \tag{27}$$

Assumption 4. *The initial mean field μ_0 satisfies*

$$\int_{\mathcal{S}} w(s^j)\mu_0(s^j) \triangleq M < \infty \tag{28}$$

Assumption 5. *There exist a $\gamma \geq 1$ and a positive scalar $\mathcal{R}$ such that for all $t \geq 0$, we define $M_t \triangleq \gamma^t \mathcal{R}$, then*

$$\sup_{(a^j,\mu)\in\mathcal{A}^j\times P(\mathcal{S})} r^j(s^j_t, a^j_t, \mu_t) \leq M_t w(s^j_t) \tag{29}$$

Now, we will state a few definitions required for our proof. For any function $g : \mathcal{S} \to \Re$, we define the w-norm as:

$$||g||_w \triangleq \sup_{s\in\mathcal{S}} \frac{|g(s)|}{w(s)} \tag{30}$$

Let $B_w(\mathcal{S})$ be the Banach space of all real valued measurable functions on $\mathcal{S}$ with a finite w-norm.

Also, for any signed measure μ in the space of $\mathcal{S}$, the w-norm can be defined as

$$||\mu||_w \triangleq \sup_{g\in B_w(\mathcal{S}):||g||_w\leq 1} \Big| \int_{\mathcal{S}} g(s)\mu(s) \Big| \tag{31}$$

Further, we define another set

$$\mathcal{T}_w(\mathcal{S}) \triangleq \{\mu \in \mathcal{P}(\mathcal{S}) : ||\mu||_w < \infty\} \tag{32}$$

Also, for $t \geq 0$, we define

$$\mathcal{T}^t_w(\mathcal{S}) \triangleq \Big\{\mu \in \mathcal{T}_w(\mathcal{S}) : \int_{\mathcal{S}} w(s)\mu(s) \leq \alpha^t M\Big\} \tag{33}$$

where the constant M is obtained from Assumption 4. Like our notation $\mathcal{P}(\mathcal{S})$, we are using the notation $\mathcal{P}(\mathcal{S}\times\mathcal{A}^j)$ to denote the probability of a state-action pair. Let us consider an element $\nu \in \mathcal{P}(\mathcal{S}\times\mathcal{A}^j)$, and use the notation $\nu_1 \triangleq \nu(\cdot\times\mathcal{A}^j)$ to denote the state marginal of ν. Now we define a new set,

$$\mathcal{T}^t_w(\mathcal{S}\times\mathcal{A}^j) \triangleq \Big\{\nu \in \mathcal{P}(\mathcal{S}\times\mathcal{A}^j) : \nu_1 \in \mathcal{T}^t_w(\mathcal{S})\Big\}. \tag{34}$$

For each $t \geq 0$, we will define

$$L_t \triangleq \sum_{k=t}^{\infty} (\beta\alpha)^{k-t} M_k \tag{35}$$

Here the constants α and M_k are obtained from Assumption 3 and Assumption 5 respectively.

It follows that the following equation holds,

$$L_t = M_t + (\beta\alpha)L_{t+1} \tag{36}$$

Let $C_w(S)$ denote the Banach space of all real valued bounded measurable functions on $\mathcal{S}$ with a finite w-norm.

Let us define a set

$$C^t_w(\mathcal{S}) \triangleq \{u \in C_w(\mathcal{S}) : ||u||_w \leq L_t\} \tag{37}$$

For an agent $j \in \{1, \dots, N\}$, let us consider an operator

$$T^{\boldsymbol{\mu}}_t u^j(s_t) = \max_{a^j_t\in\mathcal{A}^j} \Big[r(s^j_t, a^j_t, \mu_t) + \beta \int_{\mathcal{S}} u^j(s^j_{t+1}) p(s^j_{t+1}|s^j_t, a^j_t, \mu_t)\Big] \tag{38}$$

where $u^j : \mathcal{S} \to \Re$

We will first prove the following lemma.

Lemma 2. *For all $t \geq 0$ and a given mean field $\boldsymbol{\mu}$, the operator $T^{\boldsymbol{\mu}}_t$ maps $C^{t+1}_w(\mathcal{S})$ into $C^t_w(\mathcal{S})$. Also, this operator will satisfy*

$$||T^{\boldsymbol{\mu}}_t u - T^{\boldsymbol{\mu}}_t x||_w \leq \alpha\beta ||u - x||_w \tag{39}$$

for any $u, x \in C_w(\mathcal{S})$.

Proof. Let $u \in C^{t+1}_w(\mathcal{S})$. We have the following relation,

$$\begin{aligned} ||T^{\boldsymbol{\mu}}_t u||_w &\leq \sup_{(s^j_t,a^j_t)\in\mathcal{S}\times\mathcal{A}^j} \frac{\Big|r^j(s^j_t,a^j_t,\mu_t) + \beta\int_S u(s^j_{t+1}) p(s^j_{t+1}|s^j_t,a^j_t,\mu_t)\Big|}{w(s^j_t)} \\ &\leq \sup_{(s^j_t,a^j_t)\in\mathcal{S}\times\mathcal{A}^j} \frac{M_t w(s^j_t) + \beta\alpha L_{t+1} w(s^j_t)}{w(s^j_t)} \\ &= M_t + \beta\alpha L_{t+1} \\ &= L_t \end{aligned} \tag{40}$$

The second last step is from Assumption 5 and Assumption 3. Also, we use the bound on the w-norm from Eq. 37. The last step is from Eq. 36. This proves the first statement.

To prove the second statement, without loss of generality, let us assume that $u_0 \geq x_0$. Now consider,

$$
\begin{aligned}
&||T_t^{\boldsymbol{\mu}}u - T_t^{\boldsymbol{\mu}}x||_w = \\
&\sup_{(s_t^j)\in\mathcal{S}} \frac{\max_{a_t^j}\Big|r^j(s_t^j,a_t^j,\mu_t)+\beta\int_{\mathcal{S}}u(s_{t+1}^j)p(s_{t+1}^j|s_t^j,a_t^j,\mu_t)}{w(s_t^j)} \\
&\qquad\qquad \frac{-r^j(s_t^j,a_t^j,\mu_t)-\beta\int_{\mathcal{S}}x(s_{t+1}^j)p(s_{t+1}^j|s_t^j,a_t^j,\mu_t)\Big|}{w(s_t^j)} \\
&= \sup_{(s_t^j)\in\mathcal{S}} \frac{\max_{a_t^j}\Big|\beta\Big(\int_{\mathcal{S}}u(s_{t+1}^j)p(s_{t+1}^j|s_t^j,a_t^j,\mu_t)}{w(s_t^j)} \\
&\qquad\qquad \frac{-\int_{\mathcal{S}}x(s_{t+1}^j)p(s_{t+1}^j|s_t^j,a_t^j,\mu_t)\Big)\Big|}{w(s_t^j)} \\
&= \sup_{(s_t^j)\in\mathcal{S}} \frac{\max_{a_t^j}\Big|\beta\int_{\mathcal{S}}p(s_{t+1}^j|s_t^j,a_t^j,\mu_t)(u(s_{t+1}^j)-x(s_{t+1}^j))\Big|}{w(s_t^j)} \\
&\leq \sup_{(s_t^j)\in\mathcal{S}} \frac{\alpha\beta\Big|(u(s_t^j)-x(s_t^j))\Big|}{w(s_t^j)} \\
&= \alpha\beta||u-x||_w
\end{aligned} \tag{41}
$$

We apply Assumption 3 and the last step is from Eq. 30. This proves the second part of the lemma as well.

□

Let us define a new set $\mathcal{C}$ from $C_w^t(\mathcal{S})$ as follows:

$$\mathcal{C} \triangleq \Pi_{t=0}^{\infty} C_w^t(\mathcal{S}) \tag{42}$$

Also, let us assign the following metric to $\mathcal{C}$:

$$\rho(\boldsymbol{u},\boldsymbol{v}) \triangleq \sum_{t=0}^{\infty}\sigma^{-t}||u_t - v_t||_w \tag{43}$$

where $\sigma > 0$ and the following assumption holds.

Assumption 6. *The variables σ, α, and β satisfy $\alpha\sigma\beta < 1$.*

This assumption guarantees that the metric $\mathcal{C}$ is complete w.r.t ρ, and $\rho(\boldsymbol{u},\boldsymbol{v}) < \infty$ for all $\boldsymbol{u},\boldsymbol{v} \in \mathcal{C}$.

Using the operators $\{T_t^{\boldsymbol{\mu}}\}_{t\geq 0}$, let us define a new operator $T^{\boldsymbol{\mu}} : \mathcal{C} \to \mathcal{C}$ as follows (for all $t \geq 0$).

$$(T^{\boldsymbol{\mu}}\boldsymbol{u})_t = T_t^{\boldsymbol{\mu}}u_{t+1} \tag{44}$$

From Lemma 2 we know that $T^{\boldsymbol{\mu}}$ is a well-defined operator that maps $\mathcal{C}$ to itself. Also, from Eq. 39, $T^{\boldsymbol{\mu}}$ is a contraction operator on $\mathcal{C}$ with constant of contraction $\alpha\beta\sigma < 1$ (from Assumption 6). Now, $T^{\boldsymbol{\mu}}$ will have a unique fixed point by the Banach fixed point theorem in $\mathcal{C}$.

Now, we move to proving another lemma. Consider a mean field $\boldsymbol{\mu}$, the optimal value function for an agent j can be given by

$$J_{*,t}^{j,\boldsymbol{\mu}}(s^j) = \sup_{\pi^j\in\Pi^j}\mathbb{E}^{\pi^j}\Big[\sum_{t=0}^{\infty}\beta^t r^j(s_t^j,a_t^j,\mu_t)|s_0^j = s^j\Big]. \tag{45}$$

Let $\boldsymbol{J}_*^{j,\boldsymbol{\mu}} \triangleq (J_{*,t}^{j,\boldsymbol{\mu}})_{t\geq 0}$ denote the optimal value function for an agent $j \in \{1,\ldots,N\}$. This function is guaranteed to be continuous by Assumption 2.

