Multi Agent Reinforcement Learning for Karma Economies

ETH zürich

AUTOMATIC CONTROL LABORATORY

Institute for Dynamic Systems and Control
IDSC
Institut für Dynamische Systeme und Regelungstechnik

Saurabh Vaishampayan

Multi Agent Reinforcement Learning for Karma Economies

Master Thesis

Automatic Control Laboratory
Swiss Federal Institute of Technology (ETH) Zurich

Supervision

Prof. Dr. Florian Doerfler, Prof Dr. Niao He,
Ezzat Elokda, Batuhan Yardim

March 2024

Contents

Abstract

Recently, new class of dynamic resource allocation mechanisms, called karma mechanisms, have shown great promise in acheiving fair and efficient outcomes. The analysis of karma mechanisms relies on the availability of versatile computational tools to predict the Nash equilibrium of a "karma game". Existing algorithms used to compute Nash equilibria of a karma game are centralized in nature, severely limiting the size and complexity of problems they can address. In this thesis, we formulate the karma game as a Multi Agent Reinforcement Learning (MARL) problem and adopt MARL techniques to compute Nash equilibria in arbitrarilty complex karma games. In our study of MARL for large populations, we found that a mature understanding of how to empirically assess convergence to a Nash equilibrium is lacking. For this reason, we first develop empirical convergence measures in a previously studied problem instance with known Nash equilibrium, before tackling a complex problem involving a grid of roads in a city center. Motivated by the observed empirical convergence, we moreover survey the state of the art on theoretical convergence guarantees in large population MARL, highlighting the shortcomings of existing methods and the current gap between theory and practice.

Nomenclature

Acronyms and Abbreviations

MARL	Multi Agent Reinforcement Learning
RL	Reinforcement Learning
MFG	Mean Field Game
PBP	Pay Bid to Peer
PBS	Pay Bid to Society
MCMC	Markov Chain Monte Carlo
PMA	Policy Mirror Ascent

Chapter 1

Introduction

In this chapter, we introduce the problem, review recent work in the field and summarise our main contributions. We first introduce karma games and review some of the key results and limitations from previous work. We then review Multi-Agent Reinforcement Learning (MARL) approaches used to compute Nash equilibria in general sum dynamic games. Finally we motivate the use of MARL in studying Nash equilibria of general karma games and summarise our contributions towards this thesis.

1.1 Previous work

Karma Games

Karma mechanisms are a class of resource allocation mechanisms with an artificial currency called karma. A karma game consists of players bidding for resource access using this artificial currency. Karma games have been previously shown to encourage truthful revelation of private urgencies while also allowing socially equitable outcomes[3][4]. Karma mechanisms accomplish this by several means. Firstly, the artificial currency is non-tradeable, meaning karma cannot be exchanged for any other intrinsic sources of value, making the system of agents form a self-contained economy[3]. Secondly, users can gain karma by giving up access to a resource. In a dynamic karma game, a player plans his bidding strategies in order to preserve karma for future resource access demands. Thus karma game faciliate turn taking but also incentive compatibility by allowed users to express their urgencies through karma.

While previous works have demonstrated the benefits of karma mechanisms in the context of outcome efficiency and socially equitable outcomes, they rely on the following restrictive assumptions. Firstly, the Nash strategies are obtained assuming the karma game is played assuming infinite population of agents and identical strategies for agents, with all agents having full access to the multi-agent game model. An open question that remains is whether the same results will apply if players are allowed to independently and greedily learn their individual strategies. Additionally, while the methods used to compute Nash-strategies in [3] converge in a variety of cases, the algorithm used to learn the Nash strategies has currently no convergence guarantees. Finally, the karma games studied so far had a tractable analytical structure for the multi-agent model, and performance in case of complex resource competition games is not known yet. We attempt to address these in our work.

Multi-Agent Reinforcement Learning

Reinforcement Learning refers to the setting where an agent learns an optimal strategy to maximize this payoffs in an unknown environment[16]. Multi-Agent Reinforcement Learning is the multi-agent generalization of the above, where agents learn their optimal strategies in a cooperative or

competitive game. During the learning process, agents execute a strategy, gather empirical data on rewards and estimate their payoffs for the corresponding strategy. They must pick a strategy that maximizes payoffs but in order to learn the optimal strategy w.r.t. payoffs they must also explore the strategy space to estimate payoffs from observations. Hence, during learning, agents balance strategy space exploration and payoff exploitation. Various algorithms for learning in multi agent games have been devised, including Independent Q Learning[17], Counterfactual Policy Gradients[5] and QMIX[14] in cooperative multi agent settings as well as algorithms like two timescale decentralized Q learning[15] for zero sum games.

A key challenge in multi-agent games is the exponential growth of joint state-action space, which makes analysis of policy convergence difficult. Additionally, Multi-Agent Games have nonstationary environments, i.e. the model changes if few of the agents update their strategies. This problem simplifies in a special case of multi-agent games where one has infinite and identical agents, called Mean Field Games[7]. Previous works[7] have shown that mean field equilibria are approximate Nash equilibria of the finite population game with a large enough population. Moreover, conditions that guarantee convergence of different MARL algorithms to a mean field equilibrium, and correspondingly to a Nash equilibrium in a finite but large enough population, have been derived in [18][13][1][12]. Existing conditions for convergence are abstract and it is not yet well understood how restrictive they are in practice, and a study of these in dynamic resource allocations games like karma games is lacking.

1.2 Contributions

We summarise our contributions for this thesis as follows

1. Development of a modular, highly parallelized software framework that learns Nash equilibria in general karma games using decentralized and centralized Multi-Agent Reinforcement Learning algorithms.
2. Analysis of various measures to empirically track convergence in MARL and a novel algorithm for off-policy computation of exploitability.
3. Survey state of the art convergence proof techniques in mean field games and apply them to karma games.
4. Benchmarking of our MARL algorithm against the previous work in the context of karma game of [3].
5. Numerical studies of karma mechanisms applied in context of a complex setting involving the congested centre of a grid-city.

Chapter 2

Preliminaries

In this chapter we summarise key concepts and results from literature in multi-agent reinforcement learning, mean field games and karma games that are essential towards the thesis.

2.1 Multi Agent Reinforcement Learning

Systems of multiple interacting agents are common in economics, social sciences and biology. The manner of interactions and behaviour of the agents is essential in understanding the behaviour of the resulting systems in the above contexts. Multi-agent systems can be studied as games consisting of rational agents, in fully cooperative, competitive or mixed settings[10][20]. When analysing behaviour of multi-agent systems, a key concept is to study agents acting according to their optimal behaviour (strategies). In a general game, the players' optimal strategy depends on the strategies pursued by others, necessitating the study of fast algorithms to obtain the set of optimal strategies. In certain cases, where each player has knowledge of his payoffs under the set of all possible strategies of other agents in analytical form, one has dynamic programming algorithms for obtaining optimal strategies. However, such cases are rare in games with complex structure, and one often uses learning algorithms based on empirical observations to obtain optimal strategies.

In this work, we restrict our focus towards study of dynamic multi-agent games. An important approach towards obtaining optimal strategies for multiple agents playing a dynamic game is using reinforcement learning[16] for multiple agents. Here, agents execute a strategy, gather empirical data on rewards and estimate their payoffs for the corresponding strategy. They must pick a strategy that maximizes payoffs but in order to learn the optimal strategy w.r.t. payoffs they must also explore the strategy space to estimate payoffs from observations. Hence, during learning, agents balance strategy space exploration and payoff exploitation.

In this thesis, we restrict the class of multi agent dynamic games to discounted stationary setting[16]. Here agents play a game for infinite timesteps, with exponentially discounted rewards and time-invariant strategies (policies), and learn the optimal strategy using multi-agent reinforcement learning. We now introduce notation for multi-agent reinforcement learning in this setting.

Notation

We assume a population of N_{agents} agents and denote the index of an agent in the population by $i, i \in \mathcal{N} = \{1, 2, \dots N_{agents}\}$. Each agent has a strategy that consists of some set of actions which depends on the state he is in. This state may change with time, depending on the states and actions of other agents. The state of agent i at time t is denoted by $x_t^{(i)}$. Agent i takes the action $a_t^{(i)}$ at time t. He may record observations $o_t^{(i)}$ at time t, which can be a general stochastic function of others' states and actions. In case agents are not allowed to observe other agents' states and actions, it reduces trivially to null set. We denote the collection of states

of all agents except i at time t by $\mathbf{x}_t^{(-i)}$, and corresponding actions by $\mathbf{a}_t^{(-i)}$. In a multi-agent setting, the state and transition rules depend on the states and actions of other agents, which we summarise as $\rho_t^{(-i)} = (\mathbf{x}_t^{(-i)}, \mathbf{a}_t^{(-i)})$, the empirical state-action distribution of all agents except i at time t. Each agent executes his policy conditioned on his state, observations and the population distribution. We denote the probability agent i of taking action $a_t^{(i)}$ given the current state of agent $x_t^{(i)}$, observations of others' $o_t^{(i)}$ when the rest of the population is in $\rho_t^{(-i)}$ by following the policy π_i as $\pi_i(a_t^{(i)}|x_t^{(i)}, o_t^{(i)}, \rho_t^{(-i)})$. Finally the rewards and transitions obtained by agent i at time t are written as $r_t^{(i)}(x_t^{(i)}, o_t^{(i)}, a_t^{(i)}, \rho_t^{(-i)})$ and $P_t^{(i)}(\cdot|x_t^{(i)}, o_t^{(i)}, a_t^{(i)}, \rho_t^{(-i)})$ respectively. Each agent discounts his reward with a discounting rate γ_i. Given the set of strategies of the entire population, each agent has a stationary state-observation distribution $\mu^{(i)}(x, o)$, corresponding to the stationary distribution defined by resulting Markovian dynamics. The payoff for each agent, is his long term discounted reward, which depends on his strategy and stationary state-observation distribution. Since his stationary state-observation distribution is determined completely by set of policies of all other agents and his current policy, we use $J^{(i)}(\pi^{(i)}, \pi^{(-i)})$ to denote payoff for agent Finally, we introduce the value functions $V_\pi^{(i)}$ and action value functions $Q_\pi^{(i)}$ for each agent below.

$$V^{(i)}_{\pi|\pi^{(-i)}}(x, o) = \sum_t \gamma_i^t r_t^{(i)}(x_t^{(i)}, o_t^{(i)}, a_t^{(i)}, \rho_t^{(-i)}) \text{ with } x_0^{(i)} = x, o_0^{(i)} = o$$

$$Q^{(i)}_{\pi|\pi^{(-i)}}(x, o, a) = \sum_t \gamma_i^t r_t^{(i)}(x_t^{(i)}, o_t^{(i)}, a_t^{(i)}, \rho_t^{(-i)}) \text{ with } x_0^{(i)} = x, o_0^{(i)} = o, a_0^{(i)} = a$$

Concepts

We now introduce the definition for Nash equilibrium for the system of agents and exploitability for each agent below. We assume that each agent plays to maximize his payoff.

Definition 2.1. *(Nash equilibrium)*
$\Pi_{Nash} = \{\pi^{(i)}\}_{i=1}^{N_{agents}} \quad s.t. \quad J^{(i)}(\pi^{(i)}, \boldsymbol{\pi}^{(-i)}) \geq J^{(i)}(\pi, \pi^{(-i)}) \quad \forall i \in \mathcal{N}, \pi \in \Pi_H$

This means that all agents have strategies such that no agent can improve upon his payoff by switching his strategy given that other agents fix their strategies.

A concept related to Nash equilibrium is the definition of exploitability, which we state below

Definition 2.2. *(Exploitability)*
$\phi_i(\pi_t^{(i)}, \boldsymbol{\pi}_t^{(-i)}) = \max_\pi J^{(i)}(\pi, \pi^{(-i)}) - J^{(i)}(\pi^{(i)}, \boldsymbol{\pi}^{(-i)})$

This quantifies the maximum improvement in payoff each agent can obtain if he switches strategies. At Nash equilibrium exploitability is zero for all agents.

So far, we have introduced the formalism of multi-agent dynamic games, and discussed solution concepts like Nash equilibria. An important feature in multi-agent systems is that the joint state action space grows exponentially with number of agents, and that the transition model for each agent changes if any other agent updates his policies, both of which make the analysis of multi-agent dynamic games difficult.

2.2 Mean Field Games

While analysing behaviour of finite population Multi-Agent dynamic games is difficult due to exponential growth of state-action space with population, a simplification can be made if one has indistinguishable agents, symmetric interactions between players and infinite population of agents. In this limit, any single agent has an infinitesimal effect on the rest of the population. Instead of solving the multi agent system as a collection of interacting agents in a complex environment, the problem can be reformulated as an interaction between a representative agent from the population

and an environment that is defined based on the mean field interactions between players. All agents are assumed to play the same strategy in this limit. The resulting solution of the strategy of the representative player is called the mean field solution.

It has been shown that in an N player game with indistinguishable agents, the mean field solution is δ Nash[7]. This means that if all players execute the strategy corresponding to the solution of the mean field game in the N agent game, then no player can have an exploitability of more than δ, with δ going to zero as N approaches infinity. A common technique in analysing convergence properties of Multi-Agent Games with finite but large population, with indistinguishable agents and symmetric interactions, is to analyse convergence of the infinite population Mean Field Game, and then get bounds on policy error during learning for the Multi-Agent RL problem in terms of the policies during learning of the Mean Field Game. Once this is established, one can also extend this to multi-type populations, where agents belonging to one type are indistinguishable.

We provide a brief introduction to the theory of mean field games below. For a more detailed survey of mean field games, we refer the reader to [8]

Setting of the general Mean Field Game

As mentioned before, a mean field game consists of a representative player from the population interacting with an environment that is defined based on the mean field interactions between players. Mean Field Games can be defined for static games, finite horizon dynamic games as well as infinite horizon discounted stationary game. We shall be focussing on the infinite horizon discounted stationary game, where the players play a time-invariant policy. We shall omit agents' observations of others states in our discussion, this can be trivially added by extending the definition of state space for the agent. We first provide the notation for the mean field game with discounted stationary setting.

- $\mathcal{X}$ and $\mathcal{A}$ denote the (finite) set of state and actions respectively. $\Delta_{\mathcal{X}}$ and $\Delta_{\mathcal{A}}$ denote sets of probability distributions over state and action spaces respectively.
- γ denotes discounting rate.
- $\pi(a|x)$ denotes the probability of choosing action a under current state x under the policy π.
- $\mu(x)$ denotes the mean state distribution for the population at any given time instant. Each agent in the population may be in a different state, but the population aggregate is in the state $\mu(x)$.
- $\rho(x, a)$ denotes the mean state-action distribution for the population. Since we assume that all agents play the same policy, $\rho(x, a) = \mu(x)\pi(a|x)$.
- $P(x'|x, a, \mu)$ denotes the probability of transitions to state x' when the representative agent is in state x, chooses action a and the population is in the mean state distribution μ.
- $r(x, a, \mu)$ denotes the reward when choosing action a in state x when population is in mean state distribution μ

Agent payoff when the agent plays the policy π and the population is in mean state μ can be now defined. Notice that in the discounted stationary setting, agent payoff depends on what his initial state distribution is. But since we have identical agents in the population, the agent has his initial state distribution equal to the mean state distribution of the population. Hence agent payoff reduces to:

$$J(\pi, \mu) = E_{x_0 \sim \mu}\{\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t, \mu)\} \tag{2.1}$$

In case of reward regularisation, this is modified to be

$$J(\pi, \mu) = E_{x_0 \sim \mu}\{\sum_{t=0}^{\infty} \gamma^t r(x_t, a_t, \mu) + h(\pi(x_t))\}$$

Another important point that can be immediately observed the non-stationary nature of the environment w.r.t. agent policies. If the representative agent changes his policy, then he ends up in a different stationary state distribution corresponding to the resulting transitions. But since we assume that all agents follow the same policy, this also changes the resulting mean-state distribution of the population once the population switches policies. Since the environment is defined based on the mean state distribution, any change in mean state distribution results in change in the environment (transitions and rewards). We state this mathematically below, but first we define another set of operators

Definition 2.3. *(Population Update Operator[18])*
$\Gamma_{pop}(\mu, \pi) : \Delta_{\mathcal{X}} \times \Delta_{\mathcal{A}\times\mathcal{X}} \to \Delta_{\mathcal{X}}$
$\Gamma_{pop}(\mu, \pi) = \sum_{x',a'} P(x|x', a', \mu)\mu(x')\pi(a'|x')$
$\Gamma^n_{pop}(\mu, \pi) = \Gamma_{pop}(\Gamma_{pop}(\dots \Gamma_{pop}(\mu, \pi), \pi), \pi)$

Definition 2.4. *(Stable population operator [18])*
$\Gamma^{\infty}_{pop}(\pi) = \lim_{n\to\infty} \Gamma^n_{pop}(\mu, \pi)$
This operator produces the stationary mean state distribution for a given policy.

Definition 2.5. *(Best Response policy)*
$\text{BR} : \Delta_{\mathcal{X}} \to \Delta_{\mathcal{X}\times\mathcal{A}}$
$BR(\mu) = \text{argmax}_{\pi'} J(\pi', \mu)$
This operator produces the optimal policy given a mean state distribution

Mean Field Games are solved as alternating updates between stationary distribution (mean field) for a given policy and best response policy given mean field, as stated below

$$\mu_t \to \pi_t \to \mu_{t+1} \to \pi_{t+1} \to \cdots \tag{2.2}$$

Here μ_{t+1} is obtained by stable population operator mapping from μ_t, π_t and π_{t+1} is obtained from a best response map from μ_{t+1}.

Solving for the Nash equilibrium in a mean field game reduces to the following set of updates. The Nash equilibrium is the fixed point of the updates.

$$\pi_* = BR(\mu_*) \text{ and } \mu_* = \Gamma^{\infty}_{pop}(\pi_*) \tag{2.3}$$

While the updates are defined above in terms of best response policies and full mean field solutions for the given policy, one can also define other updates, or include strategies like smoothing of updates, as long as the Nash equilibrium is still the fixed point of the update.

In case of finite population multi agent setting, if all agents are given policies of the Nash equilibrium as:

$$\pi^{(i)} = \pi_* \quad \forall i \in \{1, 2 \dots N\}$$

Then one has the following result[7] for exploitability, with $\delta \to 0$ as $N \to \infty$

$$\phi_i(\pi_*, \boldsymbol{\pi_*}^{(-i)}) \leq \delta \quad \forall i \in \{1, 2 \dots N\} \tag{2.4}$$

Hence, mean field games can be a useful technique to study multi-agent games with large, but finite, populations of identical agents with symmetric interactions between agents.

2.3 Karma games

A key challenge in mechanism design for resource allocation is ensuring fair outcomes that are also incentive compatible. Recently, a new class of mechanisms, called karma mechanisms were introduced. These are based on an artificial currency called karma, to facilitate bidding for resource access. Each karma game consists of a dynamic game with agents competing for access to a resource, with time varying utilities (urgencies) for accessing the resource. They bid for access using this currency (karma) and depending on the outcome of resource competition and karma mechanism, gain or lose karma. Karma games have been shown to be fair and efficient in resource allocation[3]. We now introduce the setup, notation and key concepts of a karma game below.

Setup

We assume a population of N_{agents} agents and denote the index of an agent in the population by $i, i \in \mathcal{N} = \{1, 2, \dots N_{agents}\}$. Each agent i has a time varying state, which consists of an urgency state $u_t^{(i)}$, karma $k_t^{(i)}$ and other supplementary states $\tilde{x_t}^{(i)}$. We assume that the urgency for agent i takes values in $u_t^{(i)} \in \mathcal{U}_i$, karma $k_t^{(i)} \in \mathbb{N}$, and supplementary state $\tilde{x_t}^{(i)} \in \tilde{\mathcal{X}}_i$. The urgency state denotes the urgency (utility) for the player to access the resource at that time instant, while the karma state denotes the available karma the agent has for bidding for access to the resource. The description of supplementary states depends on the problem at hand, e.g. in the karma game of [3], this supplementary state is the empty set, while in the traffic simulation experiment of Chapter 6 it can include details regarding the route agent is travelling in a city. The state of the agent at time t is the tuple

$$x_t^{(i)} = \left[u_t^{(i)}, k_t^{(i)}, \tilde{x_t}^{(i)}\right] \quad u_t^{(i)} \in \mathcal{U}_i,\ k_t^{(i)} \in \mathbb{N},\ \tilde{x_t}^{(i)} \in \tilde{\mathcal{X}}_i$$

The agents' urgencies and supplementary states evolve according to some Markovian process, that is exogeneous to their karma states. We denote the transition rule for this Markovian process by $P_i(u^+, \tilde{x}^+ \mid u, \tilde{x})$, which can be different for each agent. At each time instant the agent i picks an action $a_t^{(i)}$ from his action space. Similar to the state composition, we assume that the action for each agent is the tuple of his bids and a supplementary action state.

$$a_t^{(i)} = \left[b_t^{(i)}, \tilde{a_t}^{(i)}\right] \quad b_t^{(i)} \in \mathbb{N}, b_t^{(i)} \leq k_t^{(i)}\ \tilde{a_t}^{(i)} \in \tilde{\mathcal{A}}_i$$

In case of the setting from [3] the supplementary action state is the empty set, since an agent is only allowed to bid for a resource, while for other complex scenarios like bidding and path planning while travelling in a city, it can involve the set of paths the agents must pick from. Note that each agent is forbidden from bidding more than his karma at any given time.

At any time t, one can have multiple sets of players competing for access to a resource z. For each resource $l \in \mathcal{L}$, one has the set of competing players denoted by $\mathcal{C}_l(t)$. Each resource may have a finite capacity of players that can gain access to it denoted by $W(l)$. In the setting of [3], this capacity is equal to one, since only one player can access the resource, while in the setting of traffic simulation of 6 it is the capacity of the road lane. For allocating the resource, we use the following resource allocation rule

Resource Allocation Rule
Input: Sealed bids $b_t^{(j)}(l)$ of agents $j \in \mathcal{C}_l(t)$ for resource l
Output: Selected agents $i^*(l,t) \in \mathcal{C}_l(t)$ to access the resource
$i^*(l,t) = \arg\max_{j \subset [\mathcal{C}_l(t)]:|j|=k} b_t^{(j)}(l)$

Upon gaining access to a resource, we assume that agents get a reward equal to zero, while upon losing access we assume that agents get a penalty equal to their private urgencies. i.e.,

$$r_t^{(i)} = \begin{cases} 0 & \text{for } i \in i^*(l,t) \\ -u_t^{(i)} & \text{for } i \notin i^*(l,t) \end{cases}$$

Additionally, as mentioned previously, we assume agents play a discounted stationary game where agent i is assumed to discount future rewards with γ_i.

At the end of each resource allocation, we collect the bids from agents and deduct/compensate karma from the agents based on their bids. In this thesis, we will mainly study two karma payment rules: Pay-bid-to-Peer (PBP) and Pay-bid-to-Society (PBS). PBP is valid when only only has two agents competing for resource access, in which the winner pays karma equal to his bid to the loser. In case of PBS, the winner pays karma equal to his bid to the societal surplus, which will be later uniformly redistributed to all agents equally.

Apart from karma payment rules, one can also have karma redistribution rules in the population. During redistribution, the surplus collected is redistributed to the population at t^r redistribution times. Each agent i is given a karma equal to $k_r^{(i)}(t^r)$, at these redistribution timesteps. In this thesis, we assume that the surplus is redistributed at every timestep. Apart from this, other redistribution rules like taxation, as studied in [3], are also possible.