Lemma 3. *For any $\boldsymbol{\mu}$, the optimal point $\boldsymbol{J}_*^{j,\boldsymbol{\mu}}$, for an agent $j \in \{1,\ldots,N\}$, belongs to the Banach space $\mathcal{C}$.*

Proof. Let π be a Markov policy. At any time $t \geq 0$, we have

$$
\begin{aligned}
&J_{*,t}^{j,\boldsymbol{\mu}}(s) \\
&= \textstyle\sum_{k=t}^{\infty}\beta^{k-t}\mathbb{E}^{\pi}(r^j(s_k^j,a_k^j,\mu_k)|s_t^j = s^j) \\
&\leq \textstyle\sum_{k=t}^{\infty}\beta^{k-t}M_k\mathbb{E}^{\pi}(v(s_k)|s_t^j = s^j) \\
&\leq \textstyle\sum_{k=t}^{\infty}\beta^{k-t}M_k\alpha^{k-t}v(s^j) \\
&= L_t v(s^j)
\end{aligned} \tag{46}
$$

The second step is from Assumption 5. The third step is from Assumption 3. Hence, the function $\boldsymbol{J}_{*,t}^{j,\boldsymbol{\mu}}$ belongs to the set $C_w^t(\mathcal{S})$ from the Eq. 37.

□

Let us define another set $\mathcal{D}$ as

$$\mathcal{D} \triangleq \Pi_{t=0}^{\infty}\mathcal{T}_w^t(\mathcal{S}\times\mathcal{A}^j) \tag{47}$$

Before proving further results, we restate a result proved previously in non-homogeneous stochastic processes (Hinderer 1970). We will adapt the result to pertain to an agent $j \in \{1,\ldots,N\}$. For any E-valued random element x, we use the notation $\mathcal{L}(x)$ to denote the distribution of x (i.e. $\mathcal{L}(x) \in P(E)$).

Consider a variable $\boldsymbol{\nu} \in \mathcal{D}$. From the operator $T^{\boldsymbol{\mu}}$, we will define another operator $T^{\boldsymbol{\nu}}$, where the relation is such that $\mu_t = \nu_{t,1}$. Recall that the subscript here refers to the state marginal of ν (refer Eq. 34). From the result of $T^{\boldsymbol{\mu}}$, we know that the operator $T^{\boldsymbol{\nu}}$ is well-defined and has a unique fixed point. Now, we can state the following result.

Lemma 4. *For an agent $j \in \{1,\ldots,N\}$, for any $\boldsymbol{\nu} \in \mathcal{D}$, the collection of value functions $\boldsymbol{J}_*^{j,\boldsymbol{\nu}}$ is the unique fixed point of the operator $T^{\boldsymbol{\nu}}$. Furthermore, $\pi^j \in M$ is optimal if and only if*

$$
\begin{aligned}
&\nu_t^{\pi^j}(\{(s_t^j,a_t^j) : r^j(s_t^j,a_t^j,\nu_{t,1}) + \\
&\beta\int_{\mathcal{S}}J_{*,t+1}^{j,\boldsymbol{\nu}}(s_{t+1}^j)p(s_{t+1}^j|s_t^j,a_t^j,\nu_{t,1}) = T_t^{\nu}J_{*,t+1}^{j,\boldsymbol{\nu}}(s_t^j)\}) \\
&= 1
\end{aligned} \tag{48}
$$

where $\nu_t^{\pi^j} = \mathcal{L}(s_t^j,a_t^j)$.

Proof. See Hinderer (1970).

□

Now, let us define a set-valued mapping $\tau : \mathcal{D} \to 2^{|\mathcal{P}(\mathcal{S}\times\mathcal{A})^{\infty}|}$, for an agent j as follows:

$$\tau(\boldsymbol{\nu}) = C(\boldsymbol{\nu}) \cap B(\boldsymbol{\nu}) \tag{49}$$

where

$$
\begin{aligned}
&C(\boldsymbol{\nu}) \triangleq \{\boldsymbol{\nu'} \in \mathcal{P}(\mathcal{S}\times\mathcal{A}^j) : \nu'_{0,1} = \mu_0 \text{ and} \\
&\nu'_{t+1,1}(\cdot) = \textstyle\int_{\mathcal{S}\times\mathcal{A}^j} p(\cdot|s^j,a^j,\nu_{t,1})\nu_t(s^j,a^j)\}
\end{aligned} \tag{50}
$$

and

$$B(\boldsymbol{\nu}) \triangleq \Big\{\boldsymbol{\nu}' \in \mathcal{P}(\mathcal{S}\times\mathcal{A}^j) : \forall t \geq 0,$$
$$\nu'_t(\{(s^j_t, a^j_t) : r^j(s^j_t, a^j_t, \nu_{t,1}) + \tag{51}$$
$$\beta \int_{\mathcal{S}} J^{j,\boldsymbol{\nu}}_{*,t+1}(s^j_{t+1}) p(s^j_{t+1}|s^j_t, a^j_t, \nu_{t,1})\} = 1\Big\}$$

Now in the next result we show that, the image of $\mathcal{D}$ under τ is contained in $2^{\mathcal{D}}$.

Proposition 1. *For any $\nu \in \mathcal{D}$, we have $\tau(\nu) \subset \mathcal{D}$*

Proof. Fix a $\nu \in \mathcal{D}$. To prove the result, it is sufficient to prove that $C(\boldsymbol{\nu}) \subset \mathcal{D}$. Let $\boldsymbol{\nu}' \in C(\boldsymbol{\nu})$. We aim to prove by induction that $\nu'_{t,1} \in P^t_v(\mathcal{S})$ for all $t \geq 0$. The claim trivially holds for $t = 0$ as $\nu'_{0,1} = \mu_0$. Assume that the claim holds for t and consider $t+1$. We have

$$\begin{aligned}
&\int_{\mathcal{S}} w(s^j_t)\nu'_{t+1,1}(s^j_{t+1}) \\
&= \int_{\mathcal{S}\times\mathcal{A}} \int_{\mathcal{S}} w(s^j_{t+1}) p(s^j_{t+1}|s^j_t, a^j_t, \nu_{t,1})\nu_t(s^j_t, a^j_t) \\
&\leq \int_{\mathcal{S}} \alpha w(s^j_t)\nu_{t,1}(s^j_t) \\
&\leq \alpha^{t+1} M
\end{aligned} \tag{52}$$

The third step is from Assumption 3 and the last step is from the fact that $\nu_{t,1} \in P^t_v(\mathcal{S})$. Hence, we can conclude that $\nu'_{t+1,1} \in P^{t+1}_v(\mathcal{S})$. □

Now we can say that $\boldsymbol{\nu} \in \mathcal{D}$ is a fixed point of τ if $\boldsymbol{\nu} \in \tau(\boldsymbol{\nu})$. The following lemma connects the mean field equilibrium and the fixed points of τ.

Lemma 5. *Suppose the set valued mapping τ has a fixed point $\boldsymbol{\nu^j} = (\nu^j_t)_{t\geq 0}$ for an agent $j \in \{1, \ldots, N\}$. Consider a Markov policy for the agent j as $\pi^j = (\pi^j_t)_{t\geq 0}$, which is obtained by factoring as $\nu^j_t(s^j_t, a^j_t) = \nu^j_{t,1}(s^j_t)\pi^j_t(a^j_t|s^j_t)$, and let $\boldsymbol{\nu}^j_1 = (\nu^j_{t,1})_{t\geq 0}$. Then the pair $(\pi^j, \boldsymbol{\nu}^j_1)$ is a decentralized mean field equilibrium.*

Proof. If $\boldsymbol{\nu}^j \in \tau(\boldsymbol{\nu}^j)$, then the corresponding Markov policy π^j satisfies the Eq. 48 for $\boldsymbol{\nu}^j$. Thus, by Lemma 4, $\pi^j \in \Phi(\boldsymbol{\nu}^j_1)$. Further, since $\boldsymbol{\nu}^j \in C(\boldsymbol{\nu}^j)$, we have $\Psi(\pi^j) = \boldsymbol{\nu}^j_1$, which completes the proof. □

From the Lemma 5, it can be seen that the set valued mapping (operator) τ having a fixed point is sufficient to guarantee the existence of the decentralized mean field equilibrium. Like several results in centralized multiagent systems (Nash 1951) and centralized mean field games (Lasry and Lions 2007; Huang et al. 2006; Saldi, Basar, and Raginsky 2018), we will use the Kakutani's fixed point theorem (Kakutani 1941), to guarantee the existence of a fixed point for the operator τ. This theorem requires the set on which the set valued mapping τ operates to be a non-empty, compact and convex subset of some Euclidean space. Further, the operator $\tau(\boldsymbol{\nu}^j)$ is required to be non-empty and convex for all $\boldsymbol{\nu}^j$. Finally, the operator τ should have a closed graph. Given these three conditions, we can conclude that τ has a fixed point using the Kakutani's fixed point theorem.

For the first condition, we need to show that the set on which $\boldsymbol{\nu}^j$ resides is non-empty, compact and convex, i.e. we need to show that $\mathcal{D}$ is non-empty, compact and convex. First, note that the function w can be expressed as a continuous moment function and hence the corresponding set $\mathcal{T}^t_v(\mathcal{S})$ is guaranteed to be compact (Hernández-Lerma and Lasserre 2012). As a consequence, the set $\mathcal{T}^t_v(\mathcal{S}\times\mathcal{A}^j)$ is tight, since the action space $\mathcal{A}^j$ is compact. Also, since the set $\mathcal{T}^t_v(\mathcal{S}\times\mathcal{A}^j)$ is closed, it is compact. Therefore, the set $\mathcal{D}$ is also compact. From Assumption 3 and Assumption 5 we can show that a line segment between any two points in $\mathcal{D}$ lies in $\mathcal{D}$ and hence the set $\mathcal{D}$ is convex. These assumptions also guarantee that the set $\mathcal{D}$ is non-empty.

For the third condition, we need to show that $\tau(\boldsymbol{\nu}^j)$ is non-empty and convex for any $\boldsymbol{\nu}^j \in \mathcal{D}$. Now, from Lemma 4 we know that $B(\boldsymbol{\nu}^j)$ is non-empty and hence $\tau(\boldsymbol{\nu}^j)$ is non-empty. Also, we can show that each of the sets $C(\boldsymbol{\nu}^j)$ and $B(\boldsymbol{\nu}^j)$ is convex (see Saldi, Basar, and Raginsky (2018)) and hence, their intersection is convex. This makes the set $\tau(\boldsymbol{\nu}^j)$ convex.

Lemma 6. *Using Assumptions 2—6, the graph of τ, i.e. the set*

$$Gr(\tau) \triangleq \{(\boldsymbol{\nu}, \boldsymbol{\mathcal{E}}) \in \mathcal{D}\times\mathcal{D} : \boldsymbol{\mathcal{E}} \in \tau(\boldsymbol{\nu})\} \tag{53}$$

is closed.

Proof. See Proposition 3.9 in Saldi, Basar, and Raginsky (2018) for complete proof. □

Now, we are ready to give the final result.

Lemma 7. *Using Assumptions 2—6, for an agent $j \in \{1, \ldots, N\}$, there exists a fixed point $\boldsymbol{\nu}^j$ of the set valued mapping $\tau : \mathcal{D} \to 2^{\mathcal{D}}$. Then, the pair $(\pi^j, \boldsymbol{\nu}^j_1)$ is a decentralized mean field equilibrium, where π^j is the policy of agent j and $\boldsymbol{\nu}^j_1$ is its mean field estimate constructed according to Lemma 5.*

Proof. Using the Lemma 6 and our previous results, we have proved that $\mathcal{D}$ is non-empty, compact and convex. Further, the set valued mapping τ has a closed graph, is non-empty and convex. Hence, by Kakutani's fixed point theorem (Kakutani 1941), τ has a fixed point. Thus, from Lemma 5 this fixed point is the decentralized mean field equilibrium. □

Theorem 2. *An agent $j \in \{1, \ldots, N\}$ in the DMFG admits a decentralized mean field equilibrium $(\pi^j_*, \mu^j_*) \in \Pi^j \times \mathcal{M}$.*

Proof. Our result follows from Lemma 7. □

C Proof of Theorem 3

In this and the subsequent theorems, we follow the theoretical results and proofs in the work by Anahtarci, Kariksiz, and Saldi (2019), and extend them to the decentralized setting.

First, we will start with some definition for the norms, similar to our previous proofs. Consider an agent $j \in \{1, \dots, N\}$. Let $w : \mathcal{S} \times \mathcal{A}^j \to \Re$ be a continuous weight function. For any measurable function $v : \mathcal{S} \times \mathcal{A}^j \to \Re$, the w-norm of v is defined as,

$$||v||_w \triangleq \sup_{s^j, a^j} \frac{|v(s^j, a^j)|}{w(s^j, a^j)}. \tag{54}$$

Also, for any measurable function $u : \mathcal{S} \to \Re$, the $w_{\max}$-norm is defined as

$$||v||_{w_{\max}} \triangleq \sup_{s^j} \frac{u(s^j)}{w_{\max}(s^j)}. \tag{55}$$

Now we will state a set of assumptions needed for our results.

Assumption 7. *For all agents $j \in \{1, \dots, N\}$, the reward function is continuous, and it satisfies the following Lipschitz bounds (for some Lipschitz constants L_1 and L_2):*

$$||r^j(\cdot, \cdot, \mu_t) - r^j(\cdot, \cdot, \hat{\mu}_t)||_w \leq L_1 W_1(\mu_t, \hat{\mu}_t), \quad \forall \mu_t, \hat{\mu}_t \tag{56}$$

where W_1 is the Wasserstein distance of order 1. Also,

$$\sup_{(a_t^j, \mu_t) \in \mathcal{A}^j \times P(\mathcal{S})} |r^j(s_t^j, a_t^j, \mu_t) - r(\hat{s}_t^j, a_t^j, \mu_t)| \leq L_2 d_X(s_t^j, \hat{s}_t^j), \quad \forall s_t^j, \hat{s}_t^j \tag{57}$$

Assumption 8. *For an agent $j \in \{1, \dots, N\}$, the transition function $p(\cdot|s^j, a^j, \mu)$ is weakly continuous in (s^j, a^j, μ) and satisfies the following Lipschitz bounds (for some Lipschitz constants K_1 and K_2):*

$$\sup_{s^j \in S} W_1(p(\cdot|s_t^j, a_t^j, \mu_t), p(\cdot|s_t^j, \hat{a}_t^j, \hat{\mu}_t)) \leq K_1(||a_t^j - \hat{a}_t^j|| + W_1(\mu_t, \hat{\mu}_t)), \quad \forall \mu_t, \hat{\mu}_t, \forall a_t^j, \hat{a}_t^j. \tag{58}$$

Also, we have,

$$\sup_{\mu \in P(S)} W_1(p(\cdot|s_t^j, a_t^j, \mu_t), p(\cdot|\hat{s}_t^j, \hat{a}_t^j, \mu_t)) \leq K_2(d_X(s_t^j, \hat{s}_t^j) + ||a_t^j - \hat{a}_t^j||), \quad \forall s_t^j, \hat{s}_t^j, \forall a_t^j, \hat{a}_t^j. \tag{59}$$

Assumption 9. *The action space $\mathcal{A}^j$ is convex for all agents $j \in \{1, \dots, N\}$.*

Assumption 10. *For all agents j and all time t, there exist non-negative real numbers M and α such that for each $(s_t^j, a_t^j, \mu) \in \mathcal{S} \times \mathcal{A}^j \times \mathcal{P}(\mathcal{S})$, we have*

$$r^j(s_t^j, a_t^j, \mu) \leq M \tag{60}$$

$$\int_S w_{\max}(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu) \leq \alpha w(s_t^j, a_t^j) \tag{61}$$

Assumption 11. *The product of constants $\beta\alpha < 1$.*

Let $C(\mathcal{S})$ denote the set of real-valued continuous functions on $\mathcal{S}$. Let $Lip(\mathcal{S})$ denote the set of all Lipshitz continuous function on $\mathcal{S}$, i.e.,

$$Lip(\mathcal{S}) \triangleq \{g \in C(\mathcal{S}) : ||g||_{Lip} < \infty\} \tag{62}$$

where $||g||_{Lip}$ is defined as

$$||g||_{Lip} \triangleq \sup_{(x,y) \in \mathcal{S} \times \mathcal{S}} \frac{|g(x) - g(y)|}{d_X(x, y)} \tag{63}$$

Here $g \in C(\mathcal{S})$. The finiteness of $||g||_{Lip}$ guarantees that g is Lipschitz continuous with constant $||g||_{Lip}$.

Also, let us define $B(\mathcal{S}, K)$ to be the set of all real valued measurable functions in $\mathcal{S}$ with $w_{\max}$-norm less than K. We use $Lip(\mathcal{S}, K)$ to denote the set of all Lipschitz continuous functions with $||g||_{Lip} < \infty$.

Finally, we define an operator $F : \mathcal{S} \times Lip(\mathcal{S}) \times \mathcal{P}(\mathcal{S}) \times \mathcal{A}^j \to \Re$ as

$$F : \mathcal{S} \times Lip(\mathcal{S}) \times \mathcal{P}(\mathcal{S}) \times \mathcal{A}^j \ni (s_t^j, v_t^j, \mu_t, a_t^j) \to r^j(s_t^j, a_t^j, \mu_t) + \beta \int_{\mathcal{S}} v_t^j(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \in \Re \tag{64}$$

Assumption 12. *Let $\mathcal{F}$ be the set of non-negative functions in*

$$Lip\Big(\mathcal{S}, \frac{L_2}{1 - \beta K_2}\Big) \cap B\Big(\mathcal{S}, \frac{M}{1 - \beta\alpha}\Big) \tag{65}$$

For an agent $j \in \{1, \dots, N\}$, for any $v \in \mathcal{F}$, $\mu \in \mathcal{P}(\mathcal{S})$, and $s^j \in S$, $F(s^j, v^j, \mu, \cdot)$ is ρ-strongly convex; that is $F(s^j, v^j, \mu, \cdot)$ is differentiable and its gradient ∇F satisfies

$$F(s_t^j, v_t^j, \mu_t, a_t^j) \geq F(s_t^j, v_t^j, \mu_t, \hat{a}_t^j) + \nabla F(s_t^j, v_t^j, \mu_t, \hat{a}_t^j)^T \cdot (a_t^j - \hat{a}_t^j) + \frac{\rho}{2} ||a_t^j - \hat{a}_t^j||^2 \tag{66}$$

for some $\rho > 0$ and for all $a_t^j, \hat{a}_t^j \in \mathcal{A}^j$.

Assumption 13. *For an agent $j \in \{1, \dots, N\}$, the gradient $\nabla F(s_t^j, v_t^j, \mu_t, a_t^j) : \mathcal{S} \times \mathcal{F} \times \mathcal{P}(\mathcal{S}) \times \mathcal{A}^j \to \Re$ satisfies the following Lipshitz bound (for some Lipschitz constant K_F):*

$$\sup_{a_t^j \in A^j} ||\nabla F(s_t^j, v_t^j, \mu_t, a_t^j) - \nabla F(\hat{s}_t^j, \hat{v}_t^j, \hat{\mu}_t, a_t^j)|| \leq K_F(d_X(s_t^j, \hat{s}_t^j) + ||v_t^j - \hat{v}_t^j||_{w_{\max}} + W_1(\mu_t^j, \hat{\mu}_t^j)) \tag{67}$$

for every $s_t^j, \hat{s}_t^j \in \mathcal{S}; v^j, \hat{v}_t^j \in \mathcal{F}, \mu_t, \hat{\mu}_t \in \mathcal{P}(\mathcal{S})$.

Assumption 12 and Assumption 13 present a constraint on the change in the value function during each of the updates. These assumptions are generally considered in literature establishing the existence of Lyapunov functions that guarantee the presence of local and global optimal points (maxima or minima) as the case may be (Tanaka, Hori, and Wang 2003).