We now define metrics used to quantify a social outcome below.

Definition 2.6. *(Efficiency)*
$\mathit{eff} = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{C}_l(t)} \frac{r_t^{(i)}}{|\mathcal{C}_l(t)|}$

Definition 2.7. *(Ex-Post-Access-Fairness)*
$\mathit{af}_T = \text{-}\mathit{std}_{i\in\mathcal{N}} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{l\in\mathcal{L}} \mathbb{1}(i \in i^*(l,t))$

Definition 2.8. *(Ex-Post-Reward-Fairness)*
$\mathit{rf}_T = \text{-}\mathit{std}_{i\in\mathcal{N}} \frac{1}{T}\sum_{t=0}^{T-1} r_t^{(i)}$

Efficiency measures the average total utilities after resource allocation, where ex-post-access-fairness and ex-post-reward-fairness measure the standard deviation in access (winning a competition) and rewards across the population. The values of these depend on the allocation mechanism. Common allocation mechanisms that serve as benchmark for copmarison are DICT, COIN, TURN mechanism. In DICT mechanism, resources are allocated to the agents with highest truthful urgencies. In COIN, resources are allocated randomly according to a fair coin toss. In TURN mechanisms, agents take turns to access the resource by selecting the agent who has received the resource the least number of times in the past.

Karma mechanisms have previously been shown to achieve near maximal efficiency and fair outcomes. While we formulated the karma game above for distinct agents, with potentially different urgency transitions, discounting factors for each agent, one can also consider multi-type populations, where agents within each type are identical. Also, as described before, by adding supplementary state and action spaces in the formalism, one can apply karma mechanisms to a large class of resource allocation games.

Chapter 3

Multi Agent Reinforcement Learning in Karma Games

In this chapter we detail the algorithm which we use to compute the Nash equilibria in a general karma game setting. We provide a brief overview of the software which can be used in any general karma game to learn policies in a decentralized manner. The applications of this algorithm will be discussed in Chapter 6.

3.1 Algorithms for Multi Agent Reinforcement Learning

As mentioned in Chapter 2, we will be focussing on infinite horizon discounted stationary setting for the multi-agent system. We assume that agents' policies are unparametrized, i.e., agents do not use function approximation techniques for their policies but instead learn the numerical value for each state-action pair in a tabular setting. We assume the population is made of multiple types of agents, with all agents having identical characteristics within the type. In case of games with all agents being distinct, this trivially reduces to the setting where number of types is equal to the number of agents, with agent agent belonging to its own, different type.

In a multi-agent dynamic game, the model transitions and rewards depend on the policies and states of other agents. Hence, the learning dynamics are nonstationary for each agent. This might result in unstable learning, or no convergence at all, if the agents change their policies too drastically and frequently. To deal with this issue, we need to smoothen the policy update rules. Towards this, we implement a setup similar to [18], where agents agree on a predetermined time window and fix their policies for that duration. The agents then learn their next policies from the data gathered during this time window and then switch to the new policies for the next time window. This implies that agents learn their next behaviour based on the previous strategies of other players, but if the problem is smooth enough [1][18], then the above dynamics can lead to learning the correct Nash equilibrium. Furthermore, to allow further smoothing of the environment, we also allow probabilistic update of policies, i.e. players switch to the new policy with a certain probability otherwise they stick to their old policies. To learn optimal policies, each player must get an estimate of the value he may obtain by playing any action for his given state and observations. Towards this, we implement variants of tabular Q learning like greedy Q-Learning[16] and ϵ SARSA learning [16][18].

The resulting algorithm we get is a sequence of single agent tabular RL algorithms for each agent, and within each stage of the sequence, the player tries to learn an optimal policy, with the sequence of learned policies converging to the Nash policy in the case of successful execution of the learning process. Additionally, while we have discussed decentralized learning till now, one can also implement centralized learning, similar to the setting discussed in [18]. In this setting, one learns a policy for all agents in a population type. The only modification one has to make in terms of

formalism is that we use data from a predetermined agent within a population type, learn from its behaviour, and update all the agents of that type with the same learned policy. In case of a game with all agents being distinct, this reduces trivially to the fully decentralized learning setting. We now provide the formal algorithms to learn policies in a population playing a discounted stationary game, for both decentralized and centralized policy learning.

Algorithm Hyperparameters

We denote initial values of agent policies $\pi_0^{(i)}$, states $x_0^{(i)}$, observations $o_0^{(i)}$ and Q functions $Q_0^{(i)}$. For computing metrics to track convergence, we run a *monitor experiment*. Since this can be computationally costly, we only do this every few training epochs. The number of such experiments is $N_{monitor}$ and number of training epochs between two successive *monitor experiments* is $N_{epochs\ per\ monitor}$. As mentioned before, agents agree to fix their policies for a certain number of timesteps, denoted by $T_{iters\ per\ epoch}$. We assume that agents have bounded rationality of β, to encourage exploration, and agents update their policies using a softmax operation with β on their learned policies. Also, as mentioned before, we may not want all agents to update their policies at the end of every epoch. We control this by having a probabilistic update with probability p_{update} per agent. In case of centralised learning, we assume that all agents belonging to type ω have their indexes from the set $\mathcal{N}_\omega$. The representative agent for population type ω, whose data we shall use to compute optimal policies is denoted by n_ω. The total number of types is denoted by $|\Omega|$. In case of centralized learning, probabilistic updates happen for each population type rather than per agent as in decentralized learning.

Algorithms

Algorithm 1 Algorithm for decentralized MARL

Initialize agent policies $\pi_0^{(i)}$, states $x_0^{(i)}$, observations $o_0^{(i)}$ and Q functions $Q_0^{(i)}$
for $n_m = 1 : N_{monitor}$ **do**
  for $n_e = 1 : N_{epochs\ per\ monitor}$ **do**
    for $t = 1 : T_{iters\ per\ epoch}$ **do**
      Run a timestep by executing $\pi_{n_e}^{(i)}(x)\ \forall i \in \{1, 2, ...N_{agents}\}$
      for $i = 1 : N_{agents}$ **do**
        $\zeta_t^{(i)} \leftarrow \{x_t^{(i)}, o_t^{(i)}, a_t^{(i)}, r_t^{(i)}, x_t'^{(i)}, o_t'^{(i)}, a_t'^{(i)}\}$
        $\hat{Q}_{t+1}^{(i)} \leftarrow \hat{Q}_t^{(i)} - \alpha_t^{(i)} \tilde{F}^{(\pi_{n_e}^{(i)})}(\hat{Q}_t^{(i)}, \zeta_t^{(i)})$
      end for
    end for
    for $i = 1 : N_{agents}$ **do**
      Update $\pi_{n_e}^{(i)}(\cdot|x, o) = \begin{cases} \text{softmax}_a \beta Q_{t+1}^{(i)}(x, o, a), & \text{with probability } p_{update} \\ \pi_{n_e-1}^{(i)}(\cdot|x, o) & \text{otherwise} \end{cases}$
    end for
  end for
  Compute metrics to monitor convergence
end for

The algorithm is summarized in graphical form in Fig 3.1.

Algorithm 2 Algorithm for Centralized MARL

Initialize agent policies $\pi_0^{(i)}$, states $x_0^{(i)}$, observations $o_0^{(i)}$ and Q functions $Q_0^{(i)}$
for $n_m = 1 : N_{monitor}$ **do**
  for $n_e = 1 : N_{epochs\ per\ monitor}$ **do**
    for $t = 1 : T_{iters\ per\ epoch}$ **do**
      Run a timestep by executing $\pi_{n_e}^{(i)}(x)\ \forall i \in \{1, 2, ...N_{agents}\}$
      for $\omega = 1 : |\Omega|$ **do**
        $\zeta_t^{(\omega)} \leftarrow \{x_t^{(n_\omega)}, o_t^{(n_\omega)}, a_t^{(n_\omega)}, r_t^{(n_\omega)}, x_t'^{(n_\omega)}, o_t'^{(n_\omega)}, a_t'^{(n_\omega)}\}$
        $\hat{Q}_{t+1}^{(\omega)} \leftarrow \hat{Q}_t^{(\omega)} - \alpha_t^{(\omega)} \tilde{F}^{(\pi_{n_e}^{(n_\omega)})}(\hat{Q}_t^{(\omega)}, \zeta_t^{(\omega)})$
      end for
    end for
    for $\omega = 1 : |\Omega|$ **do**

$$\pi_{new}(\cdot|x,o) = \begin{cases} \text{softmax}_a \beta Q_{t+1}^{(\omega)}(x,o,a), & \text{with probability } p_{update} \\ \pi_{n_e-1}^{(n_\omega)}(\cdot|x,o) & \text{otherwise} \end{cases}$$

      for $j \in 1 : |\mathcal{N}_\omega|$ **do**
        Update $\pi_{ne}^{(j)} \leftarrow \pi_{new}$
      end for
    end for
  end for
  Compute metrics to monitor convergence
end for

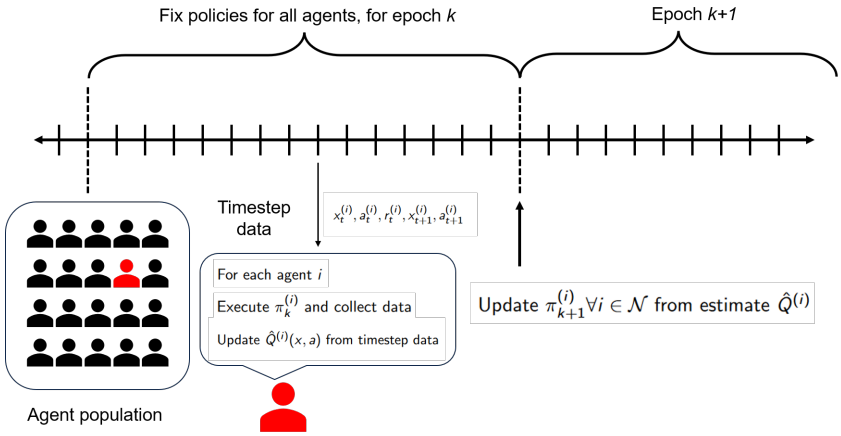


Figure 3.1: Schematic description of algorithm to learn multi-agent Nash equilibrium

Discussion

The difference between centralised and decentralised learning is that for centralized learning, we select a representative agent whose data we decide to collect to learn the policy per population type, and then assign the learned policy to all the agents within the population type. Note that apart from the iteration loop over timesteps, all other loops can be parallelised, including the Q function and Policy updates for all agents which can be computed in parallel if the agents agree to fix their policies for the epoch. The function for computing the updates of Q function values depends on the user. In case of CTD Learning as introduced in [18],

$$\tilde{F}^{\pi}(\hat{Q}, \zeta_t) = (Q(x,o,a) - r(x,o,a) - \gamma Q(x',o',a'))\mathbf{e}_{x,o,a}$$

In case of greedy Q learning, we define,

$$\tilde{G}^{\pi}(\hat{Q}, \zeta_t) = (Q(x, o, a) - r(x, o, a) - \gamma \max_{a'} Q(x', o', a'))\mathbf{e}_{x,o,a}$$

We postpone the discussion on computing metrics to track convergence during learning to next chapter.

3.2 Software overview

We now provide an overview of the Software package which can be used to compute Multi-Agent Nash equilibria for the general karma game setting. As seen in Fig 3.2, the user selects the karma game of interest. Depending on this selection a specific karma environment and agent controller are created, taking into account the constraints of the specific problem. Then the learning process is started. The learner has access to all agent attributes and can update policies learned from data. The learner requests a data buffer from the created environment. The environment interfaces with the agents and requests for actions and updates the states of the agents upon a timestep execution and passes the transition and reward data to the learner, which learns new policies as well as tracks metrics to measure convergence. Upon satisfactory performance, the process can be stopped or resumed on demand.

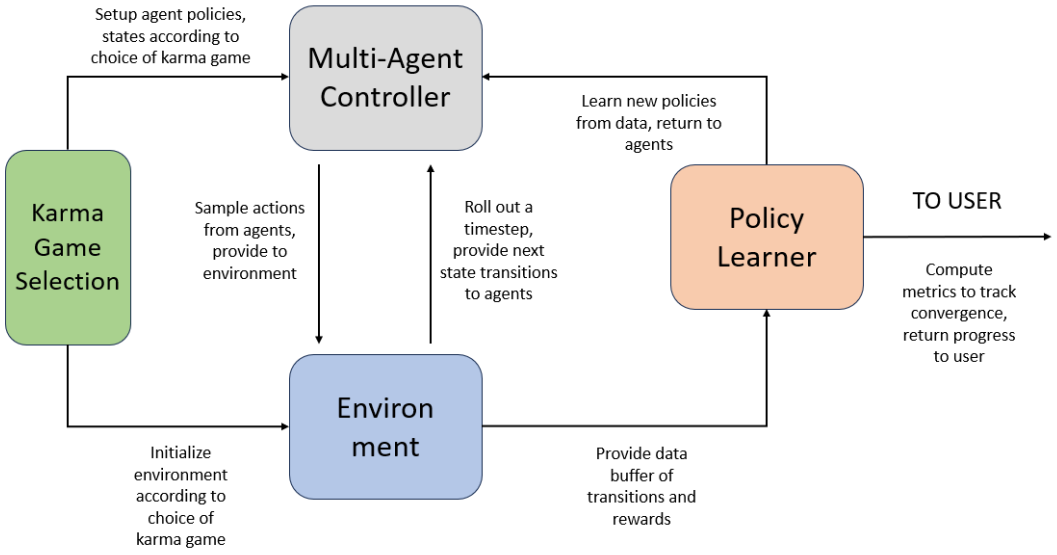


Figure 3.2: Qualitative Flowchart of Software

Chapter 4

Tracking convergence during MARL

In this Chapter we examine the use of different metrics to track convergence to Nash equilibrium during learning policies in the multi-agent game. We first discuss the use of exploitability as defined in Definition 2.2 to track learning progress and propose and evaluate a highly parallelized algorithm to compute exploitabilities for a population of agents. While tracking exploitability is a reliable metric to track convergence, its estimation is computationally expensive. We then motivate and examine the use of other metrics which can be used to track learning progress and state the conditions under which these can be used. Finally, we evaluate the reliability of these in a benchmark experiment where one knows the Nash equilibrium for a large population karma game.

4.1 Tracking convergence using exploitability

In a multi agent game, Nash equilibrium is the solution concept used to quantify whether the agents have converged to optimal behavior. At the Nash equilibrium, no player can find a better payoff by switching his strategy provided all other agents fix their strategies. One natural method to measure whether the agents are at the Nash equilibrium or not is to find how much can each agent gain by switching policies assuming others fix their policies, which is denoted by exploitability, as defined in Definition 2.2. In the ideal setting, this would be equal to zero for all agents if they are at the Nash equilibrium. In practice, agents can usually gain a small non-zero amount by switching due to stochasticity, finite compute-time in the learning algorithm. We repeat Definition 2.2 below, for the exploitability,

$$\phi_i(\pi, \boldsymbol{\pi}^{(-i)}) \equiv max_{\pi'} J^{(i)}(\pi'; \boldsymbol{\pi}^{(-i)}) - J^{(i)}(\pi; \boldsymbol{\pi}^{(-i)})$$

In Chapter 2.1, we noted that the agent payoff depends on his policy, the set of policies of the population as well as the stationary state distribution of the agent under the Markov process defined by the set of policies for all agents. Since the agents' stationary state distribution is fully specified by set of policies of all agents, we can write exploitability as:

$$\phi_i(\pi, \boldsymbol{\pi}^{(-i)}) \equiv max_{\pi'} J^{(i)}(\pi', \mu^{(i)}(\boldsymbol{\pi}); \boldsymbol{\pi}^{(-i)}) - J^{(i)}(\pi, \mu^{(i)}(\boldsymbol{\pi}); \boldsymbol{\pi}^{(-i)})$$

Here $\mu^{(i)}(\boldsymbol{\pi})$ is the stationary state distribution for the agent i under the set of agent policies $\boldsymbol{\pi} = \{\pi^{(i)}\boldsymbol{\pi}^{(-i)}\}$. We will use the shorthand $\mu^{(i)}$ for the same. To compute exploitability, one needs to obtain estimates for agent payoffs for the current policy as well as the greedy policy that obtains the maximum payoff. Towards this, we motivate the use of agents' value functions and action value functions in computing agent payoffs below.

$$J^{(i)}(\pi, \mu^{(i)}; \boldsymbol{\pi}^{(-i)}) \equiv E_{x\sim\mu^{(i)}}[V_{\pi|\pi^{(-i)}}(x) + h(\pi(x))] \tag{4.1}$$

Here we have added a regularisation term to the value function to compensate for agents having bounded rationlity to encourage exploration. Note that the agent payoff can be written as an

expectation of his value function under the initial state distribution being equal to his stationary state distribution. But, the value function itself can be written as an expectation over the action value function, when actions are sampled from the desired policy. This leads us to defining agent payoffs as:

$$J^{(i)}(\pi^{(i)} = \pi|\boldsymbol{\pi}^{(-i)}) \equiv E_{x\sim\mu^{(i)}}[V_{\pi|\pi^{(-i)}}(x)] = E_{x\sim\mu^{(i)}}\{h(\pi(x)) + E_{a\sim\pi(x)}Q_{\pi|\pi^{(-i)}}(x,a)\} \quad (4.2)$$

Using Eq 4.2, one can motivate the use of Markov Chain Monte Carlo (MCMC) techniques towards computation of exploitability. Firstly, we must artificially fix the policies of all agents except i in order to compute exploitability for agent i. We have run a new learning experiment for agent i in order to estimate his payoffs if the others fixed their policies. In order to compute agent payoffs for any given policy π, we need to compute the expectation of action value functions under the stationary distribution of the agent's current policy $\pi^{(i)}$. This can be approximated using Monte Carlo Markov Chain simulations. If the agent follows his policy for a long enough time, then his states can be assumed to be drawn from his stationary distribution. If he follows the policy π to pick actions, one can assume that his state-action pairs are drawn from the cartesian product of his stationary state distribution and policies under this stationary state distributions. Hence, to compute the agent payoff as in Eq 4.2, one could then simply sample from $Q_{\pi|\pi^{(-i)}}(x,a)$, in order to obtain estimates for his payoffs.

However, in this estimation of agent payoffs using MCMC, there will be an initial transient period before the samples are drawn from the desired state-action distribution. There are two sources of transients in this MCMC estimation. The first is the time taken for the estimates of Action Value functions $Q_{\pi|\pi^{(-i)}}(x,a)$ to converge. This is because during the learning process, the agent may not have estimated his action values accurately during the learning step, since during learning other agents are updating their policies. Additionally, in order to compute the exploitability, the agent also needs to estimate the action value distribution of a greedy policy, which he may not know beforehand and it may take time for the estimates of this action value function to converge to its true value. A second source of transients is that we need the agent's state distribution to converge to the stationary distribution induced by current policies. Once both these sources of transients have died out, then one can approximate the expected discounted reward from MCMC samples. The MCMC formulation of agent payoffs under any policy π is given as:

$$J^{(i)}(\pi^{(i)} = \pi|\boldsymbol{\pi}^{(-i)}) \equiv E_{x\sim\mu^{(i)}}[V^{(i)}_{\pi|\pi^{(-i)}}(x)+h(\pi(x))] \approx \frac{1}{T-T_{burn}}\sum_{t=T_{burn}}^{T} Q^{(i)}_{\pi|\pi^{(-i)}}(x_t^{(i)}, a_t^{(i)})+h(\pi(x_t)) \quad (4.3)$$

Here we have included a burn-in time T_{burn} to remove the effects of transients.

In the formulation described above, one has to execute the policy π to estimate action value functions and thus, agent payoffs. Since the exploitability involves payoffs of his current policy as well as his greedy policy given others fix their policies, one has to run three experiments: learn his payoffs under his current policy, learn a greedy policy and learn payoffs associated with greedy policy. However, one can also learn the action value functions using an exploring policy, to estimate action value functions in an off-policy manner [16]. Note that, in doing so, we also change the stationary state distribution from which we sample states in our MCMC estimation. Hence one has to run an additional experiment, once action value functions have converged, to sample states from the statinoary state distribution $\mu^{(i)}$, corresponding to his correct policy. This is described in Algorithm 4.

Another simplification can be made if one assumes that the current policies of agents have sufficient exploration. This assumption is not very restrictive since both in theory[18] and practice one needs to have sufficient exploration to learn optimal policies. Furthermore, since in a multi-agent game, if any agent changes his policy, the environment changes for other agents and so do

their optimal policies. Hence, agents must have a sufficient non-zero exploration to be able to adapt to environment changes. Under this assumption, one can use the agents' current policies to estimate their greedy payoffs in an off-policy manner given others' fix their policies. This implies that one can estimate exploitability for agent i by fixing policies of other agents and running an experiment where he executes his current policy for long enough time window to estimate his greedy exploitability as well. In principle one would repeat the procedure for all agents one by one to compute exploitabilies. But, note that since we can compute exploitability for each agent simply by letting them execute their current policies, we do not need to repeat the experiment for each agent. One can run a single, long experiment, where all agents fix their policies, which have sufficient exploration, and compute their individual exploitabilities in an off-policy manner without changing their policies. This is the principle behind design of Algorithm 3, which allows parallelized, single-shot, computation of exploitabilities for all agents. However, note that this requires the assumption that agents explore sufficiently.