We also need to assume the following for all the constants,

Assumption 14. *The following relation holds*

$$k \triangleq \max\{\beta\alpha + \tfrac{K_F}{\rho} K_1, \beta \tfrac{L_2}{1 - \beta K_2} + \Big(\tfrac{K_F}{\rho} + 1\Big) K_1 + K_2 + \tfrac{K_F}{\rho}\} < 1 \tag{68}$$

For the rest of the proof we will also apply Assumption 1. Consider a mean field $\boldsymbol{\mu}$, the optimal value function for an agent j is given by the Eq. 45. Using the same procedure as the result in Lemma 4, this optimal value function can be shown to be a unique fixed point of a Bellman operator $T_{\boldsymbol{\mu}}$ that is a contraction on the $w_{\max}$-norm with a constant of contraction $\beta\alpha$ (from Assumption 11). Now we can write the following equation,

$$\begin{aligned} J_*^{j,\boldsymbol{\mu}}(s_t^j) &= \max_{a^j \in \mathcal{A}^j} \Big[r^j(s_t^j, a_t^j, \mu_t) + \\ &\qquad \beta \int_{\mathcal{S}} J_*^{j,\boldsymbol{\mu}}(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \Big] \\ &\triangleq T_{\boldsymbol{\mu}} J_*^{j,\boldsymbol{\mu}}(s_t^j) \end{aligned} \tag{69}$$

Let us assume that this maximization is caused by a policy $\pi^j(a^j|s^j, \mu^j)$, which will be the optimal policy. In other words,

$$\begin{aligned} &\max_{a^j \in \mathcal{A}^j} \Big[r^j(s_t^j, a_t^j, \mu_t) + \\ &\qquad \beta \int_{\mathcal{S}} J_*^{j,\boldsymbol{\mu}}(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \Big] = \\ &r^j(s_t^j, \pi^j(a_t^j | s_t^j, \mu_t^j), \mu_t) \\ &\qquad + \beta \int_{\mathcal{S}} J_*^{j,\boldsymbol{\mu}}(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \end{aligned} \tag{70}$$

The objective is to obtain this optimal policy. Towards the same we use the Q-functions, where the optimal Q-function can be defined as

$$\begin{aligned} &Q_{\boldsymbol{\mu}}^{j,*}(s_t^j, a_t^j) \\ &= r^j(s_t^j, a_t^j, \mu_t) + \beta \int_{\mathcal{S}} J_*^{j,\boldsymbol{\mu}} p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \end{aligned} \tag{71}$$

The optimal value function given by $J_*^{j,\boldsymbol{\mu}}$ is the maximum of the Q-function given by $Q_{\boldsymbol{\mu},\max}^{j,*}$. Hence, we can rewrite Eq. 71 as follows

$$\begin{aligned} Q_{\boldsymbol{\mu}}^{j,*}(s_t^j, a_t^j) &= r^j(s_t^j, a_t^j, \mu_t) \\ &\qquad + \beta \int_{\mathcal{S}} Q_{\boldsymbol{\mu},\max_{a^j}}^{j,*}(s_{t+1}^j, a^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \\ &\triangleq H Q_{\boldsymbol{\mu}}^{j,*}(s_t^j, a_t^j) \end{aligned} \tag{72}$$

Here, the operator H is the optimality operator for the Q-functions. Our objective is to prove that this operator is a contraction and has a unique fixed point given by Q_*.

First, let us define a set of all bounded Q-functions for an agent $j \in \{1, \ldots, N\}$:

$$\begin{aligned} \mathcal{C} \triangleq \Big\{ &Q^j : \mathcal{S} \times \mathcal{A}^j \to [0, \infty); ||Q_{\max}^j||_w \leq \tfrac{M}{1-\beta\alpha} \\ &\text{and} \\ &||Q_{\max}^j||_{Lip} \leq \tfrac{L_2}{1-\beta K_2} \Big\} \end{aligned} \tag{73}$$

From Assumption 1, Assumption 12 and Assumption 13, we know that there exists a unique maximiser $\pi^j(s_t^j, Q_t^j, \mu_t^j)$ for the function F represented as,

$$\begin{aligned} &r^j(s_t^j, a_t^j, \mu_t) \\ &+\beta \int_{\mathcal{S}} Q_{\boldsymbol{\mu},\max_{a^j}}^j(s_{t+1}^j, a^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \\ &= F(s_t^j, Q_{\boldsymbol{\mu},\max}^j, \mu_t, a_t^j). \end{aligned} \tag{74}$$

.

This maximser makes the gradient of F, 0.

$$\nabla F(s_t^j, Q_{\boldsymbol{\mu},\max}^j, \mu_t, \pi^j(s_t^j, Q_t^j, \mu_t^j)) = 0 \tag{75}$$

This shows that the maximiser for the operator H_2 in Eq. 8 is unique, under the considered assumptions. Now, we are ready to prove the required theorem.

Theorem 3. *The decentralized mean field operator H is well-defined, i.e. this operator maps $\mathcal{C} \times \mathcal{P}(\mathcal{S})$ to itself.*

Proof. We first apply Assumption 1. Now, consider the decentralized mean field operator as defined in Eq. 8. To prove the given theorem, we know that $H_2(Q^j, \boldsymbol{\mu}) \in \mathcal{P}(\mathcal{S})$. We need to prove that $H_1(Q^j, \boldsymbol{\mu}) \in \mathcal{C}$. Let us consider an element $(Q^j, \boldsymbol{\mu}) \in \mathcal{C} \times \mathcal{P}(\mathcal{S})$. Then we have,

$$\begin{aligned} &\sup_{s_t^j, a_t^j} \frac{\left| H_1(Q^j, \boldsymbol{\mu})(s_t^j, a_t^j) \right|}{w(s_t^j, a_t^j)} \\ &= \sup_{s_t^j, a_t^j} \frac{\left| r^j(s_t^j, a_t^j, \mu) + \beta \int_{\mathcal{S}} Q_{\max}^j(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \right|}{w(s_t^j, a_t^j)} \\ &\leq \sup_{s_t^j, a_t^j} \frac{\left| r^j(s_t^j, a_t^j, \mu) \right|}{w(s_t^j, a_t^j)} \\ &\qquad + \beta \sup_{s_t^j, a_t^j} \frac{\left| \int_{\mathcal{S}} Q_{\max}^j(s_{t+1}^j, a_t^j, \mu_{t+1}) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \right|}{w(s_t^j, a_t^j)} \\ &\leq M \\ &+ \beta ||Q_{\max}^j||_{w_{\max}} \sup_{s_t^j, a_t^j} \frac{\left| \int_{\mathcal{S}} w_{\max}(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \right|}{w(s_t^j, a_t^j)} \\ &\leq M + \beta\alpha ||Q_{\max}^j||_w \\ &\leq M + \beta\alpha \tfrac{M}{1-\beta\alpha} \\ &= \tfrac{M}{1-\beta\alpha} \end{aligned} \tag{76}$$

We used the Assumption 10 and the fact that $||Q^j||_{w_{\max}} \leq ||Q^j||_w$.

Also, we have,

$$
\begin{aligned}
&|H_1(Q^j, \boldsymbol{\mu})_{\max}(s_t^j) - H_1(Q^j, \boldsymbol{\mu})_{\max}(\hat{s}_t^j)| \\
&= \max_{a^j \in \mathcal{A}^j} [r^j(s_t^j, a_t^j, \mu_t) \\
&\quad + \beta \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t)] \\
&\quad - \max_{a^j \in \mathcal{A}^j} [r^j(\hat{s}_t^j, a_t^j, \mu_t) \\
&\quad + \beta \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | \hat{s}_t^j, a_t^j, \mu_t)] \\
&\leq \sup_{a^j \in \mathcal{A}^j} \left| r^j(s_t^j, a_t^j, \mu_t) - r^j(\hat{s}_t^j, a_t^j, \mu_t) \right| \\
&+ \beta \sup_{a^j \in A^j} \Big| \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \\
&\quad - \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | \hat{s}_t^j, a_t^j, \mu_t) \Big| \\
&\leq L_2 d_X(s, \hat{s}) + \beta K_2 ||Q^j_{\max}||_{Lip} d_X(s, \hat{s}) \\
&\leq \tfrac{L_2}{1-\beta K_2} d_X(s, \hat{s})
\end{aligned} \tag{77}
$$

Here we are using the Assumption 7 and Assumption 8. This proves that $H_1(Q^j, \mu) \in \mathcal{C}$ and concludes our proof. □

D Proof of Theorem 4

Theorem 4. *Let $\mathcal{B}$ represent the space of bounded functions in $\mathcal{S}$. Then the mapping $H : \mathcal{C} \times \mathcal{P}(\mathcal{S}) \to \mathcal{C} \times \mathcal{P}(\mathcal{S})$ is a contraction in the norm of $\mathcal{B}(\mathcal{S})$.*

In this section, we will continue to use the set of assumptions and definitions we introduced in Appendix C.

Proof. Fix any (Q^j, μ) and $(\hat{Q}^j, \hat{\mu})$ in $\mathcal{C} \times \mathcal{P}(\mathcal{S})$, let us consider,

$$
\begin{aligned}
&||H_1(Q^j, \mu) - H_1(\hat{Q}^j, \hat{\mu})||_w \\
&= \sup_{s_t^j, a_t^j} \\
&\Big[\frac{r^j(s_t^j, a_t^j, \mu_t) + \beta \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t)}{w(s_t^j, a_t^j)} \\
&\qquad \frac{-r^j(s_t^j, a_t^j, \hat{\mu}_t) - \beta \int_{\mathcal{S}} \hat{Q}^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \hat{\mu}_t)}{w(s_t^j, a_t^j)} \Big] \\
&\leq \sup_{s_t^j, a_t^j} \frac{|r^j(s_t^j, a_t^j, \mu_t) - r^j(s_t^j, a_t^j, \hat{\mu}_t)|}{w(s_t^j, a_t^j)} \\
&+ \beta \sup_{s_t^j, a_t^j} \frac{\Big| \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t)}{w(s_t^j, a_t^j)} \\
&\qquad \frac{- \int_{\mathcal{S}} \hat{Q}^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \hat{\mu}_t) \Big|}{w(s_t^j, a_t^j)} \\
&\leq L_1 W_1(\mu, \hat{\mu}) \\
&\qquad + \beta \sup_{s_t^j, a_t^j} \frac{\Big| \int_{\mathcal{S}} Q^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t)}{w(s_t^j, a_t^j)} \\
&\qquad\quad \frac{- \int_{\mathcal{S}} \hat{Q}^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \Big|}{w(s_t^j, a_t^j)} \\
&\qquad + \beta \sup_{s_t^j, a_t^j} \frac{\Big| \int_{\mathcal{S}} \hat{Q}^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \Big|}{w(s_t^j, a_t^j)} \\
&\qquad\quad \frac{- \int_{\mathcal{S}} \hat{Q}^j_{\max}(s_{t+1}^j, a^j, \mu_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \hat{\mu}_t) \Big|}{w(s_t^j, a_t^j)} \\
&\leq L_1 W_1(\mu, \hat{\mu}) + \beta ||Q^j_{\max} - \hat{Q}^j_{\max}||_{w_{\max}} \\
&\qquad \sup_{s_t^j, a_t^j} \frac{\Big| \int_{\mathcal{S}} w_{\max}(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \Big|}{w(s_t^j, a_t^j)} \\
&+ \beta ||\hat{Q}^j_{\max}||_{Lip} \sup_{s_t^j, a_t^j} \frac{W_1(p(\cdot|s_t^j, a_t^j, \mu_t), p(\cdot|s_t^j, a_t^j, \hat{\mu}_t))}{w(s_t^j, a_t^j)} \\
&\leq L_1 W_1(\mu, \hat{\mu}) + \beta\alpha ||Q^j - \hat{Q}^j||_w + \beta \tfrac{L_2}{1-\beta K_2} K_1 W_1(\mu, \hat{\mu})
\end{aligned} \tag{78}
$$

First we apply Assumption 7. In the last step we apply Assumption 8 and Assumption 10.