Algorithm 3 Algorithm for Computing Exploitability in Parallel

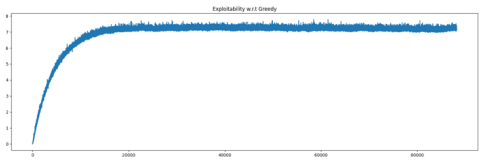
Given $\{\pi^{(i)}_{target,j}\}$, initial estimates of $Q_{\pi^{(i)}_{target,j}}$, $i \in \{1, 2 \dots N_{agents}\}, j \in \{1, 2, \dots, N_{targets}\}$
Given agent policies $\pi^{(i)}$, initial estimates of Q_{π_i} and greedy $Q^*_{\pi_i}, i \in \{1, 2 \dots N_{agents}\}$
for $t = 1 : T_{monitor}$ **do**
  Run a timestep by executing $\pi^{(i)}_{ne}(x)\ \forall i \in \{1, 2, ...N_{agents}\}$
  for $i = 1 : N_{agents}$ **do**
    $\zeta^{(i)}_t \leftarrow \{x^{(i)}_t, o^{(i)}_t, a^{(i)}_t, r^{(i)}_t, x'^{(i)}_t, o'^{(i)}_t, a'^{(i)}_t\}$
    $\hat{Q}^{(i)}_{t+1} \leftarrow \hat{Q}^{(i)}_t - \alpha^{(i)}_t \tilde{F}^{(\pi^{(i)})}(\hat{Q}^{(i)}_t, \zeta^{(i)}_t)$
    $\hat{Q^*}^{(i)}_{t+1} \leftarrow \hat{Q^*}^{(i)}_t - \alpha^{(i)}_t \tilde{G}^{(\pi^{(i)})}(\hat{Q^*}^{(i)}_t, \zeta^{(i)}_t)$
    Compute greedy exploitability sample
    $\phi_{greedy,t,i} = \max_{a'} \hat{Q^*}^{(i)}_{t+1}(x'^{(i)}_t, o'^{(i)}_t) - \hat{Q}^{(i)}_{t+1}(x'^{(i)}_t, o'^{(i)}_t, a'^{(i)}_t) + h(\pi^{(i)}(x'^{(i)}_t, o'^{(i)}_t))$
    for $j = 1 : N_{targets}$ **do**
      Sample action $a'_{j,t}{}^{(i)}$ for policy $\{\pi^{(i)}_{target,j}\}$, from state-observation $x^{(i)}_t, o^{(i)}_t$.
      $\zeta^{(i)}_{t,j} \leftarrow \{x^{(i)}_t, o^{(i)}_t, a^{(i)}_t, r^{(i)}_t, x'^{(i)}_t, o'^{(i)}_t, a'_{j,t}{}^{(i)}\}$
      $\hat{Q}^{(i)}_{j,t+1} \leftarrow \hat{Q}^{(i)}_{j,t} - \alpha^{(i)}_t \tilde{F}^{(\pi^{(i)}_{target,j})}(\hat{Q}^{(i)}_{j,t}, \zeta^{(i)}_t)$
      Compute exploitability w.r.t target policy sample
      $\phi_{target,t,i,j} = \hat{Q}^{(i)}_{j,t+1}(x'^{(i)}_t, o'^{(i)}_t, a'^{(i)}_t) - h(\pi^{(i)}(x'^{(i)}_t, o'^{(i)}_t)) - \hat{Q}^{(i)}_{t+1}(x'^{(i)}_t, o'^{(i)}_t, a'_{j,t}{}^{(i)}) - h(\pi^{(i)}_{target,j}(x'^{(i)}_t, o'^{(i)}_t))$
    end for
  end for
end for
for $i = 1 : N_{agents}$ **do**
  Compute greedy exploitability for agent i
  $\phi_{greedy,i} = \frac{1}{T_{monitor} - T_{burn}} \sum_t \phi_{greedy,t,i}$.
  for $j = 1 : N_{targets}$ **do**
    Compute exploitability of current policy of agent i w.r.t. target policy j
    $\phi_{target,i,j} = \frac{1}{T_{monitor} - T_{burn}} \sum_t \phi_{target,t,i,j}$
  end for
end for

Algorithm 4 Algorithm for Computing Exploitability one agent at a time

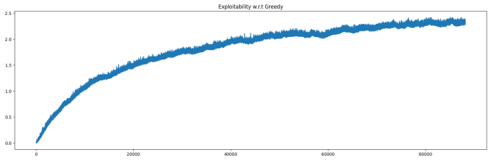
Given $\{\pi^{(i)}_{target,j}\}$, initial estimates of $Q_{\pi^{(i)}_{target,j}}$, $i \in \{1, 2 \dots N_{agents}\}, j \in \{1, 2, \dots, N_{targets}\}$
Given agent policies $\pi^{(i)}$, initial estimates of Q_{π_i} and greedy $Q^*_{\pi_i}, i \in \{1, 2 \dots N_{agents}\}$
for $i = 1 : N_{agents}$ **do**
  Change the policy of agent i while keeping others fixed.
  New policy has high exploration rate and is $\pi^{(i)}_{explore}$
  for $t = 1 : T_{monitor}$ **do**
    Run a timestep by executing $\pi^{(i)}_{n_e}(x)\ \forall i \in \{1, 2, ...N_{agents}\}$
    for $i = 1 : N_{agents}$ **do**
      $\zeta^{(i)}_t \leftarrow \{x^{(i)}_t, o^{(i)}_t, a^{(i)}_t, r^{(i)}_t, x'^{(i)}_t, o'^{(i)}_t, a'^{(i)}_t\}$
      Sample action ${a'_{t,agent}}^{(i)}$ for policy $\{\pi_{(i)}\}$, from state-observation $x^{(i)}_t, o^{(i)}_t$.
      $\zeta^{(i)}_{t,agent} \leftarrow \{x^{(i)}_t, o^{(i)}_t, a^{(i)}_t, r^{(i)}_t, x'^{(i)}_t, o'^{(i)}_t, {a'_{t,agent}}^{(i)}\}$
      $\hat{Q}^{(i)}_{t+1} \leftarrow \hat{Q}^{(i)}_t - \alpha^{(i)}_t \tilde{F}^{(\pi^{(i)})}(\hat{Q}^{(i)}_t, \zeta^{(i)}_{t,agent})$
      ${\hat{Q}^*}^{(i)}_{t+1} \leftarrow {\hat{Q}^*}^{(i)}_t - \alpha^{(i)}_t \tilde{G}^{(\pi^{(i)})}({\hat{Q}^*}^{(i)}_t, \zeta^{(i)}_t)$
      for $j = 1 : N_{targets}$ **do**
        Sample action ${a'_{j,t}}^{(i)}$ for policy $\{\pi^{(i)}_{target,j}\}$, from state-observation $x^{(i)}_t, o^{(i)}_t$.
        $\zeta^{(i)}_{t,j} \leftarrow \{x^{(i)}_t, o^{(i)}_t, a^{(i)}_t, r^{(i)}_t, x'^{(i)}_t, o'^{(i)}_t, {a'_{j,t}}^{(i)}\}$
        $\hat{Q}^{(i)}_{j,t+1} \leftarrow \hat{Q}^{(i)}_{j,t} - \alpha^{(i)}_t \tilde{F}^{(\pi^{(i)}_{target,j})}(\hat{Q}^{(i)}_{j,t}, \zeta^{(i)}_t)$
      end for
    end for
    Change the policy of agent i back to its original policy.
    for $t = 1 : T_{monitor}$ **do**
      Run a timestep by executing $\pi^{(i)}_{n_e}(x)\ \forall i \in \{1, 2, ...N_{agents}\}$
      Compute greedy exploitability sample
      $\phi_{greedy,t,i} = \max_{a'} {\hat{Q}^*}^{(i)}_{t+1}(x'^{(i)}_t, o'^{(i)}_t) - \hat{Q}^{(i)}_{t+1}(x'^{(i)}_t, o'^{(i)}_t, {a'_{t,agent}}^{(i)}) + h(\pi^{(i)}(x'^{(i)}_t, o'^{(i)}_t))$
      for $j = 1 : N_{targets}$ **do**
        Sample action ${a'_{j,t}}^{(i)}$ for policy $\{\pi^{(i)}_{target,j}\}$, from state-observation $x^{(i)}_t, o^{(i)}_t$.
        Compute exploitability w.r.t target policy sample
        $\phi_{target,t,i,j} = \hat{Q}^{(i)}_{j,t+1}(x'^{(i)}_t, o'^{(i)}_t, a'^{(i)}_t) - h(\pi^{(i)}(x'^{(i)}_t, o'^{(i)}_t)) - \hat{Q}^{(i)}_{t+1}(x'^{(i)}_t, o'^{(i)}_t, {a'_{j,t}}^{(i)}) - h(\pi^{(i)}_{target,j}(x'^{(i)}_t, o'^{(i)}_t))$
      end for
    end for
    Compute greedy exploitability for agent i
    $\phi_{greedy,i} = \frac{1}{T_{monitor} - T_{burn}} \sum_t \phi_{greedy,t,i}$.
    for $j = 1 : N_{targets}$ **do**
      Compute exploitability of current policy of agent i w.r.t. target policy j
      $\phi_{target,i,j} = \frac{1}{T_{monitor} - T_{burn}} \sum_t \phi_{target,t,i,j}$
    end for
  end for

We now show the numerical examples regarding convergence of exploitability estimates computed using MCMC. However, before one uses samples to compute an empirical mean, one needs to have a burn-in time for the transients involved in learning of action value functions as well as agent state distribution to approach its corresponding stationary distribution to die off. We plot the evolution of exploitability metrics during one monitor episode of Algorithm 3. To compute exploitabilities, we used $T_{samples} = 1000$ with a gap of 100 timesteps between samples for decorrelation. The plots displayed are values of mean exploitabilities over the population, with a coarse graining of time-axis by 100×. We display plots for exploitability at the beginning of training as well as at the end. Notice that towards the end of the training, the exploitability saturates to a lower value than at the beginning, indicating that agents have learned an approximate Nash equilibrium. This is

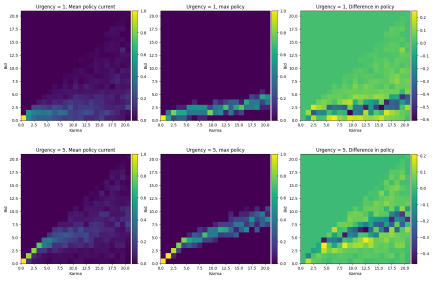
also visible in the policy plots for learned policies vs greedy policies towards the end of the training, where the greedy policies are not very different from learned policies.



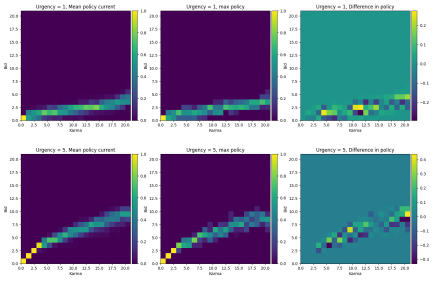
(a) MCMC sampling of greedy exploitability, at the beginning of training



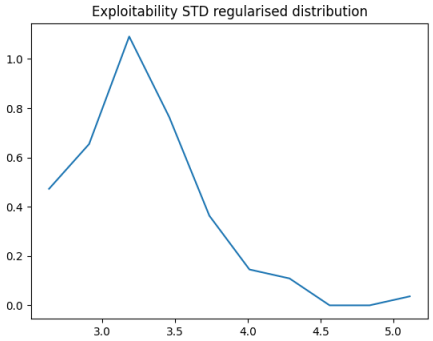
(b) MCMC sampling of greedy exploitability, at the end of training



(c) Difference in current and greedy policies at the beginning of training



(d) Difference in current and greedy policies at the end of training

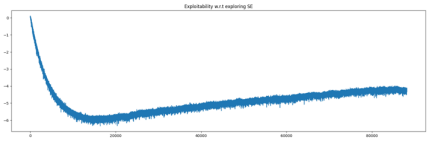


(e) Distribution of agent greedy exploitabilities at the end of training (unnormalized, normalization constant is 51.2) over a population of $N = 100$ agents

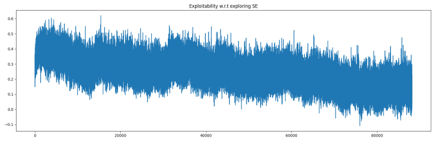
Figure 4.1: Greedy Exploitability calculation using Algorithm 3 and learned policies at the beginning and end of training

We also compute the exploitability w.r.t. stationary equilibrium policy (infinite population) as solved using methods from [3], as shown in Fig 4.2. Even in this case, notice that this exploitability at the beginning of the training saturates to a large negative value, and towards the end, the exploitability saturates to a small positive value. Note that we are in the finite multi-agent case, hence the policies are slightly different from the stationary equilibrium policy and do slightly better on average. Another important advantage of Algorithm 3 is that it allows parallelised computation of exploitabilities over the population. An example plot of distribution of exploitabilities over the population, at the end of training is displayed in Fig 4.1 e). An important point to note is that the exploitability plots report difference in agent payoffs and are not normalized. The values for

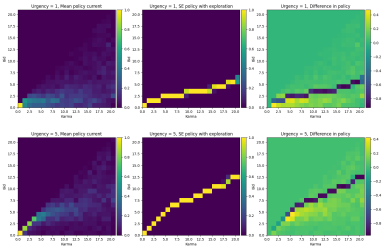
normalized exploitabilities, where one divides exploitability by payoff of current policy are in the next section as well as Chapter 6.



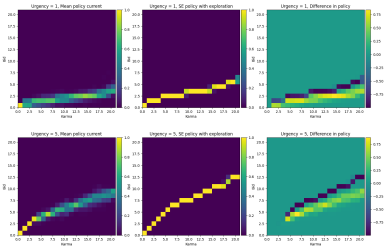
(a) MCMC sampling of exploitability w.r.t. MFG, at the beginning of training



(b) MCMC sampling of exploitability w.r.t. MFG, at the end of training



(c) Difference in current and MFG policies at the end of training



(d) Difference in current and MFG policies at the end of training

Figure 4.2: Exploitability w.r.t MFG calculation using Algorithm 3 and learned policies at the beginning and end of training

4.2 Tracking convergence using evolution of policies and payoffs

While a decreasing exploitability to zero for all agents indicates that the algorithm is converging, Monte Carlo Markov Chain simulation to obtain estimates is computationally expensive, sometimes more than the entire training scheme. Hence, we also examine the use of other metrics that are cheap to compute to track convergence.

Towards this, we focus on two main approaches. Firstly, we track maximal change in agent policies in successive epochs. If one assumes that agents have high enough learning rates, explore sufficiently, and try to learn optimal policies, then convergence of policies implies convergence to Nash equilibrium. If policies of all agents have converged, then the transition model for each agent has also stabilised. Now, if the agent is allowed to learn with high enough learning rate and sufficient exploration, and cannot find a policy different than his current one, then he has learnt the optimal policy given others have fixed their policies. Since the agent is learning policies to improve his payoffs, convergence of policies implies that he has achieved a near optimal payoff given others strategies, which is the definition of Nash equilibrium. Notice that the reverse is not true: a Nash equilibrium can have multiple policies with same payoffs, and not every Nash equilibrium will have policies that converge for all agents. But policies converging imply a Nash equilibrium, provided the above assumptions hold true. We use a distance measurement between policies before and after each training epoch for the agent. Towards measurement of distances between policies, one can use metrics like L1 norm. However, in the karma game, policies are smooth and the state action space is structured due to actions being defined as bids. Hence we use the Wasserstein distance to compare policies in the karma case, defined below. We assume that the action space available to the agent is only the bid space.

Definition 4.1. $d(\pi, \pi') \equiv \sum_{x,o} \frac{1}{b_{max}(x,o)} d_w(\pi(x, o, \cdot), \pi'(x, o, \cdot))$

Here $d_w(\cdot, \cdot)$ is the Wasserstein-1 distance between bid distributions at given state observation pair. The distance is normalized by dividing by $b_{max}(x, o)$, which is defined as the maximum bid the agent can take in state-observation pair x, o. This is done to underemphasise penalty different bidding strategies at high karma states, and penalise different bidding at low karma states. The normalization is to stretch/compress the bidding axis when computing the Wasserstein distance.

Apart from tracking maximum change in successive policy updates over the population, one can also measure maximum change in agent payoffs during learning. Similar to our discussion for exploitability, one can compute agent payoff by sampling from his current action value functions. Note that as the population of agents learn, their policies and action value functions will change. Furthermore, due to finite epoch length during training, their estimates of action value functions may not be the unbiased. But as the agents converge towards the Nash equilibrium, their action value functions will converge to the Nash values, and sampling will result in unbiased estimates of agent payoffs. Computation of agent payoff estimates using this procedure can be written as

$$\hat{J}_t^{(i)} = \frac{1}{T_p} \sum_{k=0}^{T_p} \hat{Q}^{(i)}_{t,\pi^{(i)}|\pi^{(-i)}}(x_k^{(i)}, a_k^{(i)}) + h(\pi^{(i)}(x_k)) \tag{4.4}$$

Here at learning timestep t, we run a small experiment of length T_p to get an estimate of agent payoffs following current policies. Since we only work with (imperfect) estimates action value function $\hat{Q}^{(i)}_{t,\pi^{(i)}|\pi^{(-i)}}$, rather than true action value function as in previous section, and also since we do not compute payoff of greedy policy, one does not need long run times for the estimation of agent payoffs. If the agents explore sufficiently, have high enough learning rates and if one chooses T_p correctly, then at Nash equilibrium the above payoff estimate is unbiased. Note that when compared to using changes in successive policies, one needs an additional parameter T_p which needs to be tuned for estimation of payoffs. One can then measure change in successive payoffs to track convergence, and when it is small for all agents, then we have likely converged to Nash. The advantage of this method over measuring policy changes is in games where there are multiple optimal policies at Nash equilibrium for an agent. If one tracks successive policy differences at Nash equilibrium of such games, they may not tend to zero, where measuring differences in payoffs will be a more reliable indicator of convergence.

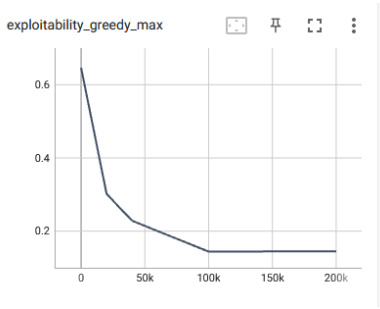
4.3 Summary of different metrics for convergence

We summarise the attributes of the different metrics examined above w.r.t. convergence guarantees and computational cost in Table 4.1. We evaluate the reliability of these in a benchmark experiment of a large population karma game. We measure and observe learning progress quantified using the above metrics. The setup of the karma game is the same as described in Chapter 6.1.

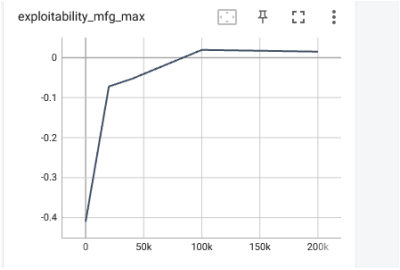
As the training progresses, notice that the greedy exploitability decreases from a large positive value to a small positive value at the end of training. This indicates that players have learnt an approximate Nash equilibrium. Additionally, in the plots for exploitability w.r.t. MFG policy, one observes that it also increases from a negative value in the beginning of the training to a small positive value at the end of training. This indicates that the policies learned using MARL are similar to the policies with centralized learning in the benchmark experiment, indicating that MARL scheme is learning the correct policies for the benchmark karma game. Alongside exploitability, we also plot the Wasserstein distances between policy updates for the agent (maximum over all agents) as well as change in payoff estimates (maximum over all agents). Notice that both of these decrease alongside exploitability, as the training progresses. This benchmark experiment demonstrates the use of alternative metrics to track convergence instead of exploitability in MARL, which will be useful in analysing convergence in more complex karma games where exploitability computation is not feasible.

Metric	Form	Guarantees	Compute cheap
Exploitability	$\max_i J^{(i)}(\pi \vert \pi_{-\mathbf{i}}) - J^{(i)}(\pi_i \vert \pi_{-\mathbf{i}}) \leq \delta$	✓	✗
Policy difference	$\max_i d(\pi_{i,t+1}, \pi_{i,t}) \leq \delta$	✓[a]	✓
Payoff difference	$\max_i \vert \hat{J}_t^{(i)} - \hat{J}_{t+1}^{(i)} \vert \leq \delta$	✓[a,b]	✓

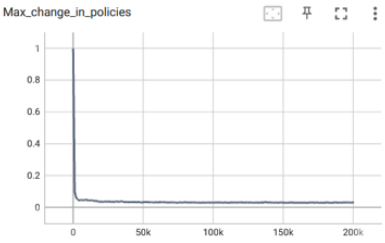
Table 4.1: Convergence metrics and properties
a: Assuming agents explore enough and have high enough learn rates
b: Requires additional tuning parameter



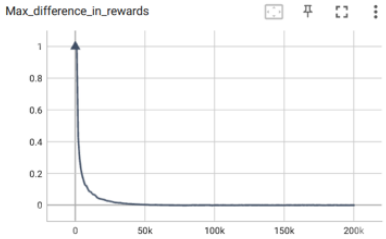
(a) Evolution of greedy exploitability (normalized) w.r.t. learned policies over the training process



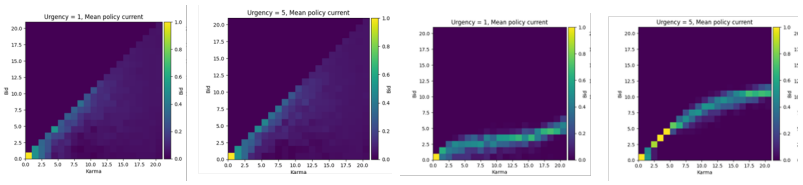
(b) Evolution of exploitability (normalized) of MARL policy w.r.t. MFG policies over the training process



(c) Evolution of maximum change in agent policies (measured with Wasserstein distance) over the training process



(d) Evolution of maximum change in agent rewards over the training process



(e) Policy of the mean agent at the start of the learning process

(f) Policy of the mean agent at the end of the learning process

Figure 4.3: Evolution of various convergence metrics over the training process

Chapter 5

Theoretical Analysis of policy convergence in Karma Games

In this chapter, we analyse karma games in large populations. We apply the mean field games formalism introduced in Chapter 2 to the analysis of karma games. We survey popular approaches towards proving convergence guarantees in mean field games and discuss their application in the karma games case. We conclude with a summary of results of convergence guarantees/violations when these approaches are applied in the karma games case and provide future outlook based on our observations.

5.1 Karma Mean Field Game

As mentioned in Chapter 2, multi-agent RL algorithms are difficult to analyse in general due to exponentially large state action space of agents. As seen before, their analysis can be simplified if one operates under the assumption of infinite and identical agents, which is called the mean field game setting. If one can obtain convergence guarantees for the corresponding mean field game of a multi-agent game, then one can obtain bounds on policy errors on learned policies for multi agent policy learning algorithms with finite but large population. We now study the mean field version of the karma game as introduced in [3]. While karma games can be extended to multiple other domains[4], for the sake of simplicity of analysis we study the basic game from [3]. As always, we consider infinite horizon discounted stationary setting. For the purpose of this discussion, we focus on populations of a single type of agents only.

Most popular techniques in proving convergence of policy updates in mean field games is to prove that the update scheme of Eq 2.2 is a contraction. A sufficient condition to prove this is to prove that each policy update step is a contraction. In mean field games of practical interest this is difficult to satisfy. One often tackles this by working with regularised mean field games, where there is an additional regularisation term in the reward and solves for Nash equilibrium of the regularised game, which has been done in [18][1]. Another set of techniques rely on exploiting structure of the game to get guarantees on rate of decrease of exploitability. Most of these rely on proving some form of monotonicity structure for the mean field game[13][12][19][6]. We formalise the karma mean field game below, and analyse policy convergence using contractivity based proofs and monotonicity based proofs for mean field games.

Setting

We assume that the state space of agents is the tuple of their private urgencies and karma. Each player undergoes urgency transitions as a Markov Process, independent of his karma. At each time instant, 2 agents from the population are randomly matched to compete for a resource. The

agent placing the higher bid wins the resource and incurs a zero cost while the loser incurs a cost equal to his urgency. The bids are redistributed to the population according to the karma mechanism. In this section, we only look at Pay-Bid-to-Peer (PBP) and Pay-Bid-to-Society (PBS) karma mechanisms. We state this setting in mathematical form below:

- $\mathcal{U}$: Urgency set for the population. At each time instant a player has an urgency $u \in \mathcal{U}$.
- $\phi(u^+|u)$: Urgency transition Markov process, exogenous of karma.
- γ: Discounting rate of agents.
- $k \in \{0, \dots k_{max}\}$ Agent karma, which can take a value between 0 and maximum allowed karma k_{max}.
- $x = (u, k)$, agent state, a tuple of karma and urgency.
- b: Agent bid for accessing the resource, $b \leq k$.
- $\mu(u, k)$: Mean field urgency-karma distribution of the population.
- $\pi(b|u, k)$: Probability of bidding b when in state tuple of urgency-karma (u, k) under policy π
- $\rho(u, k, b) = \mu(u, k)\pi(b|u, k)$: Mean state action distribution for the population
- $\nu(b) = \sum_{u,k} \rho(u, k, b)$: Mean bid distribution for the population
- $\mathcal{O} = \{0, 1\}$: Outcome set for a competition, $o = 0$ implies a win and $o = 1$ implies a loss
- $\theta[o|b, \nu]$: Probability of winning by bidding b when population mean bid distribution is ν
- $\kappa[k^+|k, b, \nu, o]$ Karma transition rule for an agent depending on his bid, population mean bid, his karma and outcome of resource competition. This transition matrix depends on the payment scheme used.