Now consider the distance between $H_2(Q^j, \boldsymbol{\mu})$ and $H_2(\hat{Q}^j, \hat{\boldsymbol{\mu}})$. First, we will consider the difference between the unique maximiser $\pi^j(s^j, Q^j, \mu^j)$ of $H_1(Q^j, \boldsymbol{\mu})(s^j, a^j)$ and the unique maximiser $\pi^j(s^j, \hat{Q}^j, \hat{\mu}^j)$ of $H_1(\hat{Q}^j, \hat{\boldsymbol{\mu}})(s^j, a^j)$ with respect to the action (using the Assumption 1). Let us consider the function,

$$F : \mathcal{S} \times \mathcal{C} \times \mathcal{P}(\mathcal{S}) \times \mathcal{A}^j \ni (s_t^j, v_t^j, \mu_t, a_t^j)$$

$$\rightarrow r^j(s_t^j, a_t^j, \mu_t) + \beta \int_{\mathcal{S}} v(s_{t+1}^j) p(s_{t+1}^j | s_t^j, a_t^j, \mu_t) \in \Re \tag{79}$$

This is ρ-strongly convex by Assumption 12, hence it satisfies the following equation (Hajek and Raginsky 2019),

$$[\nabla F(s_t^j, v_t^j, \mu_t, a_t^j + b_t^j) - \nabla F(s_t^j, v_t^j, \mu_t, a_t^j)]^T \cdot r_t^j \geq \rho ||b_t^j||^2 \tag{80}$$

For any $a_t^j, b_t^j \in \mathcal{A}^j$ and for any $s_t^j \in \mathcal{S}$, we can use the notation $a_t^j = \pi^j(s_t^j, Q_t^j, \mu_t^j)$ and $b^j = \pi^j(s_{t+1}^j, \hat{Q}_t^j, \hat{\mu}_t^j) - \pi^j(s_t^j, Q_t^j, \mu_t^j)$.

Now, $a_t^j = \pi^j(s_t^j, Q_t^j, \mu_t^j)$ is the unique maximiser of the strongly convex function $F(s_t^j, Q_{max,t}^j, \mu_t, \cdot)$, we have

$$\nabla F(s_t^j, Q_{max,t}^j, \mu_t, \pi^j(s_t^j, Q_t^j, \mu_t^j)) = 0 \tag{81}$$

Also, the term $a_t^j + b_t^j = \pi^j(s_{t+1}^j, \hat{Q}_t^j, \hat{\mu}_t^j)$ is the unique maximiser of $F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, \cdot)$. Now, using Assumption 12 and Eq. 80 we have,

$$\begin{aligned} &-\nabla F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, a_t^j)^T \cdot b_t^j \\ &\quad = -\nabla F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, a_t^j)^T \cdot b_t^j \\ &\quad +\nabla F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, a_t^j + b_t^j)^T \cdot b_t^j \geq \rho ||b_t^j||^2 \end{aligned} \tag{82}$$

Similarly, using the Assumption 13 we also have,

$$\begin{aligned} &-\nabla F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, a_t^j)^T \cdot b_t^j \\ &= -\nabla F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, a_t^j)^T \cdot b_t^j \\ &\quad +\nabla F(s_t^j, Q_{max,t}^j, \mu_t, a_t^j)^T \cdot b_t^j \\ &\leq ||b_t^j|| || \nabla F(s_t^j, Q_{max,t}^j, \mu_t, a_t^j) \\ &\quad -\nabla F(s_{t+1}^j, \hat{Q}_{max,t}^j, \hat{\mu}_t, a_t^j) || \\ &\leq K_F ||b_t^j|| (d_X(s_t^j, s_{t+1}^j) + ||Q_{max,t}^j - \hat{Q}_{max,t}^j||_{w_{max}} \\ &\quad +W_1(\mu_t, \hat{\mu}_t)) \\ &\leq K_F ||b_t^j|| (d_X(s_t^j, s_{t+1}^j) + ||Q_t^j - \hat{Q}_t^j||_w \\ &\quad +W_1(\mu_t, \hat{\mu}_t)) \end{aligned} \tag{83}$$

Therefore, from the above two equations,

$$\begin{aligned} &\pi^j(s_{t+1}^j, \hat{Q}_t^j, \hat{\mu}_t^j) - \pi^j(s_t^j, Q_t^j, \mu_t^j) \\ &\leq \tfrac{K_F}{\rho}(d_X(s_{t+1}^j, s_t^j) + ||Q_t^j - \hat{Q}_t^j||_w + W_1(\mu_t, \hat{\mu}_t)) \end{aligned} \tag{84}$$

Now, consider the W_1 distance between $H_2(Q^j, \boldsymbol{\mu})$ and $H_2(\hat{Q}^j, \hat{\boldsymbol{\mu}})$.

$$\begin{aligned} &W_1(H_2(Q^j, \boldsymbol{\mu}), H_2(\hat{Q}^j, \hat{\boldsymbol{\mu}})) \\ &= \sup_{||g||_{Lip} \leq 1} \Big| \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, Q_t^j, \mu_t^j), \mu_t\big) \mu_t(s_t^j) \\ &\quad - \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, \hat{Q}_t^j, \hat{\mu}_t^j), \hat{\mu}_t\big) \hat{\mu}_t(s_t^j) \Big| \\ &\leq \sup_{||g||_{Lip} \leq 1} \Big| \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, Q_t^j, \mu_t^j), \mu_t\big) \mu_t(s_t^j) \\ &\quad - \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, \hat{Q}_t^j, \hat{\mu}_t^j), \hat{\mu}_t\big) \mu_t(s_t^j) \Big| \\ &+ \sup_{||g||_{Lip} \leq 1} \Big| \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, \hat{Q}_t^j, \hat{\mu}_t^j), \hat{\mu}_t\big) \mu_t(s_t^j) \\ &\quad - \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, \hat{Q}_t^j, \hat{\mu}_t^j), \hat{\mu}_t\big) \hat{\mu}_t(s_t^j) \Big| \\ &\overset{(1)}{\leq} \int_{\mathcal{S} \times \mathcal{A}^j} \sup_{||g||_{Lip} \leq 1} \Big| \int_{\mathcal{S}} g(s_{t+1}^j) \\ &\qquad p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, Q_t^j, \mu_t^j), \mu_t\big) \mu_t(s_t^j) \\ &- \int_{\mathcal{S} \times \mathcal{A}^j} \int_{\mathcal{S}} g(s_{t+1}^j) p\big(s_{t+1}^j | s_t^j, \pi^j(s_t^j, \hat{Q}_t^j, \hat{\mu}_t^j), \hat{\mu}_t\big) \Big| \\ &\qquad \mu_t(s_t^j) + (K_2 + \tfrac{K_F}{\rho}) W_1(\mu_t, \hat{\mu}_t) \\ &\leq \int_{\mathcal{S} \times \mathcal{A}^j} W_1\Big(p\big(\cdot \,| s_t^j, \pi^j(s_t^j, Q_t^j, \mu_t^j), \mu_t\big), \\ &\qquad p\big(\cdot \,| s_t^j, \pi^j(s_t^j, \hat{Q}_t^j, \hat{\mu}_t^j), \hat{\mu}_t\big)\Big) \mu_t(s_t^j) \\ &\qquad +\Big(K_2 + \tfrac{K_F}{\rho}\Big) W_1(\mu_t, \hat{\mu}_t) \\ &\overset{(2)}{\leq} \tfrac{K_F}{\rho} K_1 \Big(||Q_t^j - \hat{Q}_t^j||_w + W_1(\mu_t, \hat{\mu}_t)\Big) \\ &\qquad +K_1 W_1(\mu_t, \hat{\mu}_t) + \Big(K_2 + \tfrac{K_F}{\rho}\Big) W_1(\mu_t, \hat{\mu}_t) \end{aligned} \tag{85}$$

In the above derivation, (1) and (2) are obtained from Assumption 8 and Eq. 84 (also refer Anahtarci, Kariksiz, and Saldi (2019)). Combining Eq. 78 and Eq. 85, and applying Assumption 14, the required result is proved. The constant of contraction is k defined in Assumption 14.

□

E Proof of Theorem 5

Theorem 5. *Let the Q-updates in Algorithm 1 converge to* (Q^j_*, μ^j_*) *for an agent* $j \in \{1, \ldots, N\}$. *Then, we can construct a policy* π^j_* *from* Q^j_* *which is expressed as*

$$\pi^j_*(s^j) = \arg\max_{a^j \in \mathcal{A}^j} Q^j_*(s^j, a^j, \mu^j_*).$$

Then the pair (π^j_*, μ^j_*) *is a DMFE.*

Proof. From the Theorem 4 we know that (Q^j_*, μ^j_*) is a fixed point of H. Using the Assumption 1, we consider a μ_*, where $\mu_*(s^j) = \mu^j_*(s^j)$, for all states $s^j \in \mathcal{N}^j$. Hence, we can construct the following equations.

$$Q^j_*(s^j_t, a^j_t, \mu^j_*) = r^j(s^j_t, a^j_t, \mu_{*,t}) + \beta \int_{\mathcal{S}} Q^j_{*,\max}(s^j_{t+1}, a^j, \mu^j_{*,t+1}) p(s^j_{t+1} | s^j_t, a^j_t, \mu_{*,t}) \tag{86}$$

$$\mu^j_{*,t+1}(\cdot) = \int_{\mathcal{S}\times\mathcal{A}^j} p(\cdot | s^j_t, a^j_t, \mu_{*,t}) \pi^j_*(a^j_t | s^j_t, \mu^j_{*,t}) \mu_{*,t}(s^j_t) \tag{87}$$

From Eq. 87 and Assumption 1, we can construct the mean field estimate of agent$j$, $\boldsymbol{\mu}^j_*$. Now, the above equations imply that $\pi^j_* \in \Phi(\mu^j_*)$ and $\mu^j_* = \Psi(\pi^j_*)$. Hence, (π^j_*, μ^j_*) is a decentralized mean field equilibrium.

□

F Differences Between Mean Field Settings

Table 1 captures the differences between the three mean field frameworks, MFRL, MFG, and DMFG. Particularly, we show that the DMFG is different from other frameworks introduced previously, and it is more practical than prior methods as it relaxes some strong assumptions in them.

There are several disadvantages in using a centralized solution concept like Nash equilibrium (or, by extension, the mean field equilibrium) in multiagent systems. These are listed as follows: 1) Centralized nature of the equilibrium, though practical systems have a decentralized information structure. 2) The equilibrium computation is almost intractable for more than two agents (Neumann 1928). 3) Needs strong assumptions to give theoretical guarantees of convergence of learning systems in general sum games (Hu and Wellman 2003), even in the stationary case. 4) The equilibrium requires agreement between agents even in the competitive case. 5) It is hard to verify this point in practice. Our decentralized solution concept (DMFE) is more practical than the Nash equilibrium and the mean field equilibrium, since it does not suffer these limitations noted for the centralized methods.

Method	MFRL Yang et al. (2018)	MFG Lasry and Lions (2007)	DMFG (ours)
Game formulation	Stochastic Game	Mean Field Game	Decentralized Mean Field Game
Reward function	Same for all agents	Same for all agents	Can be different
Action space	Discrete only	Can be continuous	Can be continuous
Action space	Same for all agents	Same for all agents	Can be different
Solution Concept	Nash Equilibrium (centralized)	Mean Field Equilibrium (centralized)	Decentralized Mean Field Equilibrium
Complexity of obtaining the mean field	Exponential in agents	Exponential in agents	Constant in agents
Theoretical Guarantees	Needs very strong assumptions	Needs weak assumptions	Needs weak assumptions
Number of agents	Should be large but finite	Can be infinite in the limit	Can be infinite in the limit
Mean Field Information	Global	Global	Local
Agents	Identical and homogeneous	Identical and homogeneous	Non-identical
Policy	Stationary	Can be non-stationary	Can be non-stationary
State Observable	Global	Local	Local

Table 1: Table to capture the differences between Mean Field Reinforcement Learning (MFRL), Mean Field Games (MFG), and Decentralized Mean Field Games (DMFG).

G Mean Field Modelling in DMFG-QL

In this section, we plot the mean square error between the estimated mean field action ($\mu_t^{j,a}$) (from the neural network representing Eq. 14) and the true observed local mean field action ($\hat{\mu}_t^{j,a}$) to show that the true local mean field action can be accurately modelled by the DMFG-QL algorithm. We will use the Battle game for this illustration.

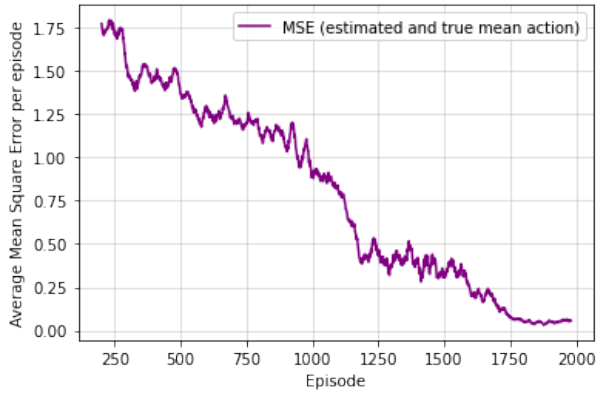


Figure 7: This figure shows the average (per agent) mean square error (MSE) between the true local mean field action and the estimated mean field action from the neural network in each episode (sum of MSE in the 500 steps) of the Battle game training experiment. The results show that the MSE steadily reduces, and hence DMFG-QL is able to accurately model the true mean field. The result shows the average of 10 runs (negligible standard deviation).

Fig. 7 shows the mean square errors (MSE) between the true mean field action and the estimated mean field action, averaged per agent, for each episode of the Battle game. The MSE converges to a small value close to 0 during training. This shows that the DMFG-QL algorithm can accurately model the mean field, which contributes to its superior performance in many of our experimental domains. As stated in Section 6, DMFG-QL algorithm formulates best responses to the current mean field action. This contrasts with approaches such as MFQ that formulate best responses to the previous mean field, which leads to a loss in performance. We start the plot in Fig 7 from episode 200 (instead of 0), since the first few episodes have a very high MSE, that simply skews the axis of the resulting graph.

H Algorithm Pseudo-Code

Algorithm 2 provides the pseudo-code of the DMFG-QL algorithm. Here, the neural networks are used as the function approximator for the Q-function. In addition, we include the use of a second target network and a replay buffer for training, as done in the popular Deep Q-learning (DQN) algorithm (Mnih et al. 2015). Also, similar to Yang et al. (2018), we specify that the policies satisfy the GLIE assumption and hence we use the max operator over the Q-value of the next state (i.e. choose the action that maximizes the Q value) to calculate the temporal difference (T.D.) error instead of maintaining a distinct value function as in Eq. 13.

Algorithm 3 provides the pseudo-code of the DMFG-AC algorithm. This algorithm uses the policy as the actor, and the value function as the critic, as is common in practice (Konda and Tsitsiklis 1999). We choose to use a stochastic actor that does not depend on the mean field. This is done to provide an algorithm that can work independent of the mean field during execution. Since the critic (not used during execution) can be dependent on the mean field, we additionally parameterize the value function with the mean field. The modified updates can be seen in Algorithm 3. For both Algorithm 2 and Algorithm 3 we initialize the estimated mean field distribution to a uniform distribution (arbitrarily).

I Experimental Details

In this section, we describe each of our game domains in detail, including the reward functions. We also provide the implementation details of our algorithms, especially the hyperparameters. We also discuss the wall-clock times of our algorithmic implementations. Each episode for the Petting Zoo environments has a maximum of 500 steps. In our implementation of MFQ and MFAC, the agents learn in a decentralized fashion with independent networks and use the previous mean field of the local neighbourhood instead of the global mean field.

The first 5 domains are obtained from the Petting Zoo environment (Terry et al. 2020), and the game parameters are mostly left unchanged from those given in Terry et al. (2020). In these games, as agents can die during game play, we use the agent networks saved in the last available episode during training for the execution experiments. Complete details of these domains are given in each of the sub-sections below.

Battle Domain

This is the first domain from the Petting Zoo environment (Terry et al. 2020) that is mixed cooperative-competitive. This domain was originally defined in the MAgents simulator (Zheng et al. 2018). We have two teams of 25 agents, each learning to cooperate against the members of the same team and compete against the members of its opponent team. The agent gets rewarded for attacking and killing agents of the opposing team. At the same time, the agent is penalized for being attacked. In our implementation, each agent learns using its local observation and cannot get global information. Each agent has a view range of a circular radius of 6 units around it. Most rewards are left as defaults, as given in Terry et al. (2020). The agents get a penalty of -0.005 for each step (step reward) and a penalty of -0.1 for being killed. The agents get a reward of 5 for attacking an opponent and a reward of 10 for killing an opponent. There is an attack penalty of -0.1 (penalty for attacking anything). Agents start with a specific number of hitpoints (HP) that are damaged upon being attacked. Agents lose 2 HP when attacked and recover an HP of 0.1 for every step that they are not attacked. They start with 10 HP, and when this HP is completely lost, the agent dies. Each agent can view a circular range of 6 units around it (view range) and can attack in a circular range of 1.5 units around it (attack range). The action space of each agent is a discrete set of 21 values, which corresponds to

Algorithm 2: Q-learning for Decentralized Mean Field Games (DMFG-QL)

Initialize the Q-functions (parameterized by weights) Q_{ϕ^j}, Q_{ϕ^j}, for all agents $j \in \{1, \ldots, N\}$
Initialize the mean field estimate (parameterized by weights) $\mu_{\eta}^{j,a}$ for each agent $j \in \{1, \ldots, N\}$
Initialize the estimated mean field $\mu_{0,\eta}^{j,a}$ for each j to a uniform distribution
Initialize the total steps (T) and total episodes (E)
Obtain the current state s_t^j
while Episode $<$ E **do**
  while Step $<$ T **do**
    For each agent j, obtain action a_t^j from the policy induced by Q_{ϕ^j} with the current estimated mean action $\mu_{t,\eta}^{j,a}$ and the exploration rate $\hat{\beta}$ according to Eq. 15
    Execute the joint action $\mathbf{a}_t = [a_t^1, \ldots, a_t^N]$. Observe the rewards $\mathbf{r}_t = [r_t^1, \ldots, r_t^N]$ and the next state $\mathbf{s}_{t+1} = [s_{t+1}^1, \ldots, s_{t+1}^N]$
    For each agent j, obtain the current observed local mean action ($\hat{\mu}_t^{j,a}$)
    Update the parameter η of the mean field network using a mean square error between the observed mean action $\hat{\mu}_t^{j,a}$ and the current estimated mean action $\mu_{t,\eta}^{j,a}$ for all j
    For each j, obtain the mean field estimates $\mu_{t+1,\eta}^{j,a}$ for the next state s_{t+1}^j using the mean field network according to Eq. 14
    For each j, store $\langle s_t^j, a_t^j, r_t^j, s_{t+1}^j, \mu_{t,\eta}^{j,a}, \mu_{t+1,\eta}^{j,a} \rangle$ in replay buffer B
    Set the next mean field estimate as the current mean field estimate $\mu_{t,\eta}^{j,a} = \mu_{t+1,\eta}^{j,a}$ and the next state as the current state $s_t^j = s_{t+1}^j$ for all agents $j \in \{1, \ldots, N\}$
  end while
  while j = 1 to N **do**
    Sample a minibatch of K experiences $\langle s_t^j, a_t^j, r_t^j, s_{t+1}^j, \mu_{t,\eta}^{j,a}, \mu_{t+1,\eta}^{j,a} \rangle$ from B
    Set $y^j = r_t^j + \gamma \max_{a_{t+1}^j} Q_{\phi_-^j}(s_{t+1}^j, a_{t+1}^j, \mu_{t+1,\eta}^{j,a})$ according to Eq. 12
    Update the Q network by minimizing the loss $L(\phi^j) = \frac{1}{K} \sum (y^j - Q_{\phi^j}(s_t^j, a_t^j, \mu_{t,\eta}^{j,a}))^2$
  end while
  Update the parameters of the target network for each agent j with learning rate τ;

$$\phi_-^j \leftarrow \tau \phi^j + (1 - \tau) \phi_-^j$$

end while

taking moving and attacking actions in the environment. In this game, we assume that agents can get perfect information about actions of other agents at a distance of 6 units from themselves and no information beyond this point. In the execution phase, a game is considered to be won by a team