5.1.1 Contractivity based proofs

As mentioned previously, one of the ways of getting convergence guarantees in mean field games is to prove that Banach Picard iterations are contractive. If one has convergence guarantees for the mean field game case, one can also analyse convergence for the finite population Multi-Agent game as is done in [18][1].

Theory

We now apply the analysis of [18] to the previously mentioned karma model to try and obtain convergence guarantees. The paper deals with large population dynamic games with regularisation. The paper proposes the use of Policy Mirror Ascent operator in the Policy Update step of the Mean Field solver. The paper assumes that the rewards and state transitions are L1 Lipschitz continuous with some problem dependent Lipschitz constants. Using this, it establishes Lipschitz continuity for one iteration of policy update step of the mean field setting and obtains the Lipschitz constants for the Policy Mirror Ascent Operator. It then obtains conditions on the strength of regularisation required to ensure contractivity of the policy update step. After this, it studies convergence in decentralized and centralized multi agent games with identical agents, using the Mean Field Game as an approximation for the finite agent game. We begin our analysis by trying to obtain Lipschitz constants for transitions and rewards of the karma game, which will enable us to move on to finding Lipschitz constants for policy mirror ascent applied to the mean field karma game. Then one can obtain conditions on regularisation needed to ensure contractivity.

We need to find Lipschitz constants for transitions and rewards as defined below:

$$\begin{aligned}\|P(.\mid u,k,b,\rho)-P\left(.\mid u',k',b',\rho'\right)\|_1 \leq & k_\rho\left\|\rho-\rho'\right\|_1+k_u d\left(u,u'\right)\\ & +k_k d\left(k,k'\right)+k_b d\left(b,b'\right)\\ \left|R(u,k,b,\rho)-R\left(u',k',b'\rho'\right)\right| \leq & L_\rho\left\|\rho-\rho'\right\|_1+L_u d\left(u,u'\right)\\ & +L_k d\left(k,k'\right)+L_b dz\left(b,b'\right)\end{aligned}$$

In the karma setting, the transitions and rewards depend on the mean state-action distribution via the mean action distribution ν. Thus, we first find Lipschitz constants in terms of ν and convert them to ρ, which is the setting from the paper

$$\begin{aligned}\|P(.\mid u,k,b,\nu)-P\left(.\mid u',k',b',\nu'\right)\|_1 \leq & k_\nu\left\|\nu-\nu'\right\|_1+k_u d\left(u,u'\right)\\ & +k_k d\left(k,k'\right)+k_b d\left(b,b'\right)\\ \left|R(u,k,b,\nu)-R\left(u',k',b'\nu'\right)\right| \leq & L_\nu\left\|\nu-\nu'\right\|_1+L_u d\left(u,u'\right)\\ & +L_k d\left(k,k'\right)+L_b dz\left(b,b'\right)\end{aligned}$$

The relation between mean state action distribution and mean bid distribution is as follows

$$\sum_{u,k}\rho(u,k,b)=\nu(b)$$

Hence using this, one can establish that

$$\begin{aligned}\|\nu(b)-\nu'(b)\|_1 &= \left\|\sum_{u,k}\rho(u,k,b)-\sum_{u,t}\rho'(u,k,b)\right\|_1\\ &= \sum_b|\sum_{u,k}\left(\rho(u,k,b)-\rho'(u,k,b)\right|\\ &\leq \sum_{b,u,k}|\rho(u,k,b)-\rho'(u,k,b)|\\ &= \|\rho-\rho'\|_1\\ \therefore\|\nu-\nu'\|_1 &\leq \|\rho-\rho'\|_1\end{aligned}$$

Thus one has $k_\nu=k_\rho, L_\nu=L_\rho$.

We define distances for urgency space, karma space and bid space as follows

$d(u,u')=\delta(u\neq u');\ d(k,k')=\delta(k\neq k');\ d(b,b')=\delta(b\neq b')$

Here δ is the indicator function.

Lipschitz constants for rewards

Note that rewards are defined in terms of losing probabilities against the population as follows

$$\begin{aligned}R(u,k,b,\nu) &= -u\theta[1\mid b,\nu]\\ &= -u\left(\sum_{b'>b}\nu\left[b'\right]+0.5\nu[b]\right)\end{aligned}$$

Difference in rewards under different urgencies, rewards, bid distributions and karmas is as follows:

$$\begin{aligned}\left|R(u,k,b,\nu)-R\left(u',k',b',\nu'\right)\right| \leq & \left|R(u,k,b,\nu)-R\left(u,k,b',\nu'\right)\right|+\left|R(u,k,b,\nu')-R\left(u',k,b,\nu'\right)\right|\\ & +\left|R(u',k,b,\nu')-R\left(u',k',b,\nu'\right)\right|\end{aligned}$$

From the definition of reward by winning probability, we have:

$$
\begin{aligned}
&|R(u',k,b,\nu') - R\left(u',k',b,\nu'\right)| = 0 \\
&|R(u,k,b,\nu') - R\left(u',k,b,\nu'\right)| = \left| \left(u-u'\right)\left(\sum_{b'>b}\nu'\left[b'\right] + 0.5\nu'[b]\right)\right| \leq u_{max} d(u,u') \\
&|R(u,k,b,\nu) - R\left(u,k,b',\nu'\right)| \leq u_{\max}\left|\sum_{\tilde{b}>b}\nu[\tilde{b}] + 0.5\nu[b] - \sum_{\tilde{b}'>b'}\nu'\left[\tilde{b}'\right] + 0.5\nu'\left[b'\right]\right| \\
&\qquad\qquad \leq u_{\max}\left(\delta\left(b,b'\right) + \left\|v-v'\right\|_1\right)
\end{aligned}
$$

$$
\boxed{\therefore L_u = u_{\max};\ L_k = 0\ L_b = u_{\max};\ L_\nu = L_b = u_{\max}} \tag{5.1}
$$

Lipchitz continuity of state transitions

We write the state transition rule as follows

$$
P\left(u^+,k^+ \mid u,k,b,\nu\right) = \phi\left(u^+ \mid u\right)\sum_o \theta[o \mid b,\nu]\kappa[k^+ \mid k,b,o,\nu]
$$

$$
\theta[o \mid b,\nu] = \begin{cases} \sum_{b'<b} v\left(b'\right) + 0.5\nu(b) & \text{for } o=0 \\ \sum_{b'>b}\nu\left(b'\right) + 0.5\nu(b) & \text{for } o=1 \end{cases}
$$

Hence the difference in transition vectors becomes

$$
\begin{aligned}
&\left\|P(\cdot \mid u,k,b,v) - P\left(\cdot \mid u',k',b',v'\right)\right\|_1 \\
&= \left\|\phi(. \mid u)\left(\sum_o \gamma[o \mid b,\nu]\kappa[k \mid k,b,o,\nu]\right) - \phi\left(. \mid u'\right)\left(\sum_{o'}\theta[o|b'\cdot\nu']\kappa[k \mid k',b',o',\nu']\right)\right\|_1 \\
&\underbrace{\leq \left\|\phi(.\|u) - \phi(.|u')\right\|_1}_{A} + \underbrace{\left\|\sum_o\left(\theta[o \mid b,\nu]\kappa[o \mid k,b,o,\nu] - \theta[o \mid b',\nu']\kappa[o \mid k',b',o,\nu']\right)\right\|_1}_{B}
\end{aligned}
$$

$A \leq 2\delta\left(u \neq u'\right)$ (Difference in probability vectors)

$$
B \leq \underbrace{\sum_o \left\|\theta\left[o \mid b,k,\nu\right] - \theta\left[o \mid b',k',\nu'\right]\right\|}_{C} + \underbrace{\left\|\kappa\left[. \mid k,b,\nu,o\right] - \kappa\left[. \mid k',b',\nu',o\right]\right\|_1}_{D}
$$

$$
\begin{aligned}
C &= \sum_o \left\|\theta[o \mid b,\nu] - \theta\left[o \mid b',\nu'\right]\right\|_1 \\
&= \sum_o \left\|\theta[o \mid b,\nu] - \theta\left[o \mid b',\nu\right] + \sum_o \left\|\theta\left[o \mid b',\nu\right] - \theta\left[o \mid b',\nu'\right]\right\|_1 \right. \\
&\leq \sum_o \underbrace{\left\|\theta[o \mid b,\nu] - \theta\left[o \mid b',\nu\right]\right\|_1}_{\leq\delta[b,b']} + \sum_o \left\|\theta\left[o \mid b',\nu\right] - \theta\left[o \mid b',\nu'\right]\right\|_1
\end{aligned}
$$

$$
\sum_o \left\|\theta\left[o \mid b',\nu\right] - \theta\left[o \mid b',\nu'\right]\right\|_1. = \left\|\sum_{b<b'} v[b] - \nu'[b] + 0.5\left(\nu\left[b'\right] - \nu'\left[b'\right]\right)\right\|_1 + \left\|\sum_{b>b'}\nu[b] - \nu'[b] + 0.5\left(0\left[b'\right] - \nu'\left[b'\right]\right)\right\|_1
$$

$$
\leq \left\|\nu - \nu'\right\|.
$$

$$
\therefore C = \quad \sum_o \left\|\theta[o \mid b,\nu] - \theta\left[o \mid b',\nu'\right]\right\|_1 \leq 2\delta\left[b,b'\right] + \left\|\nu - \nu'\right\|.
$$

Let

$$D = \sum_o \|\kappa[\cdot \mid \nu, b, k, o] - \kappa\left[\cdot \mid \nu', b', k', o\right]\|_{1\cdot} = K_\nu \| \nu - \nu' \|_1 + T_b(b, b') + T_u d(u, u') + T_k d(k, k')$$

$$\therefore \|P(\cdot \mid u, k, b, v) - P\left(\cdot \mid u', k', b', v'\right)\|_1 = K_b d\left(b, b'\right) + K_s d\left(s, s'\right) + K_\nu \left\|\nu - \nu'\right\|_1 + K_u d(u, u').$$

WLOG $K_b, K_s = 2$ (From [18]Assumption 1)

$$\begin{aligned}\|P(. \mid \nu, b, k) - P\left(. \mid \nu', b, k\right)\|_1 &\leq \sum_o \|\theta[o \mid b, v] - \theta[o \mid b.v]\|_1 \quad + \sum_o \|\kappa[\cdot 1v, b, k, o] - \kappa\left[\cdot 1v' \cdot b, k, o\right]\| \\ &= \left(k_\nu + 1\right)\left\|\nu - \nu'\right\|_1\end{aligned}$$

i.e. The transition of states is Lipchitz continuous if Karma transitions are Lipschitz continuous

We now try to obtain Lipschitz constants for PBP and PBS

PBP:

The karma transitions can be written as follows:

$$P(. \mid b, k, \nu) = \begin{cases} \sum_{b'<b} \nu\left[b'\right] + 0.5\nu[b] & k^+ = k - b \\ 0.5\nu[b] & k^+ = k + b \\ \nu\left[b'\right] & \forall b' > b, k^+ = k + b' \end{cases}$$

Firstly, we try to guess Lipschitz constants for differing bids as follows.

$$\| \left(P(. \mid b, k, \nu) - P(. \mid b', k, \nu)\right) \|_1 \leq 2\delta\left[b \neq b'\right]$$

Towards this, consider ν s.t.

$$\begin{aligned} \nu[b] = \nu\left[b'\right] = 0 \text{ and } \sum_{\tilde{b}<b} \nu[\tilde{b}] = 1 = \sum_{\tilde{b}<b'} \nu[\tilde{b}] \\ \therefore \quad P(\cdot \mid b, k, \nu) = 1 \text{ for } k^+ = k - b \\ P\left(. \mid b', k, \nu\right) = 1 \text{ for } k^+ = k - b' \end{aligned}$$

Similarly, one can obtain Lipschitz continuity for differing karma as follows:

$$\|P(\cdot \mid b, k, \nu) - P\left(\cdot \mid b, k', \nu\right)\|_1 \leq 2\delta\left[k \neq k'\right]$$

Now we obtain Lipschitz constants for differing bid distributions as follows:

$$\begin{aligned}\|P(. \mid k, b, \nu) - P\left(\cdot \mid k, b, \nu'\right)\|_1 &= \left|\sum_{b'<b} \left(\nu\left[b'\right] - \nu'\left[b'\right]\right) + \left(0.5\nu[b] - 0.5\nu'[b]\right)\right| \\ &\quad + 0.5\left|\nu[b] - \nu'[b]\right| + \sum_{b'>b} \left|\nu[b] - \nu'[b]\right| \\ &\leq \sum_b \left|\nu[b] - \nu'[b]\right| = \|\nu - \nu'\|_1\end{aligned}$$

This results in

$$\boxed{K_\nu = K_\rho = 1, L_\nu = L_\rho = u_{max}, K_u = K_s = K_k = 2, K_b = 2, L_u = L_s = u_{max}} \tag{5.2}$$

PBS

Before we write the transition rules in PBS, we recap the redistribution scheme from [3]. At each instant of time, the surplus is redistributed equally to all players. For a mean bid distribution of ν, the surplus is as follows:

$$\begin{aligned}\bar{p} &= \sum_{u,k,b} \rho(u,k,b)\theta[o=0|b]b \\ &= \sum_{b} b\nu(b)\left(\sum_{b'<b}\nu(b') + 0.5\nu(b)\right)\end{aligned}$$

This is redistributed using integer preserving redistribution rule.

- Distribute $\lfloor\bar{p}\rfloor$ to $f^{low} = \lceil\bar{p}\rceil - \bar{p}$ fraction of population
- Distribute $\lceil\bar{p}\rceil$ to $f^{high} = \bar{p} - \lfloor\bar{p}\rfloor$ fraction of population

We can now write the karma transition rules for PBS as follows:

$$P(.\mid b,k,v) = \begin{cases} f^{loww}\left(\sum_{b^2<b} v\left[b'\right] + 0.5v[b]\right); k^+ = k - b + \lfloor\bar{p}\rfloor \\ f^{\text{high}}\left(\sum_{b'<b} v\left[b'\right] + 0.5v[b]\right); k^+ = k - b + \lceil\bar{p}\rceil \\ f^{low}\left(\sum_{b'>b} v\left[b'\right] + 0.5v[b]\right); k^+ = k + \lfloor\bar{p}\rfloor \\ f^{\text{high}}\left(\sum_{b'>b} v\left[b'\right] + 0.5v[b]\right); k^+ = k + \lceil\bar{p}\rceil \end{cases}$$

Similar to PBP, one can show that

$$\begin{aligned}\left\|P(.\mid b,k,v) - P\left(\cdot\mid b',k,v\right)\right\|_1 &\leq 2\delta\left[b\neq b'\right] \\ \left\|P(\cdot\mid b,k,v) - P\left(\cdot\mid b,k',v\right)\right\|_1 &\leq 2\delta\left[k\neq k'\right]\end{aligned}$$

To find Lipchitz constants for different d bid distributions

Let $\bar{p}(\nu)$ denote average redistribution under ν.

$\left\|P(\cdot\mid b,k,\nu) - P\left(\cdot\mid b,k,\nu'\right)\right\|_1$ is easier to analyse in terms of $\bar{p}(\nu)$ and $\bar{p}\left(\nu'\right)$.

Case 1:

$$\nu,\nu' \text{ s.t. } \left|\bar{p}(\nu) - \bar{p}\left(\nu'\right)\right| > 2$$

Then clearly.

$$\left\|P(\cdot\mid b,k,\nu) - P\left(\cdot\mid b,k,\nu'\right)\right\|_1 = 2.$$

Since $2 < 2\left|\bar{p}(\nu) - \bar{p}\left(\nu'\right)\right|$

$$\left\|p(\cdot\mid k,b,\nu) - p\left(\cdot\left|k,b,\nu'\right|\right\|_1 \leq \left\|\nu-\nu'\right\|_1 + 2\left|\bar{p}(\nu) - \bar{p}\left(\nu'\right)\right|$$

Case 2:

$$\nu,\nu' \text{ s.t. } \left|\left\lfloor p\left(\bar{\nu}\right)\right\rfloor - \left\lfloor p\left(\bar{\nu'}\right)\right\rfloor\right| = 1$$

WLOG assume that $\lfloor \bar{p}(\nu) \rfloor = \lfloor \bar{p}\,(\nu') \rfloor - 1$

$$\begin{aligned}
\|P(\cdot \mid k,b,\nu) - P\,(\cdot \mid k,b,\nu')\|_1 =& \mid f^{low}\,(\nu')\,[0 \mid b,\nu] - f^{high}(\nu)\theta[0 \mid b,\nu] \mid + f^{low}(\nu)\theta[0 \mid b,\nu] \\
&+ f^{\text{high}}\ (\nu')\,\theta[0|b,\nu] + \left|f^{\text{low}}\ (\nu')\,\theta\,[1|b,\nu'] - f^{\text{high}}\,(\nu)\theta[1 \mid b,\nu]\right| \\
&+ f^{\text{low}}\,(\nu)\theta[1 \mid b,\nu] + f^{high}\,(\nu')\,\theta[1|b,\nu] \\
&= f^{low}(\nu) + f^{high}\,(\nu') + f^{\text{low}}\ (\nu')\,|\theta\,[0 \mid b,\nu'] - \theta[0 \mid b,\nu]| \\
&+ \left|f^{low}(\nu) - f^{high}\,(\nu')\right|\theta[0 \mid b,\nu] + \left|f^{low}(\nu) - f^{high}\,(\nu')\right|\theta[1 \mid b,\nu] \\
&+ f^{low}\,(\nu')\,|\theta\,[1 \mid b,\nu'] - \gamma[0 \mid b,\nu]| \\
&= f^{low}(\nu) + f^{high}\,(\nu') + f^{low}\,(\nu')\sum_o |\theta[o|b,v] - \theta\,[o \mid b,v']| \\
&+ \left|f^{low}\,(\nu') - f^{high}(\nu)\right|
\end{aligned}$$

From C, we have $\sum_o |\theta\,[o\,|,\nu] - \theta\,[o \mid b,\nu']| = \|\nu - \nu'\|_1$.

$$\begin{aligned}
\therefore \|P(\cdot \mid k,b,\nu) - P\,(\cdot \mid k,b,\nu')\|_1 &= f^{low}\,(\nu')\,\|\nu - \nu' \mid + f^{low}(\nu) + f^{high}\,(\nu') \\
&= \left|f^{low}\,(\nu') - f^{high}(\nu)\right| \\
&\leq |\nu - \nu'\|_1 + 2|\,\hat{p}(\nu) - \bar{p}\,(\nu')\mid
\end{aligned}$$

Case 3:

$$\nu, \nu' \text{ s.t. } \mid \left\lfloor p\,(\bar{\nu})\right\rfloor - \left\lfloor p\,(\bar{\nu'})\right\rfloor\mid = 0$$

$$\begin{aligned}
\|P(\cdot \mid k,b,\nu) - P\,(\cdot \mid k,b,\nu')\|_1 &= \sum_o |\theta[o|b,\nu] f_{low}(\nu) - \theta\,[o|b,\nu']\,f_{low}\,(\nu')| \\
&+ \sum_o |\theta[o \mid b,\nu] f_{low}(\nu) = \theta\,[o \mid b,v']\,f_{low}\,(\nu')| \\
&= f^{low}(\nu)\sum_o \|\theta[o \mid b,\nu] - \theta[o \mid b,\nu]\|_1 \\
&+ f_{high}(\nu)\sum_o \|\theta[o \mid b,\nu] - \theta[0 \mid b,\nu]\|_1 \\
&+ \sum_o \theta\,[o \mid b,\nu']\left|f^{low}(\nu) - f^{low}\,(\nu')\right| \\
&+ \sum_o \theta\,[o|b,\nu']\left|f^{high}(\nu) - f^{high}\,(\nu')\right| \\
&= \sum_o \|\theta[o|b,\nu] - \theta\,[o\|b,\nu']\|_1 + \left|f^{low}(\nu) - f^{low}\,(\nu')\right| \\
&+ \left|f^{high}(\nu) - f^{high}\,(\nu')\right| \\
&\leq \|\nu - \nu'\|_1 + |\lceil \bar{p}\,(\nu')\rceil - \lceil \bar{p}(\nu)\rceil + \bar{p}(\nu) - \bar{p}\,(\nu')| \\
&+ |\lfloor \bar{p}(\nu)\rfloor - \lfloor \bar{p}\,(\nu')\rfloor + \bar{p}\,(\nu') - \bar{p}(\nu)| \\
&\leq \|\nu - \nu'\|_1 + 2\,|\bar{p}(\nu) - \bar{p}\,(\nu')|
\end{aligned}$$

Combining all three cases we can write:

$$\boxed{\|P(\cdot \mid k,b,\nu) - P\,(\cdot \mid k,b,\nu')\|_1 \leq \|\nu - \nu'\|_1 + 2\,|\bar{p}(\nu) - \bar{p}\,(\nu')|} \tag{5.3}$$

Furthermore

$$|\bar{p}(\nu) - \bar{p}(\nu')| \leq b_{max} \|\nu - \nu'\|_1 = k_{max} \|\nu - \nu'\|_1$$

$$\therefore K_\nu = K_\rho = 2k_{max} + 1 \quad \text{forPBS} \tag{5.4}$$

Convergence conditions

For convergence of Policy Mirror Ascent, we need $L_n < 1$. where $\Gamma_n : \pi \to \pi$ maps current policy to next policy s.t. new policy improves reward for the induced mean field. $L_{r_n} \leqslant \frac{L_{rq}}{\rho}$ where ρ is concavity of regularizer For entropy regularisation ρ is equal to inverse bounded rationality, or inverse temperature for the softmax in policy update from learned Q.

$$\begin{aligned}
L_{\Gamma,q} &= L_{pop,\infty} L_{q,\rho} + L_{q,\pi} \\
L_{q,\rho} &= L_\rho + \gamma L_{v,\rho} + \gamma \frac{L_{v,s} K_\rho}{2} \\
L_{q,\pi} &= \gamma L_{v,\pi} + \gamma L_{v,s} K_a \\
L_{v,s} &= \frac{L_s + L_a + \Delta h}{1 - \gamma \min\left(1, \frac{k_s + k_a}{2}\right)} \\
L_{v,\pi} &= \frac{4L_a + \gamma K_a L_{v_s}}{4(1-\gamma)} \quad L_{v,\rho} = \frac{2L_\rho + \gamma K_\rho L_{v_s}}{2(1-\gamma)}
\end{aligned}$$

The Lipschitz constant for Policy Mirror Ascent depends on the allowed range of entropies from the maximum Δh as well as the Lipschitz constant for the stable population update operator $L_{pop,\infty}$. We have not calculated $L_{ppp,\infty}$ so far. But certainly $L_{pop,\infty} > 0$ and $\Delta h \geq 0$. The minimum value of $L_{\Gamma,\eta}$ is achieved for $L_{pip,\infty} = 0$ and $\Delta h = 0$. We show that even with this, the resulting Lipschitz constant requires oversmoothing in case of both PBP and PBS schemes.