Algorithm 3: Actor-Critic for Decentralized Mean Field Game (DMFG-AC)

Initialize the V-function or critic (parameterized by weights) V_{ϕ^j} and the policy or actor (parameterized by weights) π_{θ^j} for all agents $j \in 1, \ldots, N$
Initialize the mean field estimate (parameterized by weights) $\mu_{\eta}^{j,a}$ for each agent $j \in \{1, \ldots, N\}$
Initialize the estimated mean field $\mu_{0,\eta}^{j,a}$ for each j to a uniform distribution
Initialize the total episodes (E)
Obtain the current state s_t^j
while Episode $<$ E **do**
  For each agent j, obtain action a_t^j from the policy $\pi_{\theta^j}(s_t^j)$
  Execute the joint action $\mathbf{a}_t = [a_t^1, \ldots, a_t^N]$. Observe the rewards $\mathbf{r}_t = [r_t^1, \ldots, r_t^N]$ and the next state $\mathbf{s}_{t+1} = [s_{t+1}^1, \ldots, s_{t+1}^N]$
  For each agent j, obtain the current observed local mean field action ($\hat{\mu}_t^{j,a}$)
  Update the parameter η of the mean field network using a mean square error between the observed mean action $\hat{\mu}_t^{j,a}$ and the current estimated mean action $\mu_{t,\eta}^{j,a}$ for all j
  For each j, obtain the mean field estimates $\mu_{t+1,\eta}^{j,a}$ for the next state s_{t+1}^j using the mean field network according to Eq. 14
  Set $y^j = r_t^j + \gamma V_{\phi^j}(s_{t+1}^j, \mu_{t+1,\eta}^{j,a})$ as the T.D. target according to Eq. 12
  For each j, update the critic by minimizing the loss $L(\phi^j) = (y^j - V_{\phi^j}(s_t^j, \mu_{t,\eta}^{j,a}))^2$
  For each j, update the actor using the log loss $\mathcal{J}(\theta^j) = \log \pi_{\theta^j}(s) L(\phi^j)$
  Set the next mean field estimate as the current mean field estimate $\mu_{t,\eta}^{j,a} = \mu_{t+1,\eta}^{j,a}$ and the next state as the current state $s_t^j = s_{t+1}^j$ for all agents $j \in \{1, \ldots, N\}$
end while

that kills more opponent agents. If both teams kill the same number of agents, the team having the higher cumulative reward is determined as the winner.

Gather Domain

In this environment, all agents try to capture limited food available in the environment and gain rewards. Each food needs to be attacked before it can be captured (takes 5 attacks to capture food). This is a fully competitive game, where all agents try to outcompete others in the environment and gain more food for themselves. Agents can also kill other agents in the environment by attacking them (just a single attack). Each agent gets a step penalty of -0.01, an attack penalty (penalty for attack) of -2 and a death penalty (punishment for dying) of -20. Also, each agent gets a reward of 20 for attacking food and a reward of 60 for capturing the food. There are a total of 30 agents learning in our environment. The action space of each agent is a set of 33 values. The agents have a

view range of 7 and an attack range of 1. All other conditions and rewards are the same as the Battle game.

Combined Arms Domain

The Combined Arms environment is a heterogeneous large-scale team game with two types of agents in each team. The first type is a ranged agent, which can move fast and attack agents situated further away but has fewer HP. The second type is the melee agent that can only attack close-by agents and move slowly; however, they have more HP. The reward function for the agents is the same as that given in the battle game. The action space of the melee agents is a discrete set of 9 values, while the action space of the ranged agents is a discrete set of 25 values. The actions correspond to moving in the environment and attacking neighbouring agents. The ranged agents have a maximum HP of 3, and the melee agents have a maximum HP of 10. The maximum HP is the limit after which the HP of any agent cannot increase in this environment. Like the battle domain, agents lose 2 HP for each time they are attacked and gain 0.1 HP for each step without being attacked. In our experiments, each team consists of 25 agents, with 15 ranged and 10 melee agents at the start. The view range is 6 and the attack range is 1 for melee agents. The view range is 6 and attack range is 2 for ranged agents. All other conditions and rewards are the same as the Battle game. To create the mean field we choose to use the action space of the type which has the larger number of actions (i.e. we use the action space of the ranged agents).

Tiger-Deer Domain

In the tiger-deer environment, tigers are the learning agents, cooperating with each other to kill deer in the environment. At least two tigers need to attack a deer together to get high rewards. The tigers start with an HP of 10 and gain an HP of 8 whenever they kill deer. The tigers lose an HP of 0.021 at each step they do not eat a deer and die when they lose all the HP. In this game, the deer move randomly in the environment, and start with an HP of 5 and lose 1 HP upon attack. The tiger gets a reward of +1 for attacking a deer alongside another tiger. In this game, each tiger is assumed to get perfect information about the actions of other tigers at a distance of 4 units from itself and no information beyond this point. The view range of the tiger is 4 and the attack range is 1. The tigers also get a shaping reward of 0.2 for attacking a deer alone. All other rewards and conditions are the same as the Battle game.

Waterworld Domain

The Waterworld domain was first introduced as a part of the Stanford Intelligent Systems Laboratory (SISL) environments by Gupta, Egorov, and Kochenderfer (2017). We use the same domain adapted by the Petting Zoo environment (Terry et al. 2020). This is a continuous action space environment, where a group of pursuer agents aim to capture food and avoid poison. These pursuers are the learning agents, while both food and poison move randomly in the environment. This is a cooperative environment where pursuer agents need to work together to capture food. At least two agents need to attack a food particle together to be able to capture it. The action is a two-element vector, where the first element corresponds to horizontal thrust and the second element corresponds to vertical thrust. The agents choose to use a thrust to make themselves move in a particular direction with a desired speed. The action values are in the range [-1, +1]. Our domain contains 25 pursuer agents, 25 food particles and 10 poison particles. The food is not destroyed but respawned upon capture. The agents get a reward of +10 upon capturing food and a penalty of -1 for encountering poison. If a single agent encounters food alone, it gets a shaping reward of +1. Each time an agent applies thrust, the agent obtains a penalty of thrust penalty $\times$ ||action||, where the value of thrust penalty is -0.5. Since this is a cooperative environment, each agent is allowed access to the global mean field action which is composed of actions of all agents in the environment, at each time step. This is done to simplify this complex domain.

Ride-Sharing Domain

In this domain, our problem formulation and environment are the same as that described in Shah, Lowalekar, and Varakantham (2020). The demand distribution is obtained from the publicly available New York Yellow Taxi Dataset (NYYellowTaxi 2016). The overall approach follows six steps. First, the user requests are obtained from the dataset. Second, sets of feasible trips are generated (by an oracle) using the approach in Alonso-Mora et al. (2017). These feasible trips keep the action space from exploding. Third, the feasible actions are scored by the individual agents using their respective value functions. Fourth, a mapping of requests takes place by checking different constraints. Fifth, the final mapping is used to update the rewards for the individual agents. Sixth, the motion of vehicles is simulated until the next decision epoch.

We consider a maximum of 120 vehicles in our experiments. Since we are learning in a decentralized fashion, each vehicle maintains its own network and is computationally intensive. However, in practical applications, the training can be parallelized across agents and the computational demands will not be a limitation of our proposed setting.

Our goal in this experiment is to implement the mean field algorithm on a real-world problem and compare the performance to other state-of-the-art approaches. The experimental setup is along the lines of Alonso-Mora et al. (2017); Shah, Lowalekar, and Varakantham (2020), where we restrict ourselves to the street networks of Manhattan, where the vast majority of requests are contained. The New York Yellow Taxi dataset contains information about ride requests at different times of the day during a given week. Similar to Shah, Lowalekar, and Varakantham (2020), we use the pickup and drop-off locations, pickup times and travel times from the dataset. The dataset has about 330,000 requests per day. In our experiments, the taxis are assigned a random location at the start and react to incoming ride requests. We train the networks using data pertaining to 8 weekdays and validate it on a single day as done in Shah, Lowalekar, and Varakantham (2020). We assume all vehicles have similar capacities for simplicity, although our decentralized set-up is completely applicable to environments with very different types of vehi-

cles as against prior work that relied on centralized training (Lowalekar, Varakantham, and Jaillet 2019; Shah, Lowalekar, and Varakantham 2020).

In the experiments, a single day of training corresponds to one episode. Each episode has 1440 steps, where each step (decision unit) is considered to span one minute. The initial location of vehicles at the beginning of an episode (single day of training) is random. The state, action space and reward function in our system is the same as that in Shah, Lowalekar, and Varakantham (2020). The state of the system can be described as a tuple (c_t, u_t), where c_t is the location of each vehicle j denoted by $c_t^j = (p^j, t, L^j)$ representing its trajectory. This captures the current location and a list of future locations that each vehicle visits. The user request i at the time t is represented as $u_t^i = (o^i, e^i)$ which corresponds to the origin and destination of the requests. The action for each agent is to provide a score for all the requests with the objective of assigning the user requests to itself. The user request should satisfy the constraints at the vehicle level (maximum allowed wait time and capacity limits) and the constraints at the system level (each request is only assigned to a single agent). Since these constraints are environmental, we continue to use a central agent that performs the constraints check before assigning requests to individual agents. Each agent gets a reward based on the proportion of requests in its feasible action set that it is able to satisfy, as done in Shah, Lowalekar, and Varakantham (2020).