For PBP, $K_\rho = 1, L_\rho = u_{\max} K_s, K_a = 2, L_s = u_{\max}$

$$\begin{aligned}
L_b &= u_{\max} \\
Lv, \pi &= \frac{4u_{\max} + \gamma 2 L_{v_s}}{4(1-\gamma)} \quad L_{v,\mu} = \frac{2u_{\max} + \gamma L_{vs}}{2(1-\gamma)} \\
L_{v,s} &= \frac{u_{\max} + u_{\max} + \Delta h}{1-\gamma}
\end{aligned}$$

This For $u_{\max} = 1$ (normalised rewards), $\Delta h = 0, \gamma = 0.9$ this results in $L_{\Gamma_n} \approx 126$, implying temperature of softmax must be atleast 126, which is much larger than the typical values used in experiments. Using high temperatures means oversmoothing of the policies and it is obvious that the regularised games with such high regularisation will converge, not yielding anything substantial of interest.

For PBS, the constant K_ρ as calculated previously, scales with the karma in the system, making it even worse than PBP when viewed from this analysis.

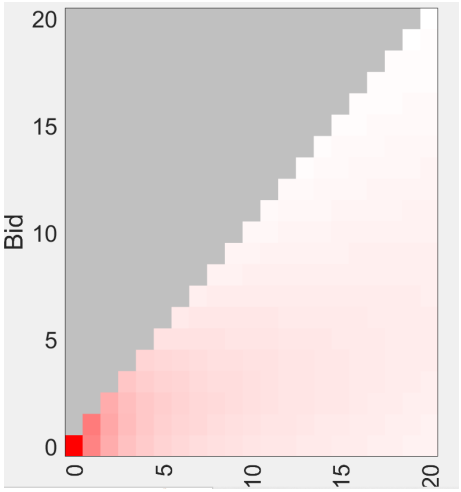
Computational proofs

While the theoretical analysis of contractivity based convergence resulted in requirement of a high temperature for entropy regularisation, it might be possible that our analysis was too conservative, and a more clever approach could have resulted in a smaller regularisation needed to ensure contraction. We show that for even for regularisations that result in nearly random policies, one

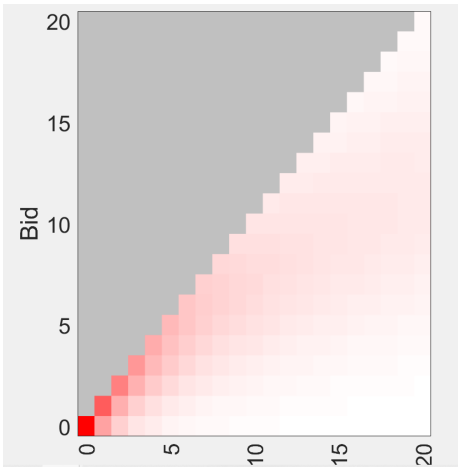
can find policies that violate contractivity in the PMA update. Towards this, we generate random policies computationally satisfying a particular value of regularisation and plot the violations of contractivity for different values of regularisation below. We also plot contractivity violations against discounting rate.

Setting: We use the same karma example from section 4.1, with urgencies $\mathcal{U} = \{1, 5\}$, with each urgency having equal probability, $\bar{k} = 10, k_{max} = 20, \gamma = 0.98$. We test the results for both PBP and PBS schemes. The strength of the regularisation term is varied. For each regularisation we compute the Nash equilibrium. We then generate $N_{exp} = 10000$ policies randomly as initialisation and only consider those that have entropy more than or equal to corresponding Nash policy (in the worst case of high regularisation these are 2000 in number). For each policy initialisation, we count one experiment. We start the MFG update using this initialisation and evolve the MFG using Policy Mirror Ascent for $T = 20$ iterations. To check for contractivity, we measure ratio of distance to the Nash policy of successive updates, if this is greater than one then contractivity is violated and we assign a failure to the experiment. The fractions of violations is calculated as ratio of experiments where this fails to total experiments. We plot the fractions of violations against different regularisations below. The maximum regularisation is $\beta = 1$, at which Nash policies become nearly random and meaningless from the karma perspective, which is shown in Fig 5.1 a and b. We show that even at this regularisation one can find violations of PMA update contractivity. Then for fixed value of regularisation, we repeat the same experiment with different discounting rates.

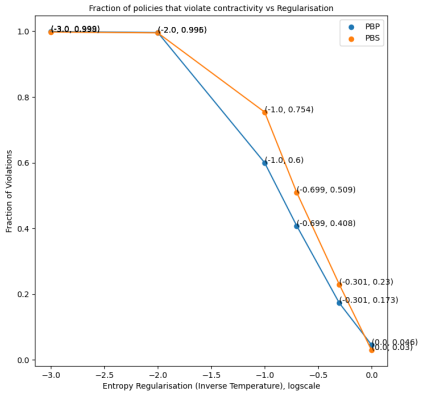
One can observe that even at high regularisations, contractivity may not always hold, although the probability of it being valid increases with regularisation for randomly generated policies satisfying the given regularisation constraint. At regularisations that are meaningful for the karma game, one can see that there are non-zero violations of contractivity, which implies that it might be difficult to get convergence guarantees using [18]. The behaviour improves with decreasing discounting rate, holding true for all experiments conducted in case of $\gamma = 0$. But $\gamma = 0$ is a trivial case in the karma game, where dominant stragies can be obtained analytically, so this result is not of much interest.



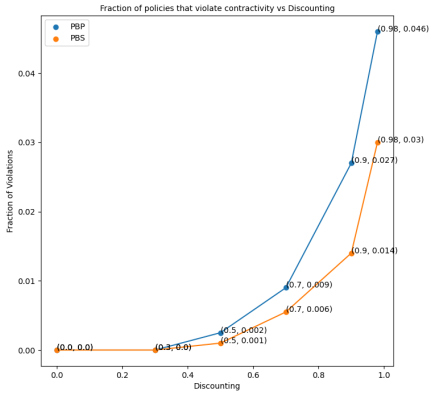
(a) Nash equilibrium for bounded rationality of 1, low urgency state, PBP



(b) Nash equilibrium for bounded rationality of 1, high urgency state, PBP



(c) Fraction of violations of contractivity for different regularisations for PBP and PBS, dicounting = 0.98



(d) Fraction of violations of contractivity for different discounting for PBP and PBS, bounded rationality = 1

Figure 5.1: Computational results for violations of contractivity

5.1.2 Monotonicity based proofs

In mean field games of practical interest, contractivity of the Banach Picard iterations is difficult to satisfy[2]. However, there are other formulations to analyse convergence based on the structure of the game. Under games satisfying monotonicity constraints for the game like Lasry-Lions monotonicity[7] or Generalised Monotonicity[6], convergence guarantees in terms of decreasing exploitability can be obtained. We investigate some of these methods below, using a mixture of theoretical and computational proofs.

Theory based on Lasry Lions monotonicity

Lasry-Lions monotonicity requires that for the mean field game of interest, we have

$$S = \sum_{x,a} (\rho(x,a) - \rho'(x,a))(r(x,a,\rho) - r(x,a,\rho')) \leq 0 \tag{5.5}$$

If the game satisfies this then a finite horizon version of the game can be shown to have convergence guarantees for exploitability [13]. Here ρ is the mean state action distribution. We investigate if this holds in the karma game. In the karma game, we have:

$$\begin{aligned}
&\rho(u,k,b) = \Pr(U=u, K=k, B=b)\\
&\nu(b) = \sum_{uk} \rho(u,k,b)\\
&r(u,k,b,\rho) = -u\theta[1 \mid b,\rho] = u[\theta[0 \mid b,\rho] - 1]
\end{aligned}$$

The probability of winning for the given bid and mean field can be written as

$$\theta[0 \mid b,\rho] = \sum_{u,k}\sum_{b'<b} \rho\left(u,k,b'\right) + 0.5\sum_{u,k}\rho(u,k,b)$$

Let $F_\nu(b) = \Pr(\nu \leq b)$

$$\begin{aligned}
&\therefore \theta[0|\nu,b] = F_\nu[b] - 0.5\nu[b]\\
&= F_\nu[b-1] + 0.5\nu[b]
\end{aligned}$$

The sum for monotonicity can be rewritten as:

$$S = \sum_{u,k,b} \left(\rho[u,k,b] - \rho'[u,k,b]\right) u(\theta[0 \mid b,\nu] - \theta[0 \mid b,\nu']) \tag{5.6}$$

We investigate the sign of this sum under different urgency distribution. For the first case, let us consider all agents have a single urgency. This simplifies the game into a zero sum dynamic game.

Case 1:
$\mathcal{U} = \{1\}$

$$S = \sum_b \left(v[b] - v'[b]\right)\left(F_v[b] - F_v,[b] - 0.5v[b] + 0.5v[b]\right)$$

$$S = -\frac{1}{2}\sum_b \left(\nu[b] - \nu'[b]\right)^2 + \sum_b \left(F_\nu[b] - F_{\nu'}[b]\right)\left(\nu[b] - \nu'[b]\right) \tag{5.7}$$

We can substitute $\nu[b] = F_\nu[b] - F_\nu[b-1]$ to rewrite the same sum as

$$S = \frac{1}{2}\sum_b \left(\nu[b] - \nu'[b]\right)^2 + \sum_b \left(\nu[b] - \nu'[b]\right)\left(F_\nu[b-1] - F_{\nu'}[b-1]\right) \tag{5.8}$$

We add 3.10 and 3.11 to get

$$\begin{aligned}
S + S &= \tfrac{1}{2}\textstyle\sum \left(\nu[b] - \nu'[b]\right)^2 - \tfrac{1}{2}\sum \left(\nu[b] - \nu'[b]\right)^2\\
&\quad + \textstyle\sum_b \left(F_\nu[b] + F_\nu[b-1] - F_{\nu'}[b] - F_{\nu'}[b-1]\right)\left(\nu[b] - \nu'[b]\right)\\
2S &= \textstyle\sum_b \left(F_\nu[b] + F_\nu[b-1] - F_{\nu'}[b] - F_{\nu'}[b-1]\right)\left(F_\nu[b] - F_\nu[b-1] - F_{\nu'}[b] + F_{\nu'}[b-1]\right)\\
S &= \tfrac{1}{2}\textstyle\sum_b \left(\left(F_\nu[b] - F_{\nu'}[b]\right)^2 - \left(F_\nu[b-1] - F_{\nu'}[b-1]\right)^2\right)\\
\therefore S &= 0
\end{aligned}$$

Hence for $\theta[0 \mid b,\nu] = \sum_{b'<b} \nu\left[b'\right] + 0.5\nu[b]$ and $u = \{1\}$ we do not have weak monotonicity but instead the term evaluates to zero

$$\sum_{u,k,b} \left(\rho[u,k,b] - \rho'[u,k,b]\right)\left(\gamma[u,k,b,\rho] - \gamma\left[u,k,b,\rho'\right]\right) = 0$$

However, if we instead modify $\theta[0 \mid b, \nu] = \sum_{b'<b} \nu[b']$ ie agent wins only if his bid is higher, then for $u = \{1\}$

$$\begin{aligned} 2S &= -\sum_b \left(v[b] - v'[b]\right)^2 + \sum_b \left(F_v[b] - F_v, [b]\right)^2 - \left(F_v[b-1] - F_{v'}, [b-1]\right) \\ &= -\sum_b \left(v[b] - v'[b]\right)^2 + 0 \\ \therefore \quad S &= -\frac{1}{2}\sum_b \left(v[b] - v'[b]\right)^2 \end{aligned}$$

ie we have strict weak monotonicity if mechanisms is changed st. agent wins only if his bid is higher. Note that this modification also changes the game from zero sum.

Now we move on to the case of general urgency spaces for all agents

Case 2
General urgency space $\mathcal{U}$

$$S = \sum_{u,k,b} \left(\rho[u,k,b] - \rho'[u,k,b]\right) u \left(\theta[0|k,b,\rho] - \theta\left[0|k,b,\rho'\right]\right)$$

We change back from mean bid distribution to mean state-action distribution

$$\nu[b \mid u] = \sum_k \rho[k, b \mid u]$$

Since u is an exogenous process, both the mean state-action distributions in the sum have the same marginal over urgencies

$$\sum_{k,b} \rho[u,k,b] = \sum_{k,b} \rho'[u,k,b] = P(u)$$

The sum term can be rewritten as

$$S = \sum_u P(u) \left(\sum_{k,b} \left(\rho[k,b \mid u] - \rho'[k,b \mid u]\right) \right) u \left(\theta[0 \mid b, \nu] - \theta\left[0 \mid b, \nu'\right]\right)$$

We use the notation $\nu[b|u]$ for $\sum_k \rho(k, b|u)$

Also, we can write $\nu[b] = \sum_{u'} P(u')\nu[b|u]$, as a sum over marginals

Note that $\theta[0|b,k,\nu] = Pr(\nu < b) + 0.5\nu(b)$, which is a linear function of the bid distribution probability vector.

We can rewrite $\theta[0|b,k,\nu] = \sum_{u'} P(u') \left(Pr(\nu[.|u'] < b) + 0.5\nu[b|u']\right) = \sum_{u'} P(u')\theta[0,b,k,\nu(\cdot|u')]$ by exploiting this structure of win probability.

Hence the sum can be rewritten as

$$S = \sum_u uP(u) \sum_{u'} P\left(u'\right) \sum_b \left(\nu[b \mid u] - \nu'[b \mid u]\right) \left(\theta\left[0 \mid b, \nu(.|u')\right] - \theta\left[0 \mid b, \nu'(.|u')\right]\right.$$

This can be written in matrix form as

$$S = (uP_u)^\top \left(\nu_{b,u} - \nu'_{b,u}\right)^\top \Theta \left(\nu_{b,u} - \nu'_{b,u}\right) P_u \tag{5.9}$$

Where

$$\begin{aligned} \nu_{b,u} &= \nu[b \mid u] \\ P_u &= P(u) \\ uP_u &= uP(u) \end{aligned}$$

$$\Theta = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 1 & 0.5 & 0 & 0 \\ 1 & 1 & \ddots & \end{bmatrix} \quad \text{if agent can win with bid}$$

$$\Theta = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & \ddots & \end{bmatrix} \quad \text{if agent loses if he bids equal to opponent.}$$

Note that $(\nu_{bu} - \nu'_{bu})$ has columns that sum to zero, because these are probability distributions that sum to 1. Also, one has
$\rho[u,k,b] = 0 \quad \forall b > k$ and $\quad \sum_u \sum_k \sum_b k\rho[u,k,b] = \bar{k} \quad \forall \rho$

To check if $S \leq 0$ under these conditions:

$$S = (uP_u)^\top \left(\nu_{bu} - \nu'_{bu}\right)^\top \Theta \left(\nu_{bu} - \nu'_{bu}\right) P_u$$

Here Θ is $n_b \times n_b$ matrix st $\Theta\nu = \theta[0 \mid b, \nu]$,

ν_{bu} is $n_b \times n_u$ matrix st $\nu_{bu} = v[b \mid u] = \sum_k \pi_k(b \mid k)$

$P(k \mid u)\ P_u$ is $n_u \times 1$ st $\{P_u\}_{j1} = P_r\left(u = u_j\right)$

uP_u is $n_{u\times 1}$ st $\{uP_u\}_{j1} = u_j \Pr\left(u = u_j\right)$

Lemma 5.1. *$\exists P_{k|u}, P'_{k|u}, \pi(b \mid k), \pi'(b \mid k), \mathcal{U}, P_u$ st.*

$$\begin{aligned} &\sum_{u,k} P(u)P_{k|u}k = \sum_{k,u} P(u)P'_{k|u}k = \bar{k} \\ &\pi_u(b \mid k) = 0 = \pi'_u(b \mid k) \quad \forall b > k, \forall u, k \\ &S \geq 0 \end{aligned}$$

We prove the above by finding a theoretical counterexample that violates monotonicity.

Proof: Let $P_{k|u}, P'_{k|u} st$

$$\sum_k P_{k|u}k = \bar{k} = \sum_k P'_{k|u}k \quad \forall u.$$

Clearly this also implies $\sum_{k,u} P_u P_{k|u}k = \bar{k} = \sum_{k,u} P_u P'_{k|u}k$

Also let π, π' st $\Pi_u(b \mid k) = 0 = \Pi_u(b \mid k) \quad \forall b > k, \forall k.$

We show that with above policies and karma distributions (which are hardly restrictive), we can find counter examples to monotonicity, by choosing appropriate urgency distributions. Towards this, we compose the terms in the matrix product of summation as

$$S = \underbrace{(uP_u)^\top}_{x^\top} \underbrace{(\nu - \nu')^\top \Theta (\nu - \nu') P_u}_{y}$$
$$S = x^\top y.$$

Suppose y is st. $y_j \geq 0$ for some $j \in \{1, \dots n_u\}$. Then one can set u_j to be very large s.t.

$$x^\top y = \sum u_j (P_u)_j y_j \geq 0$$

Hence if $S \leq 0$, then $y_j \leq 0 \quad \forall j \in \{1, \dots n_u\}$.

But y is composed of the following matrix products.

$$y = (\nu - \nu')^\top \gamma (\nu - \nu') P_u$$

Here ν is $n_b \times n_u$ st $\nu_{bu} = \sum_k P_{k|u} \pi_u(b \mid k)$.

y can be decomposed as another matrix product as follows

$$y = Ap \quad \text{where} \quad A \text{ is } n_{u\times} n_u, \quad P \text{ is } n_{u\times 1}.$$

Since we have selected $P_{k|u}, P'_{k|u}$ st $\sum k P_{klu} = \bar{k} = \sum k P'_{klu} \ \forall u$, we have full freedom on choice of P_u.

Since $y_j \leq 0 \quad \forall j \in \{1, \dots n_u\}$ for monotonicity to hold

$$\Rightarrow \quad (Ap)_j \leq 0 \quad \forall j \in \{1, \dots n_u\}$$

Let $A_{ij} \geq 0$ for some i, j

Then one can set $p(u = u_i) \approx 1$, and still satisfy average Karma constraint due to choice of $P_{k|u}, P'_{k|u}$

$$\Rightarrow y_j \geq 0$$

Hence if $y_j \leq 0$ then $A_{ij} \leq 0 \quad \forall i, j$

But $\quad A = (\nu - \nu')^\top \Theta (\nu - \nu')$, which is a skew symmetric matrix

$$A_{ij} = \left(v_{\cdot,i} - v'_{\cdot,i}\right)^\top \gamma \left(v_{\cdot,j} - v'_{\cdot,j}\right)$$

Suppose that for given $P_{k|u}, P'_{k|u}, \pi, \pi$ ' one has

$$A_{ij} \leq 0 \quad \forall i, j$$

Let us propose new $\tilde{P}_{k|u} \cdot \tilde{P}'_{k|u}, \tilde{\pi}, \tilde{\pi'}$ s.t. we keep the same karma distribution as before but flip the policies of the two mean field terms at one of the urgency states u_l

$$\begin{aligned}
&\tilde{p}_{k|u} = p_{k/u} \quad \forall k, u \\
&\tilde{p}'_{k|u} = p'_{k|u} \quad \forall k, u \\
&\tilde{\pi}_u(b \mid k) = \pi_u(b \mid k) \quad \forall b, k \text{ and } u \neq u_l \\
&\tilde{\pi}_{u_l}(b \mid k) = \pi'_{u_l}(b \mid k) \\
&\tilde{\pi}'_u(b \mid k) = \pi'_u(b \mid k) \quad \forall b, k \text{ and } u \neq u_l \\
&\tilde{\pi}'_{u_l}(b \mid k) = \pi_{u_l}(b \mid k)
\end{aligned}$$

$$\Rightarrow \left(\tilde{v}_{\cdot,j} - \tilde{v}'_{\cdot,j}\right) = \left(v_{\cdot,j} - v'_{\cdot,j}\right) \quad \forall j \neq l$$
$$and \left(\tilde{v}_{\cdot,l} - \tilde{v}'_{\cdot,l}\right) = \left(v'_{\cdot,l} - v_{\cdot,l}\right)$$

Hence if we have $A_{ij} \leq 0 \quad \forall i, j$ Then we can change policies for one of the urgencies u_l such that

$$\tilde{A}_{il} \geq 0 \text{ for some } l.$$

Hence one can construct $P_{k|u}, P'_{k|u}, \pi, \pi$, st $S \geq 0$ and average Karma constraint is satisfied. Notice that in case of single urgency, Eq 3.12 simplifies to

$$S = (P_u)^\top \left(\nu_{b,u} - \nu'_{b,u}\right)^\top \Theta \left(\nu_{b,u} - \nu'_{b,u}\right) P_u = x^T \Theta x$$

Here x is a probability vector which sums to 1. Due to the structure of Θ, the summation reduces to a value less than or equal to zero in case of single urgency. But in case of general urgency distributions, one does not get the above simplification and as proved above, can find counterexamples where monotonicity fails.

Hence we can show that Lasry Lions monotonicity does not hold for karma games with general urgencies.

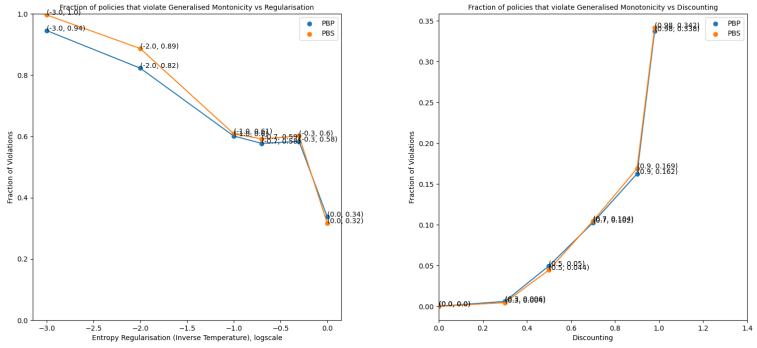
Computational proof for generalised monotonicity

Generalised monotonicity is another form of monotonicity that can be used to get guarantees on convergence for exploitability. Lasry-Lions monotonicity can be shown to be a subset of Generalised monotonicity. In a Mean Field Game, with Nash equilibrium π_*, generalised monotonicity can be written as:

$$J(\pi_*, \Gamma^\infty_{pop}(\pi)) \geq J(\pi, \Gamma^\infty_{pop}(\pi)) \forall \pi \in \Pi_H \tag{5.10}$$

i.e. for any policy in the hypothesis set, the Nash policy performs better than or equal to the same policy on the environment characterised by mean field of the same policy.

Similar to the computational results for testing contractivity, we also check for generalised monotonicity below, in the same setting of Chapter 4.1. We plot the results for different regularisations and discounting rates below



(a) Fraction of violations of generalised monotonicity for different regularisations

(b) Fraction of violations of generalised monotonicity for different discounting

Figure 5.2: Computational results for violations of generalised monotonicity

5.2 Summary of proof techniques

We summarise the observations from theoretical and computational proof approaches below in Table 5.1. While we were not successful in obtain convergence guarantees for the karma game in the mean field game case, we were able to demonstrate, using a mixture of theoretical and computational techniques, the conditions and reasons for failure of guarantees of popular analysis techniques for convergence in mean-field games. An important observation was that while some of the conditions required for convergence failed at meaningful regularisations, the percentage of violations did indeed decrease with increasing regularisation. Also, as we shall also see in Chapter 4, we could not find more than one Nash equilibrium in the karma game, and the algorithm for learning the MFG solution converges empirically under a variety of settings. This indicates that the mean field update is strongly attractive to the Nash equilibrium, but the update may not result in guaranteed monotonic improvement for metrics like exploitability or distance to Nash equilibrium. Future approaches could include annealing regularisation in MFGs[11] or convergence guarantees with parametrized policies[9]. We have a theoretical result on log-linear shape of policies in Appendix A, which can be used with parametrized policy learning in MFGs

Proof technique	Conditions	Theoretical proof	Computational proof
MFG contractivity	Lipschitz continuity of rewards, transitions	Lipschitz continuous, but need very high regularization for contractivity	Contractivity fails to hold for meaningful regularizations
Lasry Lions reward monotonicty	$\sum_{x,a}\{\rho(x,a)-\rho'(x,a)\}\{r(x,a,\rho)-r(x,a,\rho')\} \leq 0$	Doesn't hold for multi-urgency games (proof by contradiction)	-
Generalized monotonicity	$J_h(\pi^*, \rho(\pi)) \geq J_h(\pi, \rho(\pi))$	-	Fails to hold for large fraction randomly generated policies, worse at lower regularisations

Table 5.1: Summary of theoretical and computational proofs

Chapter 6

Numerical experiments of MARL in karma games

In this chapter we apply the MARL simulator setup developed in Chapter 3 to a variety of karma mechanisms. Firstly, we benchmark the MARL simulator against existing centralized techniques for Nash equilibrium in karma game defined in [3]. We analyse the conditions under which MARL converges and compare it against the centralized solution. We also conduct a variety of robustness studies to demonstrate the applicability of our method in a variety of karma settings. We then analyse the learned Multi-Agent Nash equilibrium in the karma game from the perspective of different social optima metrics and compare it to non-karma centralized resource allocation mechanisms. Then we study a different problem in which karma games can be applied: a karma game applied in traffic in a congested city model. We study convergence and compute different metrics to characterise the social optima and compare it to non-karma centralized resource allocation mechanisms. Finally we summarise our results and propose future research directions based on our results.

6.1 Benchmark experiment

In this section we conduct experiments with MARL applied to a karma model proposed in [3], which also provides a centralized mean field method to compute Nash equilibria in such karma games. This serves as a benchmark to test the performance and convergence of our MARL algorithm in a karma game with a known solution (in the limit of large populations), as well as test whether the results of [3] regarding fairness and efficiency hold when agents learn independently, privately and greedily.

Setting

We assume a population of different types of agents. Each agent has a private static type $\omega \in \Omega = \{\omega_1, \dots \omega_{|\Omega|}\}$. Agents from this population compete for access to a scarce resource at every timestep. Each agent has a private time varying urgency $u \in \mathcal{U}$ to access the resource. Additionally, each player has karma $k \in \{0, \dots k_{max}\}$. At every timestep, the agents' urgencies undergo transitions by a Markovian process $P(u^+|u)$, which is exogeneous to his karma. At every timestep, two agents are randomly matched from the population to compete for the resource and they bid for access using their karma. The bid for an agent is denoted by b and cannot exceed his karma k. If the agent loses the bid, he incurs a negative reward $r = -u$ equal to his urgency. All agents belonging to type $\omega \in \Omega$ discount their future rewards by γ_ω in the infinite horizon disocunted game. Furthermore, agents are assumed to follow a stationary policy that is time invariant. After the outcome of a bid, the bids by the agent are redistributed by a redistribution scheme specified by the user. In our experiments, we only consider the Pay-bid-to-Peer (PBP) and Pay-Bid-to-Society (PBS), as introduced in Chapter 2.3. In the PBP scheme, the winner pays his

bid to the loser while in the PBS scheme, the winners' bids are collected to the societal surplus and redistributed uniformly. In case of redistribution, karma is paid to the user until the maximum karma k_{max} is reached and any remaining karma is added to the societal surplus for redistribution.

Hyperparameters

We now discuss the setup for algorithm to obtain Nash equilibria for the above mentioned karma games using Multi-Agent Reinforcement Learning (MARL). Setup of the MARL scheme and its convergence behaviour is conditional on values of the hyperparameters for the learning algorithm. A detailed definition of all hyperparameters of MARL scheme was done in Chapter 2.2, here we discuss the hyperparameters relevant to experiments in following sections.

- Learning rate α: Learning rate for Q-function updates of Algorithm 1 in the MARL experiment. We assume that this is the same for all agents belonging to a population type.
- Discounting rate γ: Rate at which agents discount future rewards, assumed to be the same for all agents within a type.
- Number of agents in a type ω: $|\mathcal{N}_\omega|$.
- Exploration rate ϵ: Any agent is assumed to execute his current policy with probability $(1-\epsilon)$ and a uniformly random policy with probability ϵ. Lower exploration rates along with high bounded rationality β would point to highly rational agents, however in practice exploration rate cannot be too small. This is because agents learn by exploring the state-action space and need to collect enough data to know the value of their policies. Additionally, due to the non-stationary nature of the environment Multi Agent dynamic games, any player cannot afford to have small exploration rate in case the environment changes due to others changing their strategies.
- Epoch length $T_{\text{iters per epoch}}$: Agents fix their policies for $T_{\text{iters per epoch}}$, gather data and update their policies based on this. Smaller epoch lengths mean frequent policy updates but potentially rapidly changing environment.
- Learning algorithm: We consider two learning algorithms: $\epsilon - SARSA$[16], also followed in [18], where agents find the next policy based on action value estiamtes of current policy and Q Learning[16], where agents find the next policy based on greedy action value estimates.

6.1.1 Robustness of MARL in benchmark karma games

We now run experiments for MARL in karma games. First we do a hyperparameter sensitivity analysis, where we keep the karma game fixed and choose a set of default hyperparameter values. We then perform a hyperparameter sensitivity analysis by changing hyperparameters one by one, keeping others fixed and report our findings. In the experiments where either the karma game, or hyperparameters, or both are changed, we will specify these accordingly. The karma game we consider has all players belonging to a single population type, and have $k_{max} = 20$. We let the average amount of karma in the system $\bar{k} = 10$, which remains constant throughout by redistribution and non-tradeable nature of karma. The urgencies are chosen to as

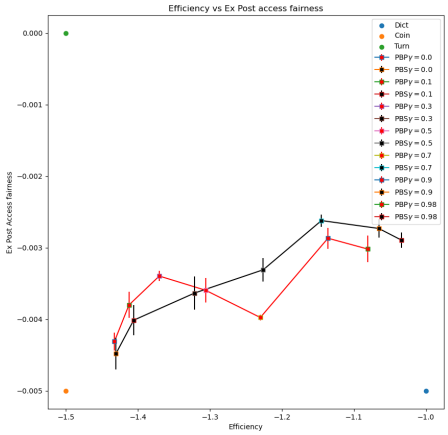
$$\mathcal{U} = \{1, 5\} \quad ; P(u^+|u) = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$$

This results in both urgencies being equally probable at the stationary distribution of urgency Markov Chain, resulting in $\bar{u} = 3$. We consider all agents having a discounting of $\gamma = 0.98$, and the karma mechanism to be Pay-Bid-to-Peer (PBP).
The default values for MARL are chosen as $\alpha = 0.01$, $\epsilon = 0.05$, $N = 100$, bounded rationality $\beta = 1000$, $T_{\text{iters per epoch}} = 1000$. The total timesteps are equal to $T_{timesteps} = 3.2 \times 10^7$, which was sufficient to ensure convergence in all cases. The default learning algorithm is chosen to be

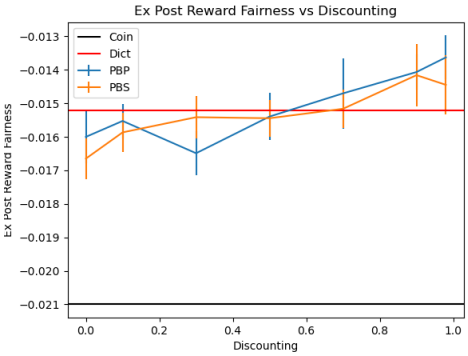
ϵ- SARSA. We vary learning rates, exploration rates, population size, epoch lengths, learning algorithms for this benchmark setting and record our observations. We also perform simulations to test robustness of the learning algorithm, under different karma mechanisms, urgency models and multi-type populations. We summarise the results for the above experiments in Table 6.2, a more detailed analysis can be found in Appendix A.1.

6.1.2 Social outcome evaluation

We now turn our attention to evaluating the performance of MARL policies in the karma game from the perspective of social outcomes measured using *efficiency, ex-post-access-fairness, ex-post-reward-fairness*, as introduced in Chapter 2.3. We keep the urgency transition model the same as the default case, set number of agents to $N = 200$. We evaluate the performance of both PBP and PBS for different discounting rates and compare them to other centralized allocation schemes that are not based on karma. For learning, instead of exploration like in previous experiments, we use bounded rationality for players, with $\beta = 5$ (as opposed to $\beta = 1000$, from the paper). This is to encourage sufficient exploration for convergence, although this hyperparameter can be optimized in the future. Other hyperparameters are kept the same as in previous experiments. For evaluation of the social function, we measure *reward efficiency, ex-post-access-fairness, ex-post-reward-fairness*, for time duration of $T = 10000$. We plot these metrics for different discounting rates below, for both PBP and PBS and compare them with non-karma resource allocation. The karma mech-



(a) ex-post-access-fairness vs efficiency for different mechanisms



(b) ex-post-reward-fairness for different discounting rates in karma games, compared to mean values for centralized allocation

Figure 6.1: Evaluation of social outcomes in MARL applied to karma games

anisms are compared to allocation mechanisms like TURN, COIN and DICT, as introduced in Chapter 2.3, similar to [3].

As the discounting rate increases, policies learned using MARL applied to the relevant karma game approach the efficiency of DICT while having greater *ex-post-access-fairness*. MARL games with PBS mechanisms perform slightly better than PBP. In the plots for *ex-post-reward-fairness*, one can observe that as discounting rate increases, *ex-post-reward-fairness* increases and both PBP and PBS beat DICT at high discounting rates. This is due to the ability of karma to both encourage turn taking but also as a means to express private urgencies for the players.

An interesting case to analyse would be to compare the policies learned using MARL to policies learned using MFG, in a finite population setting. We do so below for the case of PBS with

$\gamma = 0.98$. Since MFG policies are learned in the assumption that population of agents is infinite, one may not expect them to outperform policies learned using decentralized learning in the finite population game. But the decentralized finite population game also has a disadvantage that players learn independently and greedily, which may prove disadvantageous from a social outcome perspective. In our results in Table 6.1, we see that if players play the policy given to them by the centralized MFG learner, they perform better than the case when they learn their own greedy policies. One may conclude that independent and greedy learning hurts everyone involved, compared to following the centralized learning policy. One potential reason that could explain worse performance of MARL is also lack of hyperparameter optimization, especially the value of bounded rationality used. Further work is needed to strengthen the results above.

Mechanism	Efficiency	Ex Post reward Fairness	Ex Post Access Fairness
MARL PBS ($\gamma = 0.98$)	-1.033	-0.0144	-0.0042
MFG PBS ($\gamma = 0.98$)	-1.012	-0.0128	-0.0025
DICT	-1.00	-0.0152	-0.0049
COIN	-1.49	-0.021	-0.0051
TURN	-1.50	-0.018	-0.0001

Table 6.1: Performance of various mechanisms from the perspective of social outcomes

Conclusions and Summary of results

We summarise the results of experiments with Multi-Agent Reinforcement Learning applied to karma games of urgency models in Table 6.2. We observe that MARL converges to MFG solution of [3]under a wide variety of settings. Important results of interest are that even with independent greedy Q learning and no synchronisation of policy updates (epoch length=1), MARL converges to the MFG solution of [3] for small enough learning rates. Additionally, the results are also robust to the case of multi-type populations. This part of the analysis implies that if an agent reveals his true urgency transitions (utilities) to a central policy planner in a large population karma game, then he can can trust the solution strategy given to him by the central planner since he cannot do much better even if he is allowed to greedily and independently learn his own behaviour.

We also analysed efficiency and fairness of the social outcomes for MARL in karma games and found that for high discounting rates, the Nash equilibrium learnt using MARL has high efficiency along with equitable outcomes. We also noted that MARL seems to perform slightly worse compared to centralized solution, due to greedy and independent learning hurting everyone. This further strengthens the case for policies learned using [3], not only do MARL algorithms under various settings converge to the centralized strategy solution in large enough populations, but following the policies learned using this method is also beneficial from a social outcome perspective.

Type of analysis	Category	Notes	Results and Conclusions
Hyperparameters	Learning rate α	Experiments with different constant α	MARL converges for small enough learning rates, and converges to MFG solution in large populations
	Exploration rate ϵ	Experiments with different ϵ	MFG with fully rational players is the limiting case of MARL with small ϵ and large populations
	Population size	Experiments under number of agents	MFG is the limiting case of MARL with large populations
	Epoch length	Players keep their policies constant during this window and then update	MARL converges to MFG for all epoch lengths given sufficient time, even for epoch length of 1, for small learning rates
	Policy Learning Algorithm	Greedy Q Learning vs ϵ SARSA	Both algorithms converge to approximately MFG. Independent greedy learning with small enough learning rates also converges to MFG.
Robustness	Different urgency	Experiments with rare high urgency state	MARL converges to MFG
	Different mechanism	Experiment with PBS instead of PBP	MARL converges to MFG
	Multitype populations	Multitype populations with different γ and different $\bar{u}$	MARL converges to MFG
Social outcomes	Measure efficiency and fairness	PBP and PBS with different γ's	MARL with high γ approaches has fair and efficient outcomes. Independent greedy learning hurts compared to centralized MFG solution

Table 6.2: Summary of results for MARL applied to karma games with urgency models like [3]

6.2 Karma mechanisms applied to traffic in congested city

We now formulate a karma game applied to a model of traffic congestion in a grid city, with agents travelling along different paths in the city and bidding for lane access by using their karma. Such a system is too complex to be analysed as a mean field game with analytical model transitions like was the case before, and is a case where we can expect our MARL simulator for karma games to be highly useful in analysis. This work builds on previous work in karma mechanisms applied to traffic in congested lane access[4].

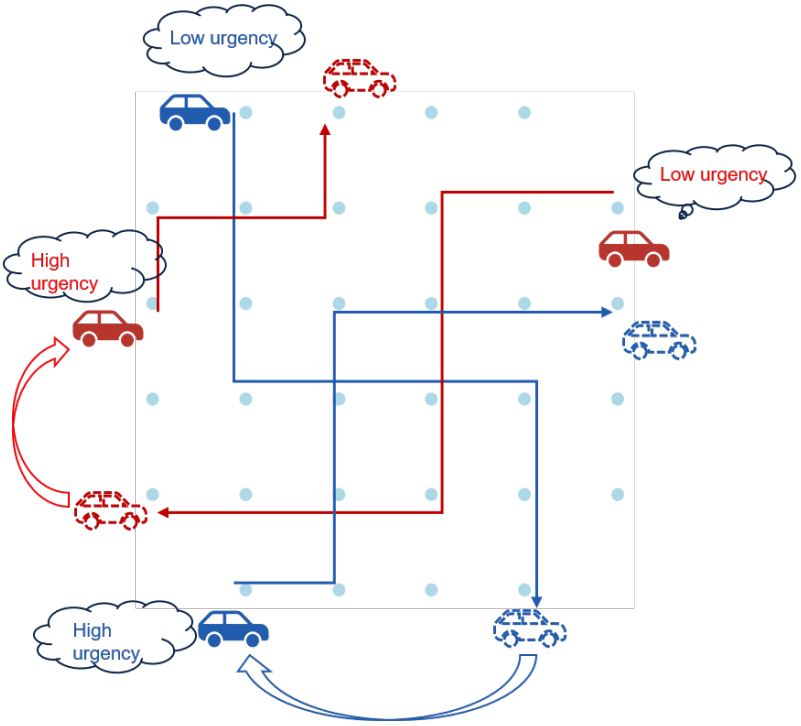


Figure 6.2: Schematic of a karma game in a congested city

Setting

We assume a large population of agents travelling along a grid city of finite size, as shown in Fig 6.2. We assume that lanes connecting two nodes in a city are directional in nature, meaning that traffic on a directed edge between two nodes has no effect on traffic in the other edge for the same two nodes. Additionally, we restrict each directed lane in the city to have a finite capacity, which can be lane dependent. At any timestep, agents are present at one of the nodes in the city and compete for access for a lane towards the next node. The number of agents travelling through any lane is upper bounded by the capacity of the lane, and in case there are more agents wishing to access a lane than its capacity, only total agents equal to capacity are allowed to travel while others remain at the origin node for the lane for that timestep.

For sake of simplicity, we assume that each agent has an original and destination (O, D) pair located only on the boundary points of the city, and not in the interior, although such an extension is trivial in our framework below. Furthermore, for the sake of this discussion, we assume that agents are not allowed to plan their paths to travel along the city, but are given a set of predetermined paths which they travel on. We denote the set of paths for agent i as $\mathcal{P}_i$. Any path $p_i \in \mathcal{P}_i$ is restricted to be the shortest path for its induced (O, D) pair $\forall i \in \{1, \dots N\}$. Furthermore, we also add a dormant path to the set $\mathcal{P}_i$. At any timestep, an agent is in one of the paths from $\mathcal{P}_i$. If

the agent is not in the dormant path, then he continues travelling through the city. If he is in the dormant path, then at next timestep he is assigned a new path from $\mathcal{P}_i$ (including the dormant path) according to some probability rule. If the agent has reached the assigned destination, he is assigned a new path from $\mathcal{P}_i$ with some probability transition rule. Note that the dormant path can also be defined as a directed edge between two nodes outside the city, with infinite capacity for that directed edge.

Additionally, we also have a set of urgencies $\mathcal{U}$ for all agents. Unlike the karma game of previous section, where urgency transitions were an independent Markov process every timestep, here urgency remains constant as long as the driver has not completed his trip. Once the agent i completes his trip, he is assigned a new path and urgency from the set $\mathcal{P}_i \times \mathcal{U}$. We assume that if the agent is in the dormant path, his urgency is restricted to be $u = 0$. Finally, each agent is endowed with karma, with the maximum karma being k_{max}. He bids for lane access using his karma, and the karma is redistributed among all agents using some redistribution rule after every timestep. The agent is not allowed to place a bid higher than his karma. Also, in case the agent is in the dormant path, he is not allowed to bid karma, but may be a recipient of redistribution. This can be trivially modified later to forbid dormant agents from redistribution. We summarise the parameters of the model below:

- $\mathcal{N} = \{1, \dots N\}$, the set of agents playing the game.
- $\mathcal{U}$: Set of urgencies for all players. Note that $u = 0 \subset \mathcal{U}$, since an agent in dormant state is assumed to have zero urgency.
- $\mathcal{P}_i$: Set of paths for agent i, which includes the dormant path. $\mathcal{P} = \bigcup_i \mathcal{P}_i$ is the set of all possible shortest paths between two nodes on the edges of the city.
- $P_i(p^+, u^+|p, u)$: Transition matrix for next path and urgency, once the agent has reached his destination. This includes the dormant path and zero urgency state tuple.
- γ_i: Discounting rate for agent i.
- $x(p)$: Location of the agent when he is travelling in a path. Each path consists of a sequence of origin nodes with $|p|$ nodes for path p. $x(p)$ denotes which of these nodes on path p is the agent currently in.
- $k \in \{0, \dots, k_{max}\}$: Allowed Karma for each agent.
- b Bid for the agent.
- $\mathcal{X}_i$ State space of the agent i. $x \in \mathcal{X}_i = (u, k, p, z(p))$, where $u \in \mathcal{U}$, $k \in \{0, \dots, k_{max}\}$, $(z(p), p) :\in \{0, \dots |p| - 1\} \times \mathcal{P}_i$.
- $\mathcal{A}$: Action space for all agents, which is the bid space.
- $\mathcal{L}$: Set of all directed edges (lanes) in the city.
- $W : \mathcal{L} \to \mathbb{N}$, capacity per lane.

Model parameters

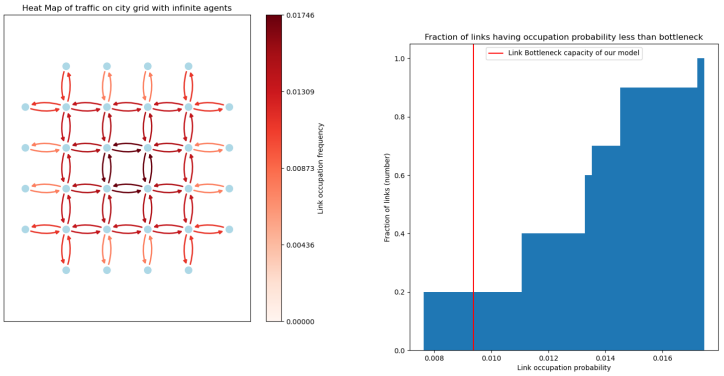
We now state the model parameters used for the simulation. We select a city with a grid layout as shown in Fig 6.3 a) and Fig 6.2. Origin and Destination pairs (O, D) for each agent travelling in the city are restricted to be on one of the outer edges of the city. To generate the set of all possible paths, we only consider the shortest path between any (O, D) pairs. Due to the grid structure of the city, there are multiple shortest paths between any (O, D) pair, which grow combinatorially with size of the city. In the city of dimension 4×4 as selected, there are $N_p = 1024$ possible shortest paths between any (O, D) pairs on the grid. The set $\mathcal{P}_{-d}$ is formed by collected the set of

all these paths, with $|\mathcal{P}_{-d}| = 1024$. The set of of paths including dormant path $\mathcal{P} = \mathcal{P}_{-d} \bigcup p_d$ is naturally of size $|\mathcal{P}| = 1025$. The total number of directed edges (lanes) in this city is $|\mathcal{L}_{-d}| = 80$, and the set of all possible lanes including dormant lane has size $|\mathcal{L}| = 81$. Now we need to set the bottleneck capacity for the lane. For doing so, we first plot the probability of lane occupancy if there are infinite agents travelling in the grid with unlimited capacity per lane. The heat map is shown in Fig 6.3 a), which also indicates that our choice of city model is sufficient to capture dynamics like city centre congestion. Additionally, the lanes at entry and exit to the main grid have the lowest congestion. We set the bottleneck capacity per lane to be equal for all lanes and such that entry and exit to the main grid should be unrestricted assuming the dynamics of Fig6.3 a). This is done by choosing a value of capacity $= \frac{0.75 \times N}{|\mathcal{L}_{-d}|}$. This ensures that entry and exit are unrestricted assuming steady state behaviour. This is shown in Fig 6.3 b).

In our simulations, we select number of agents $N = 1000$. Each agent is allocated $n_p^{(i)} = 2$ paths from this set, selected uniformly randomly with replacement. Additionally, each player path set $\mathcal{P}_i$ is composed from these paths along with the dormant path. The set of urgencies is selected as $\mathcal{U} = \{0, 1, 5\}$. Each player is assumed to discount his future rewards with $\gamma = 0.98$. The maximum allowed karma is $k_{max} = 14$ and mean karma is $\bar{k} = 9$. In our choice of the city, the maximum path length $max_p|p| = z_{max} = 8, p \in \mathcal{P}$. This results in agent state space having a total dimension of $|\mathcal{U}| \times k_{max} \times |\mathcal{P}_i| \times z_{max} = 1080$ and the action space of dimension $|\mathcal{A}| = 15$. Of course, not all elements of the state space will be accessed, since not all paths have maximum length, and $u = 0$ when dormant and $u \in \mathcal{U} \backslash \{0\}$.