J Hyperparameters and Implementation Details

The implementation of DMFG-QL, IL and MFQ almost use the same hyperparameters, with the learning rate set as $\alpha = 10^{-2}$. The temperature for the Boltzmann policy is set as 0.1. Additionally, we also conduct epsilon greedy exploration which is decayed from 20% to 1% during the training process. The discount factor γ is equal to 0.9. The replay buffer size is 2×10^5, and the agents use a mini-batch size of 64. The target network is updated at the end of every episode.

Our implementations of DMFG-AC and MFAC almost use the same hyperparameters, where the learning rate of the critic is 10^{-2} and the learning rate of the actor is 10^{-4}. The mean field network of the DMFG-AC uses a learning rate of 10^{-2}. The discount factor is the same as the other three algorithms. In our implementation of MFAC, we do not use replay buffers unlike the implementation of Yang et al. (2018). We found this version of the actor-critic algorithm using the current updates (instead of delayed updates through the replay buffer) is more stable and performs better than the implementation of Yang et al. (2018) in our experiments. Additionally, our implementations of both MFAC and DMFG-AC use complete decentralization during execution where the actors only need to use their local states (mean field does not need to be maintained anymore). Also since a separate network is being maintained for the stochastic policy (actor) we do not use the Boltzmann policy for the actor-critic based methods (MFAC and DMFG-AC).

For the continuous action space Waterworld environment, every component of the obtained estimated mean field is normalized to be in the range $[-1, 1]$ (the range of the action values) and we do not use a softmax for the output layer (since this a mixture of Dirac deltas as discussed in Section 7).

Most hyperparameters are the same or closely match those used by prior works (Yang et al. 2018; Subramanian et al. 2020; Guo et al. 2019; Subramanian et al. 2021) in many agent environments.

For the ride-sharing experiments, DMFG-QL uses the same network architecture as given in Shah, Lowalekar, and Varakantham (2020) for the NeurADP algorithm. The implementation of CO and NeurADP uses the implementation provided by Shah, Lowalekar, and Varakantham (2020) except that all agents are fully decentralized as mentioned before. Unlike the approach in Shah, Lowalekar, and Varakantham (2020), each of our agents train their independent neural network using their local experiences. This network learns a suitable value function that can assign an appropriate score to each of the ride requests. This is done for both our implementations of NeurADP and DMFG-QL. For our baseline of CO (Alonso-Mora et al. 2017), we used the implementation from Shah, Lowalekar, and Varakantham (2020), which used the immediate rewards as the score for the given requests along with a small bonus term pertaining to the time taken to process a request (faster processing of requests is encouraged). The mean field for the DMFG-QL implementation is obtained by processing a distribution of the ride requests across every node in the environment at each step. This mean field is made available to all agents during both training and testing. Also, we use a slightly different architecture for estimating the mean field in this domain (3 Relu layers of 50 nodes and an output softmax layer).

The DDPG hyperparameters are based on Lillicrap et al. (2016) and the PPO hyperparameters are based on Schulman et al. (2017). DDPG uses the learning rate of the actor as 0.001 and that of the critic as 0.002 and a discount factor of 0.9. We use the soft replacement strategy with learning rate 0.01. The batch size is 32. The PPO implementation uses the same batch size and discount factor. The actor learning rate is 0.0001 and critic learning rate is 0.0002. Independent PPO uses a single thread implementation, since the data correlations are already being broken by the non-stationary nature of the environment induced by the multiple agents. This is also computationally efficient.

We use a set of 30 random seeds (1 – 30) for all training experiments and a new set of 30 random seeds (31 – 60) for all execution experiments.

K Complexity Analysis

A tabular version of our DMFG-QL algorithm is guaranteed to be linear in the total number of states, polynomial in the total number of actions, and constant in the number of agents. These guarantees are the same for both space complexity and time complexity. Comparatively, a tabular version of mean field Q-learning (MFQ) algorithm from Yang et al. (2018) has a space complexity that is linear in the number of states, polynomial in the number of actions and exponential in the number of agents. This is due to Eq. 3 requiring each agent to maintain Q-tables of all other agents. The time complexity is also the same as the space complexity in this case, since

each entry in the table may need to be accessed in the worst case.

L More Related Work

Mean field games have been used in the inverse reinforcement learning paradigm, where the objective is to determine a suitable reward function given expert demonstrations (Yang et al. 2017). Model-based learning solutions have also been proposed for mean field games (Kizilkale and Caines 2013; Yin et al. 2014), though these methods suffer from restrictive assumptions on the model and the analysis does not generalize to other models. Methods such as Cardaliaguet and Hadikhanloo (2017) perform a very computationally expensive computation using the full knowledge of the environment, which does not scale to large real-world environments. The work by Fu et al. (2019) provides a mean field actor-critic algorithm along with theoretical analysis in the linear function approximation setting, unlike other works which only analyze the tabular setting (Guo et al. 2019; Yang et al. 2018). However, it is not clear if this algorithm is strong empirically since it does not contain empirical experiments that illustrate its performance. Further, this work considered the linear-quadratic setting that contains restrictions on the type of reward function. Mguni, Jennings, and de Cote (2018) explore the connection between MARL and mean field games in the model-free setting.

The mean field setting under the cooperative case can be handled using a mean field control model (Carmona, Laurière, and Tan 2019a). In the linear-quadratic setting, Carmona, Laurière, and Tan (2019a) prove that policy-gradient methods converge to local equilibrium. In further work, the same authors prove that Q-learning algorithms also converge (Carmona, Laurière, and Tan 2019b).

In the experiments, we consider a real-world application of our methods on the Ride-Pool Matching Problem (RMP) as originally defined in Alonso-Mora et al. (2017). This is the problem considered by top ride-sharing platforms such as UberPool and Lyft-Line. The problem studies the efficiency of accepting ride requests by individual vehicles in such a way that the vehicles make more money (cater to more requests) per trip and the ride-sharing platform improves its ability to serve more orders. Previous approaches to solving the RMP problem have used standard optimization techniques (Ropke and Cordeau 2009). However, these methods are not scalable to large environments. One example from this class of methods is the zone path construction approach (ZAC) (Lowalekar, Varakantham, and Jaillet 2019). The ZAC solves for an optimization objective where the environment is abstracted into zones and each zone path represents a set of trips. The available vehicles are assigned to suitable zone paths using the ZAC algorithm. Another proposed solution was to make greedy assignments (Alonso-Mora et al. 2017), which considers maximizing the immediate returns and not the long term discounted gains traditionally studied in RL. Prior work has also considered RL approaches for this problem (Xu et al. 2018; Wang et al. 2018). However, these works consider very restrictive settings, which do not model multiagent interactions between the different vehicles. The work by Li et al. (2019) used a mean field approach for order dispatching, however, it assumes vehicles serve only one order at a time. The recent work by Shah, Lowalekar, and Varakantham (2020) introduced a very general formulation for the RMP where vehicles of arbitrary capacity are designed to serve batches of requests, with the whole solution scalable to thousands of vehicles and requests. They introduced a Deep Q-Network (DQN) (Mnih et al. 2015) based RL approach (called neural approximate dynamic programming or NeurADP) that learns efficient assignment of ride requests. However, the proposed approach assumes a set of identical vehicles and uses centralized training, where all vehicles learn the same policy using centralized updates. This is not practical in real-world environments, where the individual vehicles are typically heterogeneous (many differences in vehicular capacity and preferences). We propose a mean field based decentralized solution to this problem, which is more practical than the approach by Shah, Lowalekar, and Varakantham (2020).

M Wall Clock Times

The training for all the experiments on the simulated Petting Zoo domains was run on a 2 GPU virtual machine with 16 GB GPU memory per GPU. We use Nvidia Volta-100 (V100) GPUs for all these experiments. The CPUs use Skylake as the processor microarchitecture. We have a CPU memory of 178 GB. The Battle, Combined Arms and Gather experiments take an average of 5 days wall clock time to complete for all the considered algorithms. The Tiger-Deer experiments take an average of 4 days wall clock time to complete and the Waterworld experiments take an average of 2 days wall clock time to complete.

The majority of our experiments on the RMP problem were run on a virtual machine with 4 Ampere-100 GPUs with a GPU memory of 40 GB each. The CPUs use Skylake as the processor microarchitecture. Each training takes an average of 5 days to complete execution.

N Statistical Significance Tests

We run a statistical significance test for all the main results in our paper. In the MAgent environments, for the training results we conduct an unpaired two-sided t-test at the last episode (2000) of training. For the execution results, we conduct a Fischer's exact test for the average performances. The tests are between the best performing algorithm (DMFG-QL or DMFG-AC) and the best performing baseline (IL, MFQ, MFAC). In the ride-sharing experiments we conduct an unpaired two-sided t-test for the average and standard deviation of performances in Figure 6. The test is conducted between DMFG-QL and NeurADP (second best performing algorithm). We report the p-values for all the tests. As convention, we treat all p-values less than 0.05 as statistically significant outcomes.

From the results in Table 2, we see that the better performance given by our algorithms (DMFG-QL or DMFG-AC) is statistically significant in all domains except the Tiger-Deer domain. As noted in Section 7, the MFQ and MFAC algorithms perform as well as the DMFG-QL and DMFG-AC algorithms in this cooperative domain. Though we observe

Experimental Domain	Training (p-value)	Testing (p-value)	Statistically Significant
Battle	$p < 0.01$	$p < 0.01$	Yes
Combined Arms	$p < 0.01$	$p < 0.01$	Yes
Gather	$p < 0.01$	$p < 0.01$	Yes
Tiger-Deer	$p < 0.5$	$p < 0.7$	No
Waterworld	$p < 0.01$	$p < 0.01$	Yes

Table 2: Statistical significance of our results in simulated experiments.

# of Vehicles (N)	Maximum Pickup Delay (τ) (sec)	Capacity (c)	p-value	Statistically Significant
80	580	10	$p < 0.01$	Yes
100	580	10	$p < 0.01$	Yes
120	580	10	$p < 0.01$	Yes
100	520	10	$p < 0.01$	Yes
100	640	10	$p < 0.01$	Yes
100	580	8	$p < 0.01$	Yes
100	580	12	$p < 0.02$	Yes

Table 3: Statistical significance of our results in the ride-sharing domain.

that DMFG-QL gives the best overall average performance in both training and execution in the Tiger-Deer environment, the results are not statistically significant as noted from the p-values in Table 2.

The statistical significance results for the ride-sharing experiment in Table 3 shows that our observations regarding the superior performance of DMFG-QL are statistically significant.