For the urgency path transitions for all agents $P_i(p^+, u^+|p, u)$, we use the following rule. If the player completes his path, he is assigned a dormant path with probability $p_{dormant} = 0.1$. If he is assigned the dormant path then he is restricted to have urgency zero. He is assigned an active path with probability $1 - p_{dormant} = 0.9$. If he is assigned an active path he is restricted to have non-zero urgency, which are selected from $\{1, 5\}$ with equal probability conditioned on this event. Urgencies remain constant till he completes his path. If he is on the dormant path, he will certainly become available for new path assignment. Finally we use PBS redistribution where bids are redistributed uniformly to all agents, regardless of their assigned paths.

For learning the policies, we use the following hyperparameters. We set learning rate $\alpha = 0.01$ for all agents, bounded rationality $\beta = 10$ for all agents, $\gamma = 0.98$ for all agents, $T_{iters\ per\ epoch} = 100$ and number of epochs $N_{epochs} = 10^5$, giving the agents 10^7 timesteps to gather data, which is roughly $600\times$ the state action space per agent. We track the progress of reward efficiency, ex-post-access-fairness, ex-post-reward-fairness throughout the learning process.

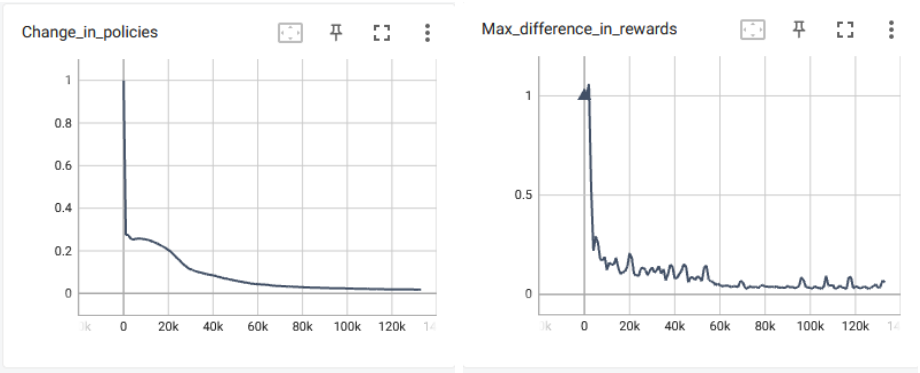


(a) Heat map of the city traffic with infinite agents

(b) Selection of bottleneck capacity

Figure 6.3: Setting of city grid with traffic

Observations



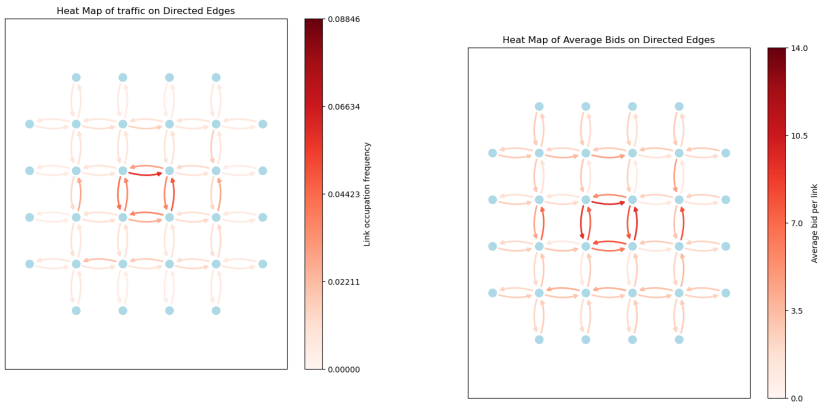
(a) Evolution of maximum change in successive policies of agents while training

(b) Evolution of maximum change in successive payoffs of agents while training

Figure 6.4: Convergence of MARL in karma game of congested city

We first verify convergence of the learning process. Due to the size of state action space, computing exploitability is expensive. In this setting, we track convergence using maximum change in agent policies and maximum change in agent payoffs (maxium over all agents in the game), as per our discussion in Chapter 4. The results for convergence are shown in Fig6.4

We analyse the congestion behaviour from the policies learnt above. Firstly we plot the resultant traffic congestion for the system of agents in Fig6.5. One can observe that city centre is congested, similar to Fig6.3 a). When we plot the average bid per lane in the city in Fig6.5 b), we can qualitatively see that congested areas tend to overlap with higher bids. This is also confirmed by a plot of average bid per lane against congestion for that lane in Fig6.6 which show an increasing relationship. This behaviour is expected since agents are penalised for every timestep in the city, so the maximize reward they would bid higher to avoid congestion.



(a) Heat map of observed congestion in the city

(b) Heat map of average bids in the city

Figure 6.5: Observed behaviour of agents in the city

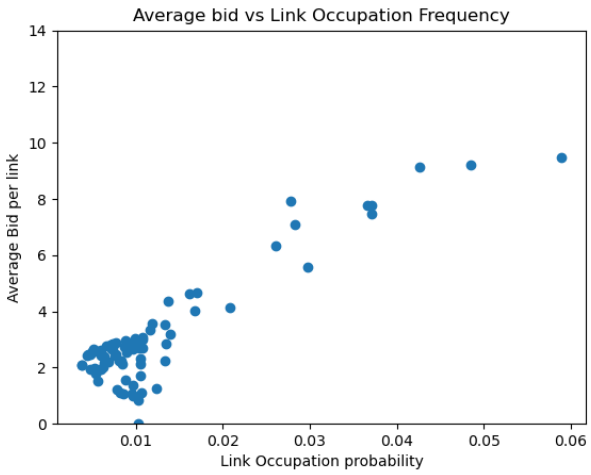
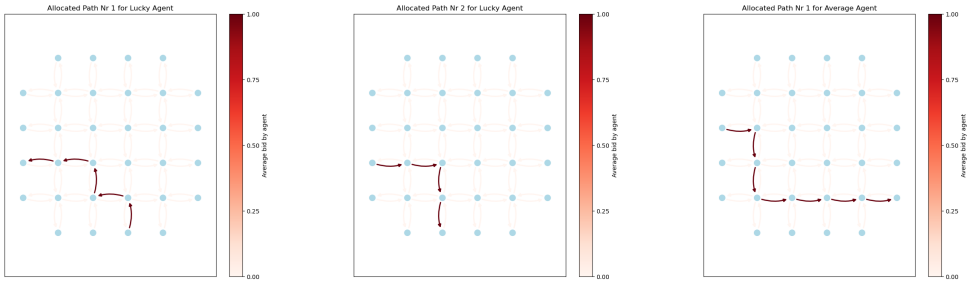


Figure 6.6: Bidding vs congestion

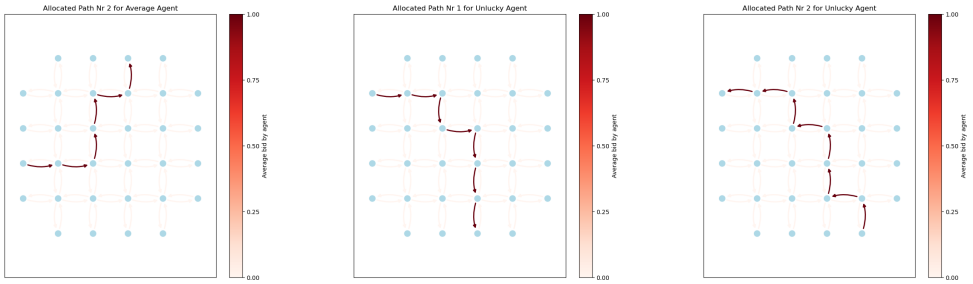
We now analyse the behaviour of different agents in the population. Note that we have randomly allocated two paths to agent, and agents cannot deviate from the path assigned to them. Hence, as a result of the randomised path allocation, one should expect to have three kinds of agents, depending on the set of paths assigned to them: Lucky Agent, Unlucky Agent, Average Agent. The lucky agent has both his paths that avoid the congested city centre. The unlucky agent has both his paths that pass through the city centre. The average agent has one path that passes through the city centre and one path that bypasses it. We expect these dynamics to reflect in their bidding strategies which we summarise in Table A.4 and in Fig 6.8. The paths of the three agents are plotted in Fig 6.7.

Firstly, we observe that all the three players bid higher when in a high urgency state. This is to minimize travel time in this state which is more costly in high urgency state compared to low urgency state. The unlucky player, who has both his paths passing through the city centre, has the highest bids amongst all the three. When comparing the lucky player to the average player, we get interesting results. The lucky player has similar bids for both his paths, since both of this paths are uncongested. When the average player travels through a congested path with high urgency, he bids higher than the lucky player. This is expected since he is trying to minimize incurred cost. However, when the average player travels though an uncongested path, he bids lower than the lucky player, in both high and low urgency states. Recall that in the karma redistribution scheme, we redistribute the bids to all agents uniformly, irrespective of their assigned paths. As a result, the lucky player ends up getting subsidised by unlucky and average players, because they have to bid higher when passing through congested lanes but do not get a proportional compensation for travelling through congested lanes. The lucky player is left with more karma compared to his needs due to always travelling through uncongested lanes and can afford to spend it to further minimize his costs. The average player, meanwhile, must save karma in the uncongested path to spend it in the future in a congested path so ends up bidding lower than the lucky player for both low and high urgency states. This results in unlucky player subsidizing lucky players, instead of the other way around, which is also reflected in wide disparity of average rewards computed for the three agents in Table A.4.

The unfair nature of karma redistribution in this example is also reflected when one measures social functions for efficiency and fairness of allocation. We compute social functions like *efficiency, ex-post-access-fairness, ex-post-reward-fairness*, as introduced in Chapter 2.3 and also studied in previous section, and compare karma mechanisms to centralized allocation schemes. We summarise our results in Table 6.3. One can observe that despite having high discounting $\gamma = 0.98$, karma



(a) Path number 1 for Lucky Player (b) Path number 2 for Lucky Player (c) Path number 1 for Average Player



(d) Path number 2 for average player (e) Path number 1 for unlucky player (f) Path number 2 for unlucky player

Figure 6.7: Path assignments for Lucky, Unlucky and Average players

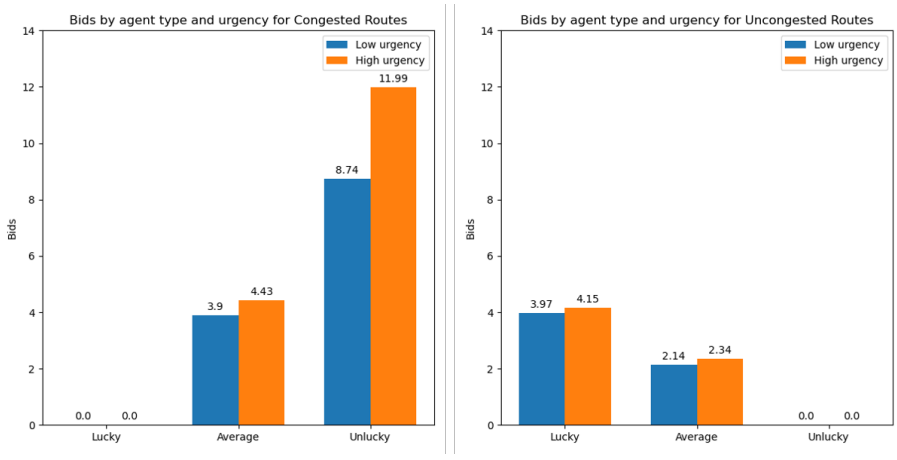


Figure 6.8: Bidding strategies of different agent types in the city

mechanisms do not replicate the result from the previous section. While the results of karma game are better than COIN in terms of efficiency, they are quite far off from DICT, unlike the setting from previous section where karma mechanisms had an efficiency almost equal to DICT. Additionally while the karma scheme results in better access fairness than DICT it is considerably worse than COIN.

Note that the above negative results for social functions are not due to incorrect learning proce-

Mechanism	Efficiency	Ex-Post-Access-Fairness	Ex-Post-Reward-Fairness
DICT	-0.42	-0.21	-0.21
COIN	-1.23	-0.10	-0.33
Karma	-0.82	-0.15	-0.43

Table 6.3: Social outcome evaluation in congested city model

dure. The behaviour learned by agents is sensible for the game they are given, but rather it is the setup of the game and traffic model that causes poor performance. One of the primary reasons for this is our assumption that players are not allowed to plan their paths from experience. This is not a rational assumption for the agents in this case. In the current setup, agents are not only forced to proceed along congested paths without an alternative, but also end up subsidising other players that do not have to go through the congested lanes. In such a scenario, a rational strategy for the players could be to switch to alternative paths. While we skipped the problem of path planning along with bid planning in our analyses due to lack of time, a realistic simulation would also involve path planning. We leave this for future work.

Another method to improve performance, without introducing path planning would be to do redistribution by lane instead of a global redistribution of karma. An approach to improve the performance of our experiments could be to have congestion dependent TAX mechanisms that subsidise players having to travel through the centre. While both of these solutions might work in the case of agents following pre-allocated paths, they may not be practically feasible due to not being robust to collusion among players: agents might end up travelling in congested lanes just to collect karma, or pay other agents to collect karma for them. However a study of karma games in traffic could be a hybrid the above suggestions: incorporate path planning but also allow subsidies for congested lanes. This also opens up the possibility of parametrized subsidies, where a policy planner learns to set optimal parameters to encourage a socially beneficial outcome. One could envision a bilevel mechanism design problem, where the planner plays a game against a population of agents to encourage social optimum. We leave this for future work.

Chapter 7

Conclusions and Future Work

Conclusions

We studied various aspects of the dynamic karma game in our work. Our work focussed on aspects related to real-life applicability of simplified karma games, with regards to convergence and distributed policy learning. We first proposed a decentralized multi-agent reinforcement learning algorithm that can be used to compute Nash equilibria in general karma games. We also examined various measures to track convergence to Nash equilibrium during the learning process of a general multi agent game.

We surveyed state of the art theoretical analysis techniques for convergence guarantees in large population dynamic games. We applied some of these in the karma game case. For some methods we were able to theoretically demonstrate that they fail to guarantee convergence. For some other methods, we were able to show their non-applicability via computational proofs. While our analysis might not have yielded positive results in terms of convergence guarantees, it shed light on the reasons for failure of above analysis techniques regarding convergence guarantees in karma games.

We tested the performance of our MARL setup in a benchmark experiment of a well studied karma game. Firstly, we empirically examined the conditions for MARL to converge in the benchmark karma game setting. Then, we compared the learned policies using MARL to the existing centralized solution in these karma games with large populations. We showed that in this case, distributed greedy reinforcement learning, without any revelation of utilities by players, finds a Nash equilibrium solution similar to the existing centralized solver which assumes truthful revelation of private utilities. This implies that if players truthfully provide their private urgencies and transitions to a centralized planner, they can trust the solution of the centralized planner to be as good as independent greedy learning of strategies.

Finally, we applied our techniques to the study of karma mechanisms applied to model of traffic in congested city. We analysed learned policies upon convergence to Nash equilibrium for different types of agents in the city and found their behaviour to be consistent for their assigned type and the given karma game. When analysing the social outcome of the karma game in this application, we found that it was not fair and efficient in allocation outcomes. We pointed out that this was not due to incorrect learning, but the setup of the karma game, where we had restricted agents from path planning for computational reasons. This resulted in unfair karma redistribution schemes. We pointed out future modifications w.r.t. path planning and karma redistribution schemes to mitigate this.

Future Work

The MARL simulator developed in this thesis can be applied to a wide variety of karma games, but is designed assuming agents learn their policies with reinforcement learning in a tabular manner.

Future work could be to extend this to allow parametrized policies for agents, which could be useful for karma games with large state-action spaces. Additionally, while we provided empirical evidence for use of different metrics to track convergence to Nash equilibrium during MARL, a rigorous theoretical analysis of the assumptions and conditions regarding their use will be insightful.

The analysis techniques we studied to obtain convergence guarantees for karma mean field game used some form of monotonic improvement during policy learning towards the Nash equilibrium, which is sufficient but not necessary for convergence. From empirical simulations, we noted that empirically the mean field game is strongly attractive to the Nash equilibrium. Hence, future work could focus on techniques like regularization annealing to analyse convergence in such games. One could also exploit smooth structure of karma game Nash policies to analyse convergence using parametrized policies, towards which we provided an initial result in Appendix B.

Finally, as noted in the observations regarding karma games applied to traffic simulation in a congested city, future work could be to allow path planning for agents. One could also look at parametrized karma redistribution schemes, which also opens up possibilities of a bilevel optimization framework for karma mechanism design. One could also study robustness to coalitions in karma games using this application as a motivating example.

Appendix A

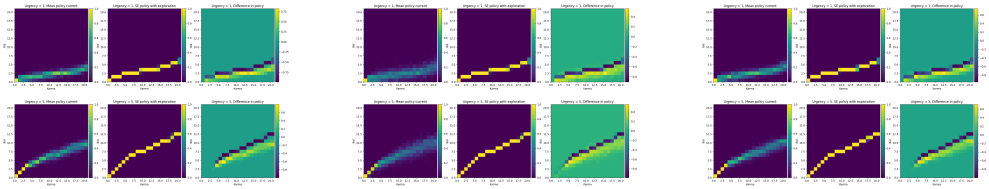
Numerical experiments

A.1 Benchmark experiment

A.1.1 Effect of learning rate

We run simulations for different learning rates that are kept constant throughout the learning.

Qualitative results



(a) Learned policies vs MFG policies $\alpha = 0.01$

(b) Learned policies vs MFG policies $\alpha = 0.05$

(c) Learned policies vs MFG policies $\alpha = 0.1$

Figure A.1: Learned policies for different learning rates

Metrics

Metric,Learning rate	$\alpha = 0.01$	$\alpha = 0.05$	$\alpha = 0.1$
$J(\pi_{curr}) - J(\pi_{mfg})$	0.1	0.066	0.006
$J(\pi_*) - J(\pi_{curr})$	0.006	0.024	0.041
$\delta(\pi)(\times 10^{-3})$	11.31	20.39	26.19

Table A.1: Numerical metrics comparison for different learning rates

Observations

Larger learning rates result in faster policy updates for each agent, but make the environment changes after every epoch more drastic, resulting in poor convergence in multi-agent setting. The effect of this can be seen in the measurements for exploitability at steady state convergence, increasing the learning rate increases the exploitability of the greedy policy. Also, the exploitability w.r.t. MFG policy, which corresponds to infinite agent case, is smaller at larger learning rates, another demonstration of poor performance at larger learning rates.

A.1.2 Effect of exploration rate

We run simulations for different exploration rates. Note that if one restricts policies to have a fixed nonzero exploration rate, the resulting Nash equilibrium will be different from the Nash equilibrium with zero exploration rate from [3], because the bid distribution of the population will be significantly different. Ideally, in the comparison of MARL against MFG we should run MARL with zero exploration rate so that it operates at the true Nash equilibrium (fully rational agents) in the infinite agent setting. But this would imply agents do not get a chance to learn the optimal policies. Therefore, we need to measure the exploitability of MARL against two infinite population settings: karma MFG where agents follow their best response policy with no exploration (fully rational), and another karma MFG where agents follow a combination of best response policy and random exploration, with the same exploration rate as MARL. MARL in the infinite agent limit with exploration tending to zero should match MFG of zero exploration. So we simulate MARL for different explorations, and extrapolate its behaviour to the limiting zero exploration case.

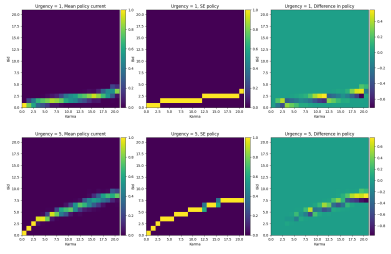
Metrics

Metric,ϵ	$\epsilon = 0.005$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.25$
$J(\pi_{curr}) - J(\pi_{mfg})$	-0.01	0.001	0.008	0.016
$J(\pi_{curr}) - J(\pi_{mfg,\epsilon})$	0.09	0.10	0.08	0.18
$J(\pi_*) - J(\pi_{mfg})$	0.003	0.006	0.002	0.01
$d(\pi_{curr}, \pi_{mfg})(10^{-3})$	14.99	28.09	37.74	54.9
$d(\pi_{curr}, \pi_{mfg,\epsilon}), (10^{-3})$	46.98	55.87	47.72	34.34

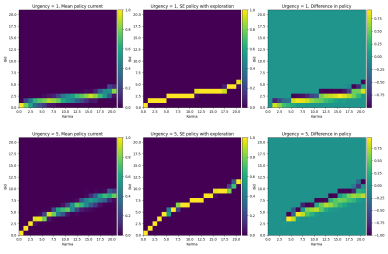
Table A.2: Numerical metrics comparison for different epoch lengths

Here $J(\pi_{curr}) - J(\pi_{mfg})$ is the normalised exploitability of the current policy (which includes exploration) w.r.t MFG policy for no exploration. $J(\pi_{curr}) - J(\pi_{mfg,\epsilon})$ is the normalised exploitability of the current policy (which includes exploration) w.r.t MFG policy with the same exploration rate.

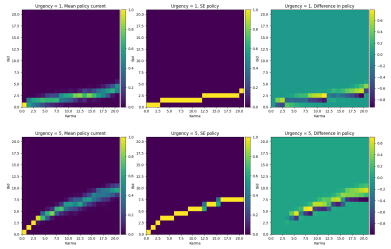
Policies



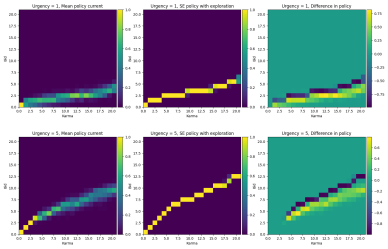
(a) Policy comparison w.r.t MFG (no exploration) for $\epsilon = 0.005$



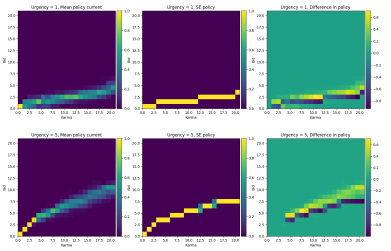
(b) Policy comparison w.r.t MFG (same exploration) for $\epsilon = 0.005$



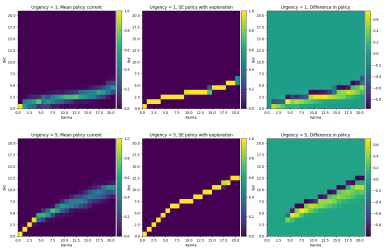
(c) Policy comparison w.r.t MFG (no exploration) for $\epsilon = 0.05$



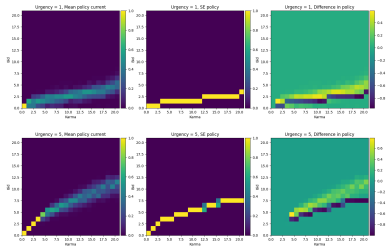
(d) Policy comparison w.r.t MFG (same exploration) for $\epsilon = 0.05$



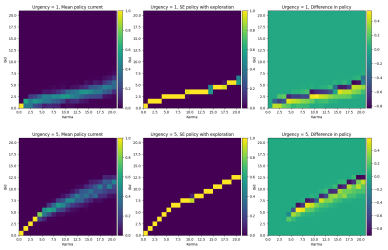
(e) Policy comparison w.r.t MFG (no exploration) for $\epsilon = 0.1$



(f) Policy comparison w.r.t MFG (same exploration) for $\epsilon = 0.1$



(g) Policy comparison w.r.t MFG (no exploration) for $\epsilon = 0.25$



(h) Policy comparison w.r.t MFG (same exploration) for $\epsilon = 0.25$

Figure A.2: Policies at Equilibrium for different exploration rates

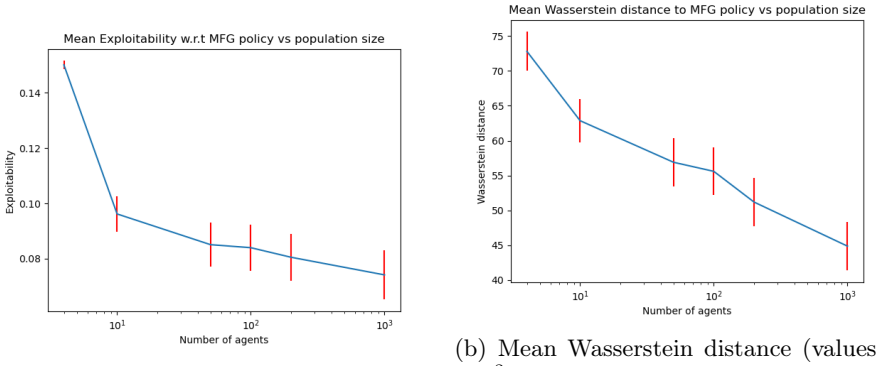
Observations

From the policy plots as well as the numerical Wasserstein distance estimations, we can see that as exploration decreases, the learned policy approaches the MFG Nash policy(zero exploration). The exploitability of the learned policy w.r.t MFG policy (no exploration) also decreases with exploration rate, since the multi-agent game becomes similar to the MFG with no exploration, which can be thought of as the limiting case. Also, both the distance and exploitability of the MARL learned policy w.r.t MFG policy with same exploration remain roughly the same w.r.t. exploration rates. This is to be expected because this MFG policy will be the Nash equilibrium for infinite population setting, and will always have a finite, constant bias w.r.t. MARL. Hence we can conclude that if given enough training time then in the limiting case of exploration going to zero, along with infinite population, MARL will converge to the fully rational MFG policy.

A.1.3 Effect of population size

In this section, we run experiments with different population sizes, from $N = 2, 4, 10, 100, 200, 1000$. Note that $N = 2$ is very different from the mean field setting since an agent exactly knows the karma of the other agent due to karma preservation. As N increases, the effect any single agent has on the environment will decrease and we expect that the policies approaches the MFG policy

Metrics

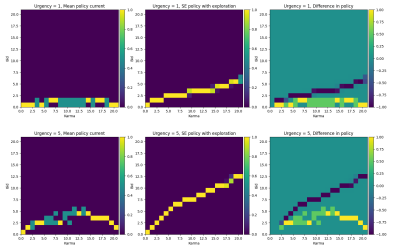


(a) Exploitability of MARL vs MFG for different population sizes

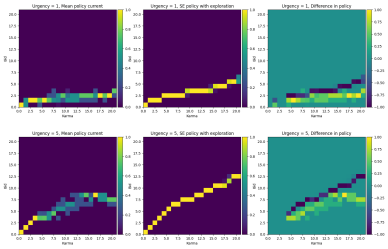
(b) Mean Wasserstein distance (values in 10^{-3}) from Chapter 3 between MARL policies and MFG policies

Figure A.3: MARL vs MFG for different population sizes

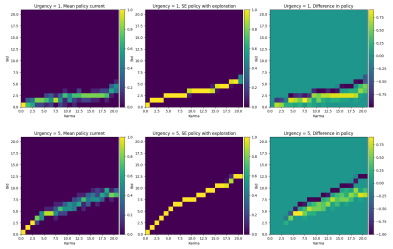
Qualitative results



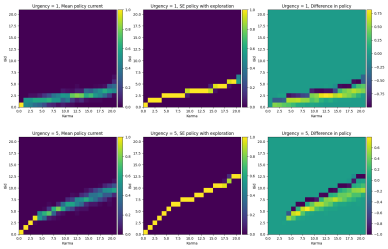
(a) Policies for $N = 2$



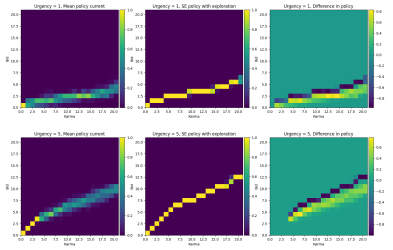
(b) Policies for $N = 4$



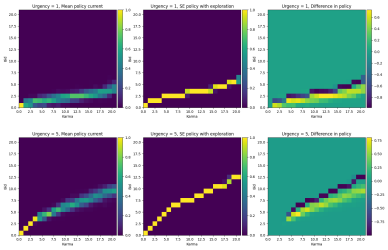
(c) Policies for $N = 10$



(d) Policies for $N = 50$



(e) Policies for $N = 100$



(f) Policies for $N = 200$

Figure A.4: Policies at Equilibrium for different population sizes

Observations

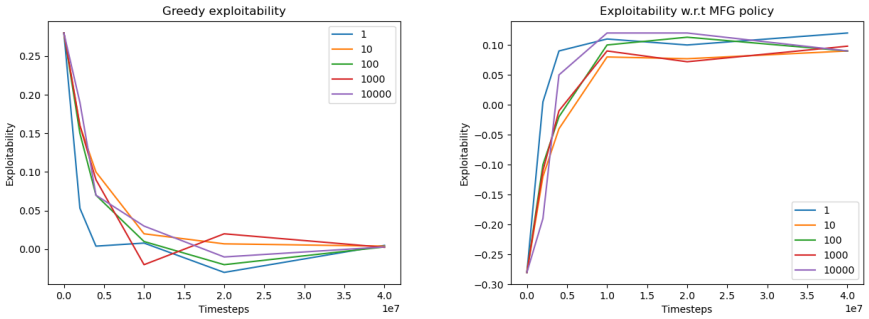
One can see that the policies start approaching the MFG equilibrium policy as the population size increases. The exploitability w.r.t this policy also decreases with population size. For distributed learning setting, there is a small bias in equilibrium policy as well as exploitability of MARL vs MFG, which scales slower than $\sqrt{N}$ for large N[18]. From this we can conclude that in the infinite population limit, MFG policy is a good indicator of what the agents will learn even if they learn in a decentralized and greedy manner. This implies that if an agent provides his urgency transitions to a central planner, then he can trust the policies given by the planner to be optimal for him individually if the population is large enough.

A.1.4 Effect of epoch length

We keep the time budget fixed and vary the epoch length, the duration in which agents agree to fix their policies.

Metrics

We plot the evolution of greedy exploitability and exploitability w.r.t. MFG for MARL simulations with different epoch lengths as follows:



(a) Evolution of Greedy exploitability for different epoch lengths

(b) Evolution of exploitability w.r.t MFG policy for different epoch lengths

Figure A.5: Comparison of convergence speeds for different epoch lengths

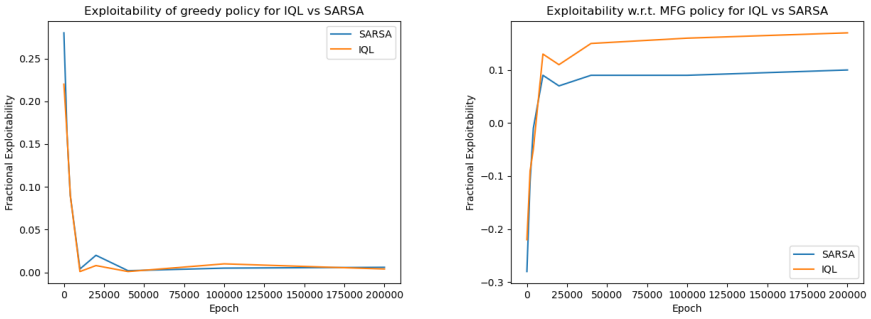
Observations

At convergence, all the simulations have very similar performance. Smaller epoch lengths have faster convergence, since they provide a chance for agents to do frequent updates to policies. Convergence to MFG is observed even in the case of epoch length being equal to one, which is a scenario one would expect in real life. Upon first glance, one would not expect convergence for epoch length of one, since this would mean that the environment is rapidly fluctuating due to agents changing their policies every timestep. But notice that the learning rate is small, which means the action value functions for agents do not change drastically, implying policies do not change drastically also. Additionally, the karma state-action space is smooth, resulting in furthering smoothing of environment updates. Overall, this experiment implies that even if agents learn independently, with no synchronisation in policy updates, they will still converge to the MFG case in large population karma games. This is important for practical application of karma games.

A.1.5 Effect of different learning algorithms

We compare performance of two learning algorithms ϵ SARSA (default algorithm used so far) to greedy independent Q Learning.

Convergence speeds



(a) Evolution of Greedy exploitability for different learning algorithms

(b) Evolution of exploitability w.r.t MFG policy for different learning algorithms

Figure A.6: Comparison of convergence speeds for different epoch lengths

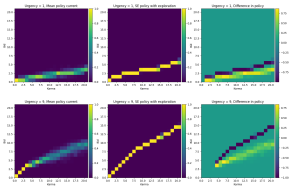
Observations

We observe that both algorithms result in convergence to the Multi-Agent Nash, as evidenced by the greedy exploitability plots. With respect to the MFG, Q learning does slightly better than ϵ SARSA. Along with previous results in epoch lengths, this implies that if agents decide to play the karma game greedily and independently, provided that they have small enough learning rates, they will converge to the MFG solution in the infinite agent limit. Hence a player in a karma game can trust the solution strategy given to him by the central planner since he cannot do much better even if he is allowed to greedily and independently learn his own behaviour.

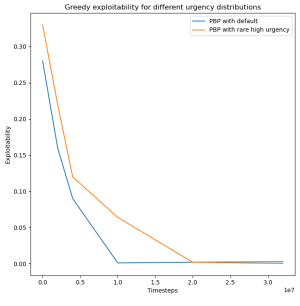
A.1.6 Robustness

We now test the behaviour of MARL in different karma games, including games with different urgency transitions, different karma mechanisms and multi-type populations.

Different urgency distribution



(a) Comparison of MARL policies with MFGs for rare igh urgency



(b) Evolution of greedy exploitability, comparison of cases with different high urgencies but same average urgencies

Metric	Value
$J(\pi_{curr}) - J(\pi_{mfg})$	0.018
$J(\pi_*) - J(\pi_{curr})$	0.00012
$d(\pi_{curr}, \pi_{se})$	0.0269
$\delta(\pi)$	0.0116

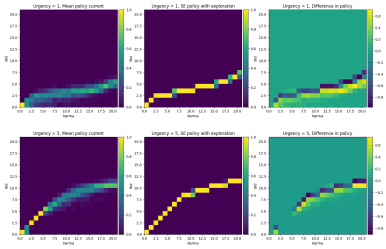
(c) Metrics at convergence for rare high uregncy case

Figure A.7: Results for MARL in rare high urgency case

We change the urgency space to be $\mathcal{U} = \{1, 9\}$, with $P(1) = 0.75$ and $P(9) = 0.25$, i.e. a karma game with a rare high urgency state. We observe that compared to the default case (which had $P(1) = P(5) = 0.5$ for urgency), this converges slower despite having same urgency. This is due to the fact that the high urgency state is rare and the agents need more timesteps in order to obtain sufficient data to plan policies. However, upon convergence, the policies are similar to the MFG as evidenced by the small exploitability.

Different karma mechanism: PBS

We use the same urgency space as the default setting but change the payment scheme to PBs, keeping rest of hyperparameters the same. Below are comparisons of learned policies to MFG and values for exploitability. One can clearly see that even in this case the MARL converges to MFG solution.



(a) Comparison of MARL policies with MFGs for PBS

Metric	Value
$J(\pi_{curr}) - J(\pi_{mfg})$	0.013
$J(\pi_*) - J(\pi_{curr})$	0.003
$d(\pi_{curr}, \pi_{mfg})$	0.0349
$\delta(\pi)$	0.0129

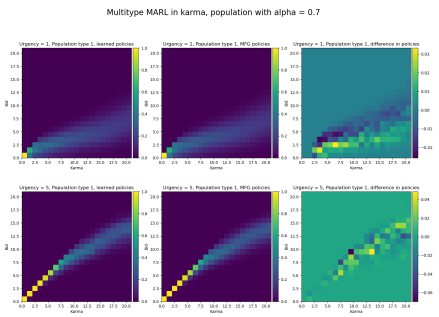
(b) Metrics at convergence for PBS case

Figure A.8: Results for MARL in PBS

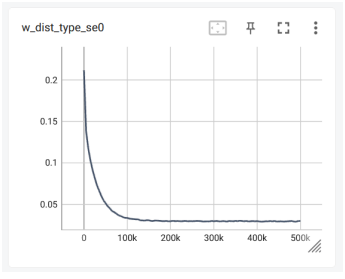
Multi-type populations

We run two experiments for multi-type populations. The first experiment consists of two population types, each with population of 100 agents but the first type having a discounting of $\gamma = 0.7$ and second type having discounting of $\gamma = 0.9$. The second experiment consists of two population types, each with population 100 but the first type has an average urgency of $\bar{u} = 3$, while second type has an average urgency of $\bar{u} = 2$.

Multi-type: Discounting rates

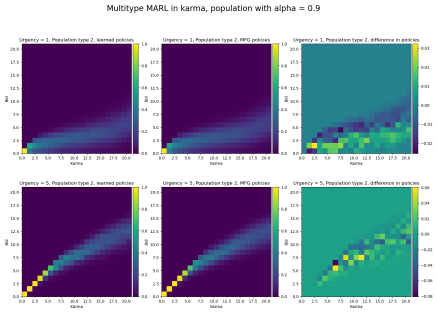


(a) Policy comparison with MFG, for multitype population with different γ, $\gamma = 0.7$

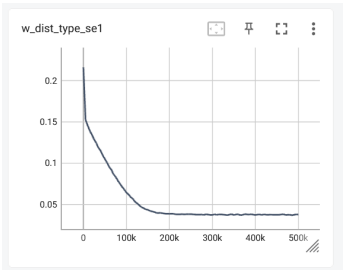


(b) Convergence of Wasserstein distance to MFG, for multitype population with different γ, $\gamma = 0.7$

Figure A.9: Comparison of policies for multitype population with different γ, $\gamma = 0.7$



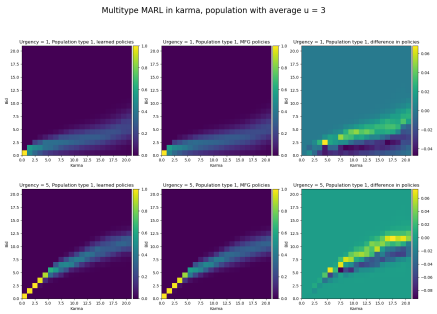
(a) Policy comparison with MFG, for multitype population with different γ, $\gamma = 0.9$



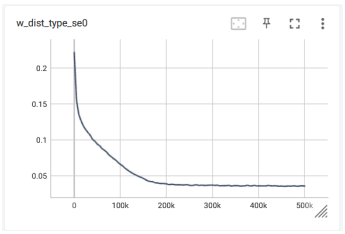
(b) Convergence of Wasserstein distance to MFG, for multitype population with different γ, $\gamma = 0.9$

Figure A.10: Comparison of policies for multitype population with different γ, $\gamma = 0.9$

Multi-type: Urgencies

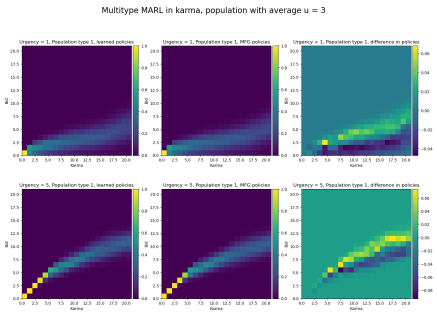


(a) Policy comparison with MFG, for multitype population with different $\bar{u}$, $\bar{u} = 3$

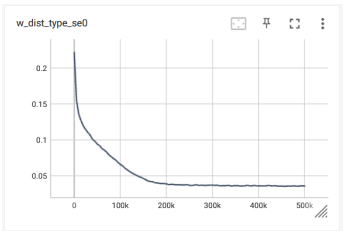


(b) Convergence of Wasserstein distance to MFG, for multitype population with different $\bar{u}$, $\bar{u} = 3$

Figure A.11: Comparison of policies for multitype population with different $\bar{u}$, $\bar{u} = 3$



(a) Policy comparison with MFG, for multitype population with different $\bar{u}$, $\bar{u} = 2$



(b) Convergence of Wasserstein distance to MFG, for multitype population with different $\bar{u}$, $\bar{u} = 2$

Figure A.12: Comparison of policies for multitype population with different $\bar{u}$, $\bar{u} = 2$

Type	Different discounting		Different urgencies	
Metric	$\gamma = 0.7$	$\gamma = 0.9$	$\bar{u} = 3$	$\bar{u} = 2$
$J(\pi_{curr}) - J(\pi_{mfg,\epsilon})$	-0.03	0.01	0.05	0.02
$J(\pi_{greedy}) - J(\pi_{curr})$	0.06	0.04	0.07	0.03

Table A.3: Metrics at the end of convergence, for experiments with multitype populations

Observations

We observe that in both experiments with multitype populations, one with different discounting rates and one with different average urgencies, we get approximate convergence to the MFG solution, both in terms of policies and in terms of exploitabilities. This proves that the solution of MARL with multitype populations will also converge to the MFG solution in large populations.

A.2 Karma game in congested city

Value/Player type	Lucky player	Unlucky Player	Average Player	Remarks
Average Bid Path 1, $u = 1$	4.03	8.94	2.14	Uncongested for Lucky and Average, Congested for Unlucky
Average Bid Path 1, $u = 5$	4.12	13.40	2.34	Uncongested for Lucky and Average, Congested for Unlucky
Average Bid Path 2, $u = 1$	3.92	8.54	3.90	Uncongested for Lucky, Congested for Unlucky and Average
Average Bid Path 2, $u = 5$	4.18	10.58	4.43	Uncongested for Lucky, Congested for Unlucky and Average
Average reward	-0.0134	-1.02	-2.22	Wide disparity in rewards

Table A.4: Empirical measurements of behaviour of lucky, unlucky and average player

Appendix B

Sublinear policies

From experiments with PBP and PBS we observe that the Optimal policy tends to be sublinear w.r.t k We lay the groundwork for proving it below

Assumption A.1: Let $\Delta b_{\min}, \Delta k_{\min} \to 0$, ie., we are operating with continuous karma and bids. Alternatively let $\bar{k} \gg 1$

Let $b = \pi(u, k, \bar{k})$ be optimal bid distribution at Nash equilibrium, for u, k and when mean korma is $\vec{k}$ Since it is a greedy policy, $\pi(u, k, \bar{k})$ is deterministic

Since Karma is a currency with no intrinsic value, and $u^+ \perp k^+ \mid u, k$, if we rescale average karma, then bids should be scaled accordingly

ie $\pi(u, \alpha k, \alpha\bar{k}) = \alpha\pi(u, k, \bar{k}) \quad \forall \alpha > 0$
Lemma A.1: If $\pi\left(u, k, \frac{\bar{k}}{\alpha}\right) \leq \pi(u, k, \bar{k}) \forall k$ and $\alpha \in (1, \infty)$

Then $\pi(u, \alpha k, \bar{k}) \leq \alpha\pi(u, k, \bar{k})$
Proof: $\pi(u, \alpha k, \bar{k}) = \pi\left(u, \alpha k, \alpha\frac{\bar{k}}{\alpha}\right) = \alpha\pi\left(u, k, \frac{\bar{k}}{\alpha}\right)$

Since $\pi\left(u, k, \frac{k}{\alpha}\right) \leq \pi(u, k, \bar{k})$,

$$\pi(u, \alpha k, \bar{k}) \leq \alpha\pi\left(u, k, \frac{\bar{k}}{\alpha}\right) \quad \forall k, \alpha > 1$$

ie. we have sublinear policies if optimal bid distribution increases if we increase mean karma in the system.

Appendix C

Code repository

The code for replicating the results as well as applying the MARL numerical simulator is available at: `https://gitlab.ethz.ch/svaishampaya/marl_karma.git`

The repository also provides relevant documentation on applying the code to various karma settings.

Bibliography

[1] Berkay Anahtarci, Can Deha Kariksiz, and Naci Saldi. Q-learning in regularized mean-field games, 2022.

[2] Kai Cui and Heinz Koeppl. Approximately solving mean field games via entropy-regularized deep reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[3] Ezzat Elokda, Saverio Bolognani, Andrea Censi, Florian Dörfler, and Emilio Frazzoli. A self-contained karma economy for the dynamic allocation of common resources. *Dynamic Games and Applications*, Apr 2023.

[4] Ezzat Elokda, Carlo Cenedese, Kenan Zhang, Andrea Censi, John Lygeros, Emilio Frazzoli, and Florian Dorfler. Carma: Fair and efficient bottleneck congestion management via non-tradable karma credits, 2023.

[5] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[6] Georgios Kotsalis, Guanghui Lan, and Tianjiao Li. Simple and optimal methods for stochastic variational inequalities, i: Operator extrapolation. *SIAM Journal on Optimization*, 32(3):2041–2073, 2022.

[7] Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Japanese Journal of Mathematics*, 2(1):229–260, Mar 2007.

[8] Mathieu LauriÃ¨re, Sarah Perrin, Julien PÃ©rolat, Sertan Girgin, Paul Muller, Romuald Ã‹lie, Matthieu Geist, and Olivier Pietquin. Learning in mean field games: A survey, 2024.

[9] Weichao Mao, Haoran Qiu, Chen Wang, Hubertus Franke, Zbigniew Kalbarczyk, Ravishankar Iyer, and Tamer Basar. A mean-field game approach to cloud resource management with function approximation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36243–36258. Curran Associates, Inc., 2022.

[10] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[11] Julien Perolat, Remi Munos, Jean-Baptiste Lespiau, Shayegan Omidshafiei, Mark Rowland, Pedro Ortega, Neil Burch, Thomas Anthony, David Balduzzi, Bart De Vylder, Georgios Piliouras, Marc Lanctot, and Karl Tuyls. From poincare recurrence to convergence in imperfect information games: Finding equilibrium via regularization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8525–8535. PMLR, 18–24 Jul 2021.

[12] Julien Perolat, Sarah Perrin, Romuald Elie, Mathieu Lauriere, Georgios Piliouras, Matthieu Geist, Karl Tuyls, and Olivier Pietquin. Scaling mean field games by online mirror descent. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pages 1028–1037, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems.

[13] Sarah Perrin, Julien Perolat, Mathieu Lauriere, Matthieu Geist, Romuald Elie, and Olivier Pietquin. Fictitious play for mean field games: Continuous time analysis and applications. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13199–13213. Curran Associates, Inc., 2020.

[14] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, 2018.

[15] Muhammed Sayin, Kaiqing Zhang, David Leslie, Tamer Basar, and Asuman Ozdaglar. Decentralized q-learning in zero-sum markov games. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 18320–18334. Curran Associates, Inc., 2021.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[17] Ming Tan. Multi-agent reinforcement learning: Independent versus cooperative agents. In *International Conference on Machine Learning*, 1997.

[18] Batuhan Yardim, Semih Cayci, Matthieu Geist, and Niao He. Policy mirror ascent for efficient and independent learning in mean field games. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 39722–39754. PMLR, 23–29 Jul 2023.

[19] Fengzhuo Zhang, Vincent Y. F. Tan, Zhaoran Wang, and Zhuoran Yang. Learning regularized monotone graphon mean-field games, 2023.

[20] K. Zhang, Zhuoran Yang, and Tamer Basar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *ArXiv*, abs/1911.10635, 2019.